Model Order Selection for Boolean Matrix Factorization

Pauli Miettinen
Max Planck Institute for Informatics
Saarbrücken, Germany
pmiettin@mpi-inf.mpg.de

Jilles Vreeken
Dept. of Mathematics and Computer Science
Universiteit Antwerpen, Belgium
jilles.vreeken@ua.ac.be

ABSTRACT

Matrix factorizations—where a given data matrix is approximated by a product of two or more factor matrices—are powerful data mining tools. Among other tasks, matrix factorizations are often used to separate global structure from noise. This, however, requires solving the 'model order selection problem' of determining where fine-grained structure stops, and noise starts, i.e., what is the proper size of the factor matrices.

Boolean matrix factorization (BMF)—where data, factors, and matrix product are Boolean—has received increased attention from the data mining community in recent years. The technique has desirable properties, such as high interpretability and natural sparsity. But so far no method for selecting the correct model order for BMF has been available. In this paper we propose to use the Minimum Description Length (MDL) principle for this task. Besides solving the problem, this well-founded approach has numerous benefits, e.g., it is automatic, does not require a likelihood function, is fast, and, as experiments show, is highly accurate.

We formulate the description length function for BMF in general—making it applicable for any BMF algorithm. We extend an existing algorithm for BMF to use MDL to identify the best Boolean matrix factorization, analyze the complexity of the problem, and perform an extensive experimental evaluation to study its behavior.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications–*Data mining*

General Terms

Theory, Algorithms, Experimentation

1. INTRODUCTION

A typical task in data mining is to find observations and variables that behave similarly. Consider, for instance, the standard example of supermarket basket data. We are given transactions over items, and we want to find groups of transactions and groups of items such that we can (approximately) represent our data in terms of these groups, instead of the original transactions. Such representation is called a low-dimensional representation of the data, and is usually obtained using some form of matrix factorization.

In matrix factorizations the input data (represented as a matrix) is decomposed into two (or more) factor matrices. Usually the aim is to have low-dimensional factor matrices whose product approximates the original matrix well. By imposing different constraints, one obtains different factorizations. Perhaps the two best-known factorizations are Singular Value Decomposition (SVD), closely related to Principal Component Analysis (PCA), and Non-negative Matrix Factorization (NMF). SVD and PCA restrict the factor matrices to be orthogonal, while NMF requires the data and the factor matrices to be non-negative.

When the input data is Boolean, (that is, contains only 0s and 1s, as is typical with supermarket basket data), one can apply Boolean Matrix Factorization (BMF). Similarly to NMF, it restricts the factor matrices for added interpretability and sparsity. In BMF, the factor matrices are required to be Boolean, i.e., contain only 0s and 1s. Also the matrix product is changed, from normal to Boolean. As a consequence, it is possible that BMF obtains smaller reconstruction error than SVD for the same decomposition size—something that NMF, by definition, cannot do [20]. Furthermore, it can be shown that for sparse Boolean matrices, there is always a sparse exact factorization [21].

But no matter what factorization method one applies, one always has to solve the model order selection problem: what is the correct number of latent dimensions? In some situations the answer is obvious, for example, if a user wants to have a three-dimensional representation of the data (say, for visualization). But when the user wants a good description of the structure in the data, selecting the number of latent dimensions comes down to the question: what is structure and what is noise.

Whereas various methods have been proposed to answer this question for non-Boolean matrix factorization, varying from statistical methods based on likelihood scores (such as the Bayesian Information Criterion, BIC) to subjective analysis of error (so-called elbow methods), there is no known applicable method for selecting the model order for BMF (other than visual analysis of errors).

In this paper, we study the model order selection problem in the framework of BMF. To that end, we merge two orthogonal lines of research, namely those of Boolean matrix factorizations and Minimum Description Length (MDL) principle. We formulate description length functions that can be used for model order selection with any BMF algorithm. We then extend an existing algorithm for BMF to use MDL in an effective way, and via extensive experimental evaluation we show that using our description length formulation, the algorithm is able to identify the correct number of latent dimensions in synthetic and real-world BMF tasks.

KDD'11, August 21–24, 2011, San Diego, California, USA.

2. RELATED WORK

Matrix factorization methods such as Singular Value Decomposition (SVD) [13] or Non-negative Matrix Factorization (NMF) [29] are ubiquitous in data mining and machine learning. Two of the most popular uses for matrix factorizations are separating structure and noise, and learning missing values of matrices.

Boolean matrix factorizations have been studied extensively in combinatorics (see, e.g., [25] and references therein). The use of Boolean factorizations in data mining was proposed in [22], and they are related to many other data mining methods (see below for some related work and [20] for more). Outside data mining, Boolean factorizations have found application in, e.g., finding roles for access control [33, 36].

The Asso algorithm to solve BMF was proposed by Miettinen et al. [22]. Later, Lu et al. [19] proposed a heuristic based on a mixed-integer-programming formulation. Independently, Belohlavek and Vychodil [1] gave an algorithm for computing the Boolean rank of a matrix based on solving the Set Cover problem. At worst case this algorithm can take exponential time and does not guarantee any non-trivial approximation factor, but recently it was shown that with certain sparsity constraints, the algorithm runs in polynomial time and provides a logarithmic approximation guarantee [21].

Matrix factorizations have a long history in various fields of science. SVD and its close relative PCA have been of particular importance. Hence, it is no surprise that many methods for model order selection for these two decompositions have been proposed. One of the earliest suggestions was the Guttman–Kaiser criterion, dating back to the Fifties (see [41]). In that criterion, one selects those principal vectors that have corresponding principal value greater than 1. It is perhaps not surprising that this simple criterion has shown to perform poorly [41]. Another often-used method is Cattell's scree test [2], where one selects the point where the ratio between two consecutive singular values (in descending order) is high. Usually, this is done by visual analysis, but automated methods have also been proposed (e.g., [42]).

Since these two classical methods, researchers have proposed many alternative methods. For example, in a probabilistic framework one can use Bayesian model selection (e.g. [23, 32]). For BMF, however, it would be very hard, if not impossible, to construct a good likelihood function. Yet another approach is to use cross validation. While this is perhaps mostly used when learning missing values of the matrix, it can also applied to the noise removal. Assumption is that when the model order is too high, the factors start to specialize to noise, and hence, the cross-validation error increases. Normally, hold-out set would contain either rows or columns, but not both. Recently, Owen and Perry [28] proposed a method to leave out a sub-matrix. The method is based on the assumption that the remaining matrix has the same rank as the original matrix, as this is needed to fit the factors to the test data.

The concept of intrinsic dimension of the data is related to the model order. While often the intrinsic dimension refers to the number of variables needed to explain the data completely (e.g., the rank of a matrix), also noise-invariant approaches have been studied [30]. Tatti et al. [34] defined intrinsic dimensionality to Boolean data based on fractal dimensions.

As discussed by Faloutsos and Megalooikonomou [8], the Minimum Description Length principle [14] is a powerful, well-founded, and natural approach to data mining, as it allows us to clearly identify the most succinct and least redundant model for a dataset. As such, MDL has been successfully employed for a wide range of data mining tasks, including, for example, discretization [9], imputation [37], and clustering [4, 16].

Basically, a Boolean matrix factorization gives us a group of patterns (the left-hand matrix $\mathbf{B}$) and their occurrences (the right-hand matrix $\mathbf{C}$). As such, BMF essentially describes the data with a *set of patterns*. Therefore, pattern set mining techniques are related.

KRIMP [38] pioneered the use of MDL for identifying good pattern sets, and selects that group of frequent itemsets that describes the data best. LESS [15] and PACK [35] follow a similar approach, but respectively describe data using low-entropy sets and decision trees. A major difference between these methods and BMF is that rows are only covered using subsets of that row, and approximate matches are not allowed. Further, KRIMP and LESS do not allow overlap between patterns covering the same row. All three typically return many more, and much more specific, patterns than BMF.

Summarization, proposed by Chandola et al. [3], is a compression-based approach that identifies a group of itemsets such that each transaction is summarized by one itemset with as little loss of information as possible. Wang and Parthasararthy [40] find summary sets, i.e., sets of itemsets such that each transaction is (partially) covered by the largest itemset that is frequent. In BMF, however, we do not require every row to be modeled by at least one factor.

More closely related to BMF is Tiling [12], which essentially employs the well known greedy set-cover algorithm to iteratively cover the data with that itemset that covers the most uncovered 1s in the data. Kontonasios and De Bie [17] iteratively discover the most interesting 'noisy tile', where they define interestingness through a local MDL score. Unlike our situation, it does not return a model for the data, but rather orders a given collection of itemsets.

3. NOTATION

Before we introduce the theory behind our approach, we introduce the notation we will use throughout the paper.

Throughout this paper, we identify the datasets with Boolean matrices. Matrices are denoted by upper-case bold letters ($\mathbf{A}$). Vectors are lower-case bold letters ($\mathbf{a}$). If $\mathbf{A}$ is an n-by-m Boolean matrix, $|\mathbf{A}|$ denotes the number of 1s in it, i.e., $|\mathbf{A}| = \sum_{i,j} a_{ij}$. We extend the same notation to Boolean vectors. The scalar product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted as $\langle \mathbf{x}, \mathbf{y} \rangle$.

Let $\mathbf{X}$ be n-by-k and $\mathbf{Y}$ be k-by-m Boolean matrices (i.e., $\mathbf{X}$ and $\mathbf{Y}$ take values from $\{0, 1\}$). Their *Boolean matrix product*, $\mathbf{X} \circ \mathbf{Y}$, is the Boolean matrix $\mathbf{Z}$ with $z_{ij} = \bigvee_{l=1}^{k} x_{il} y_{lj}$.

If $\mathbf{X}$ and $\mathbf{Y}$ are two n-by-m Boolean matrices, we have the following element-wise matrix operations. The *Boolean sum* $\mathbf{X} \vee \mathbf{Y}$ is the normal matrix sum with addition defined as $1 + 1 = 1$. The *Boolean subtraction* $\mathbf{X} \ominus \mathbf{Y}$ is the normal element-wise subtraction with $0 - 1 = 0$. Notice that this does not define an inverse of Boolean sum, as $1 + 1 - 1 = 0$. The *Boolean element-wise product* $\mathbf{X} \wedge \mathbf{Y}$ is defined as normal element-wise matrix product. The *exclusive or* $\mathbf{X} \oplus \mathbf{Y}$ is the normal matrix sum with addition defined as $1 + 1 = 0$ (i.e., addition is done over the field $\mathbb{Z}_2$).

The *Boolean rank* of an n-by-m Boolean matrix $\mathbf{A}$, $\operatorname{rank}_B(\mathbf{A})$, is the least integer k such that there exists an n-by-k Boolean matrix $\mathbf{B}$ and a k-by-m Boolean matrix $\mathbf{C}$ for which $\mathbf{A} = \mathbf{B} \circ \mathbf{C}$.

Matrices $\mathbf{B}$ and $\mathbf{C}$ are *factor matrices* of $\mathbf{A}$, and the pair $(\mathbf{B}, \mathbf{C})$ is the (approximate) *Boolean factorization* of $\mathbf{A}$. Lastly, all logarithms are of base 2, and we employ the usual convention that $0 \log 0 = 0$.

4. BOOLEAN MATRIX FACTORIZATION

In this section we introduce Boolean matrix factorization (BMF) and give a short description of Asso, one of the existing algorithms for Boolean matrix factorization.

4.1 BMF, a brief primer

In Boolean matrix factorization, the goal is to (approximately) represent a Boolean matrix as the Boolean product of two Boolean matrices. The crux is the Boolean product: as the product is not over a field, but over a semiring $(0, 1, \vee, \wedge)$, Boolean matrix factorizations have some unique properties. For example, the Boolean rank of a matrix $\mathbf{A}$ can be only a logarithm of the normal matrix rank of $\mathbf{A}$ [25]. As a consequence, Boolean factorizations can yield smaller reconstruction error than factorizations of same size done under the normal arithmetic. Unfortunately, unlike normal rank, computing the Boolean rank is NP-hard [27], and even approximation is hard [22] (although recent work shows that logarithmic approximations can be obtained by assuming sparsity [21]).

But even assuming we could compute the Boolean rank efficiently, this is rarely what we actually want. Similarly to normal rank, one would assume that most of the real-world data matrices have full or almost full Boolean rank, due to noise; instead, we often want to have a low-rank approximation of a matrix. Such approximation is usually interpreted to contain the latent structure of the data, while the error it causes is regarded as the noise. When the target rank is given, we have the Boolean matrix factorization problem:

PROBLEM 1 (BMF). *Given n-by-m Boolean matrix $\mathbf{A}$ and integer k, find n-by-k Boolean matrix $\mathbf{B}$ and k-by-m Boolean matrix $\mathbf{C}$ such that $\mathbf{B}$ and $\mathbf{C}$ minimize*

$$|\mathbf{A} \oplus (\mathbf{B} \circ \mathbf{C})| . \tag{1}$$

Unsurprisingly, also this optimization problem is NP-hard, and has strong inapproximability results in terms of multiplicative and additive errors (see [20]). But there is also another, more fundamental problem: the formulation of the BMF problem requires us to have *a priori* knowledge on k, the Boolean rank of the decomposition. With the structure/noise interpretation above, this means that we have to have *a priori* knowledge of the dimensionality of the latent structure—something we most likely do not have.

This problem is, by no means, unique to BMF. Indeed, the same issue underlies any matrix factorization method. And also in clustering, for example, we have to deal with the same problem, known as the model order selection problem. The main contribution of this paper is to provide a method to (approximately) solve the model order selection problem in the BMF framework.

4.2 The Asso Algorithm

Knowing the latent dimensionality of the data is usually not enough—we also want to know the latent factors, i.e., we want to solve the BMF problem. As the problem is NP-hard, even to approximate well, we will solve it using a heuristic approach. We have opted to use an existing algorithm for BMF, called Asso [22]. We chose to use Asso as previous studies have shown it performs reasonably well [21, 22, 33], and because the algorithm is hierarchical, i.e., the rank-$(k-1)$ decomposition gives the first $k-1$ columns of $\mathbf{B}$ (and first $k-1$ rows of $\mathbf{C}$) of rank-k decomposition. The latter property is particularly useful when doing the model order selection, as we will see later. We emphasize, though, that the proposed model order selection method is not bound to any specific algorithm for BMF.

For the sake of completeness, we provide a quick explanation of how Asso works. For more detailed explanation, see [20, 22]. The name of Asso stems from it using pairwise association accuracies to generate so-called candidate columns. More precisely, Asso generates an n-by-n matrix $\mathbf{X} = (x_{ij})$ with $x_{ij} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle / \langle \mathbf{a}_j, \mathbf{a}_j \rangle$ where $\mathbf{a}_i$ is the jth row of $\mathbf{A}$. That is, x_{ij} is the association accuracy for rule $\mathbf{a}_j \Rightarrow \mathbf{a}_i$.

The columns of $\mathbf{B}$ are selected from the columns of $\mathbf{X}$, after the latter have been rounded to Boolean columns. The threshold t for the rounding is a user-specified parameter. The selection of columns of $\mathbf{B}$ happens in a greedy fashion: each not-used column of rounded $\mathbf{X}$ is tried, and the selected column is the one that maximizes the gain, defined being the number of newly-covered 1s of $\mathbf{A}$ minus the number of newly-covered 0s of $\mathbf{A}$. Element a_{ij} is newly-covered if $(\mathbf{B} \circ \mathbf{C})_{ij} = 0$ before adding the new column to $\mathbf{B}$. The row of $\mathbf{C}$ corresponding to the column of $\mathbf{B}$ is build using the same technique: if the gain of using the new column of $\mathbf{B}$ to cover a column of $\mathbf{A}$ is positive, then the corresponding element of the new row of $\mathbf{C}$ is set to 1; otherwise it is 0.

As Asso never tracks back its decisions, it clearly has the desired hierarchical property. But Asso also requires the user to set an extra parameter: the rounding threshold t. Selecting this parameter can be daunting, as it is hard to anticipate the difference it makes to the factorization. To solve this problem, we will use our model order selection mechanism to slightly larger question of *model selection* and in addition to selecting the best k, we also select the best t.

5. MDL FOR BMF

In this section we give our approach for selecting model orders for BMF by the Minimum Description Length principle.

5.1 MDL, a brief primer

The MDL (Minimum Description Length) [14, 31] principle, like its close cousin MML (Minimum Message Length) [39], is a practical version of Kolmogorov complexity [18]. All three embrace the slogan *Induction by Compression*. For MDL, this principle can be roughly described as follows.

Given a set of models[1] $\mathcal{H}$, the best model $H \in \mathcal{H}$ is the one that minimizes

$$L(H) + L(D \mid H)$$

in which $L(H)$ is the length, in bits, of the description of H, and $L(D \mid H)$ is the length, in bits, of the description of the data when encoded with H.

This is called two-part MDL, or *crude* MDL. As opposed to *refined* MDL, where model and data are encoded together [14]. We use two-part MDL because we are specifically interested in the compressor: the factorization that yields the best compression. Further, although refined MDL has stronger theoretical foundations, it cannot be computed except for some special cases. Note that MDL requires the compression to be *lossless* in order to allow for fair comparison between different $H \in \mathcal{H}$.

To use MDL, we have to define what our models $\mathcal{H}$ are, how a $H \in \mathcal{H}$ describes a database, and how all of this is encoded in bits. Note, however, that with MDL we are only interested in the length of the description, and not in the encoded data itself. That is, we are only concerned with the length of the used codes, and do not have to materialize the codes themselves.

5.2 Encoding BMF

We now proceed to define how we can use MDL to identify the best Boolean factorization $(\mathbf{B}, \mathbf{C})$ for a given dataset $\mathbf{A}$.

Recall that an essential requirement of MDL is that the encoding is lossless. That is, whether or not a factorization $(\mathbf{B}, \mathbf{C}) \in \mathcal{H}$ for $\mathbf{A}$ is exact, we need to be able to reconstruct $\mathbf{A}$ without loss. We do this by explicitly encoding the difference, or error, between the original data $\mathbf{A}$ and its approximation as given by the Boolean product of its factor matrices $\mathbf{B}$ and $\mathbf{C}$, i.e., $\mathbf{B} \circ \mathbf{C}$. That is, we

[1] MDL-theorists talk about *hypothesis* in this context, hence the $\mathcal{H}$.

define error matrix $\mathbf{E}$ for $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, to be the unique Boolean matrix of dimensions n-by-m such that

$$\mathbf{E} = \mathbf{A} \oplus (\mathbf{B} \circ \mathbf{C}) \,. \tag{2}$$

Vice versa, when we are given matrices $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{E}$ we can reconstruct $\mathbf{A}$ without loss. Then, the total compressed size $L(\mathbf{A}, H)$, in bits, for a Boolean dataset $\mathbf{A}$ and a Boolean matrix factorization $H = (\mathbf{B}, \mathbf{C})$, with $H \in \mathcal{H}$, is defined as

$$L(\mathbf{A}, H) = L(H) + L(\mathbf{E}) \,, \tag{3}$$

where $\mathbf{E}$ follows from $\mathbf{A}$ and H, using Eq. 2. Following the MDL principle, the best factorization for $\mathbf{A}$ is found by minimizing Eq. 3. As we will discuss later, this is not as simple as it sounds. But let us first discuss how we encode H and $\mathbf{E}$, or most importantly, how many bits this requires.

We start by defining how to compute the number of bits required for a factorization $H = (\mathbf{B}, \mathbf{C})$, of dimensions n-by-k and k-by-m, for $\mathbf{B}$ and $\mathbf{C}$ respectively, as

$$L(H) = L(n) + L(m) + L(k) + L(\mathbf{B}) + L(\mathbf{C}) \,. \tag{4}$$

That is, we encode the dimensions n, m, k, and then the content of the two factor matrices. By explicitly encoding the dimensions of the matrices, we can subsequently encode matrices $\mathbf{B}$ and $\mathbf{C}$ using an optimal prefix code [5]. To encode m and k, we use the Elias-Delta code, which is a Universal code for integers [5]. A Universal code is a code that can be decoded unambiguously without requiring the decoder to have any background information, but for which the expected length of the code words are within a constant factor of the true optimal code [14]. Elias-Delta coding requires

$$L(x) = \log(x) + \log\log(x) \tag{5}$$

bits to encode an integer x. Note that, as desired, $L(n) + L(m)$ is constant between any $(\mathbf{B}, \mathbf{C}) \in \mathcal{H}$ with $\mathbf{B} \circ \mathbf{C}$ of dimensions n-by-m. (Which is the case when we regard Boolean matrix factorizations of an n-by-m matrix $\mathbf{A}$.)

As we want the selection for the best factorization to depend strictly on the structure of the factorization and the error it introduces —and do not want to introduce any bias to small k by the encoding of the value of k—we do not encode k using the ED code (Eq. 5). Instead, we use a fixed number of bits, i.e., block-encoding, which gives us

$$L(k) = \log(\min(m, n)) \,. \tag{6}$$

This gives us a number of bits in which we can encode values for k up to the minimum of m and n. Note that larger values do not make sense in BMF, as for $k = \min(m, n)$ there already is a trivial factorization $(\mathbf{B}, \mathbf{C})$ of $\mathbf{A}$ with $\mathbf{B} = \mathbf{A}$ and $\mathbf{C}$ the identity-matrix, or vice-versa.

With the dimensions encoded, we continue by encoding the factor matrices $\mathbf{B}$ and $\mathbf{C}$. To not introduce bias between different factors, these are encoded per factor. That is, we encode $\mathbf{B}$ per row and $\mathbf{C}$ per column. For transmitting the Boolean values of each factor we use an optimal prefix code, for which Shannon entropy, $-\log P(x)$, gives us the optimal code lengths. In order to use this optimal code, we need to first encode the probability $p_1^{\mathbf{b}_i} = P(1 \mid \mathbf{b}_i) = |\mathbf{b}_i|/n$ of encountering 1 in column i of $\mathbf{B}$. As the maximum number of 1s in a column of $\mathbf{B}$ is n, we need k times $\log(n)$ bits to encode these probabilities. This gives us, for $\mathbf{B}$ of n-by-k,

$$L(\mathbf{B}) = k \log(n) + \sum_{i=1}^{k} \left(|\mathbf{b}_i| l_1^{\mathbf{b}_i} + (n - |\mathbf{b}_i|) l_0^{\mathbf{b}_i} \right) \tag{7}$$

in which $k \log(n)$ is the number of bits to transmit the number of 1s in each column vector $\mathbf{b}_i$, and

$$l_1^{\mathbf{b}_i} = -\log(\frac{|\mathbf{b}_i|}{n}) \quad \text{and} \quad l_0^{\mathbf{b}_i} = -\log(\frac{n - |\mathbf{b}_i|}{n})$$

are the optimal prefix code lengths for 1 and 0, respectively, for vector $\mathbf{b}_i$ corresponding to the ith column of $\mathbf{B}$.

Analogously, we encode $\mathbf{C}$, of k-by-m, per row and have

$$L(\mathbf{C}) = k \log(m) + \sum_{j=1}^{k} \left(|\mathbf{c}_j| l_1^{\mathbf{c}_j} + (m - |\mathbf{c}_j|) l_0^{\mathbf{c}_j} \right) \tag{8}$$

with

$$l_1^{\mathbf{c}_j} = -\log(\frac{|\mathbf{c}_j|}{m}) \quad \text{and} \quad l_0^{\mathbf{c}_j} = -\log(\frac{m - |\mathbf{c}_j|}{m})$$

where $\mathbf{c}_j$ corresponds the jth row of $\mathbf{C}$.

With the above definitions we now have all elements to calculate $L(H)$. By H, the receiver knows $\mathbf{B}$ and $\mathbf{C}$, and only needs $\mathbf{E}$ to be able to lossless reconstruct $\mathbf{A}$. We will now discuss four increasingly involved alternatives for encoding $\mathbf{E}$; we will explore their quality experimentally in Section 6.

5.2.1 *Encoding* $\mathbf{E}$*: Naïve Factors*

As we are scoring Boolean matrix factorizations, it would be natural to also encode the entries of $\mathbf{E}$ as a factorization, e.g., such that $\mathbf{E} = \mathbf{F} \circ \mathbf{G}$. Of course, we have to keep in mind that $\mathbf{B}$ and $\mathbf{C}$ encode the structure in $\mathbf{A}$, whereas $\mathbf{E}$ encodes the noise, and noise by definition is unstructured. Hence, we have to encode each 1 in $\mathbf{E}$ in a separate factor, i.e., separate columns/rows in $\mathbf{F}$ and $\mathbf{G}$. The amount of bits this requires is given by

$$L_f(\mathbf{E}) = \log mn + |\mathbf{E}| \Big(-(n-1)\log\left(\frac{n-1}{n}\right) - \log\left(\frac{1}{n}\right) - (m-1)\log\left(\frac{m-1}{m}\right) - \log\left(\frac{1}{m}\right)\Big) \tag{9}$$

in which we essentially use the same encoding for the factors as in Eq. 7 and 8 (but with the extra knowledge that every row/column in the factor matrices contains only one 1). We refer to this encoding as the Naïve Factor encoding for $\mathbf{E}$.

Quick analysis of this encoding tells us it is monotonically increasing for larger error, which is good, but also that we are spending too many bits, as we essentially encode full rows and columns, whereas we only want to transmit the locations of the 1s in $\mathbf{E}$.

5.2.2 *Encoding* $\mathbf{E}$*: Naïve Indices*

This observation suggests that we should simply transmit the coordinates of the 1s in $\mathbf{E}$. Clearly, this takes $\log m + \log n$ bits per entry. Then,

$$L_i(\mathbf{E}) = |\mathbf{E}| \left(\log m + \log n\right) \,, \tag{10}$$

gives us the total cost for transmitting $\mathbf{E}$. We refer to this encoding as Naïve Indices.

It saves us bits compared to Naïve Factors, albeit a marginal amount, and hence by MDL it is a better encoding. Further, it is monotonically increasing with the amount of 1s in $\mathbf{E}$.

5.2.3 *Encoding* $\mathbf{E}$*: Naïve Exclusive-Or*

Although perhaps counter-intuitive, typically we can encode $\mathbf{E}$ more succinctly if we transmit the whole matrix instead of just the 1s. That is, we can save bits by transmitting, in a fixed order, and using an optimal prefix code, not only the 1s but also the 0s.

We do this by first transmitting the number of 1s in $\mathbf{E}$, i.e., $|\mathbf{E}|$, in $\log mn$ bits, which allows the receiver to calculate the probability $p_1^{\mathbf{E}} = \frac{|\mathbf{E}|}{mn}$, and hence, the optimal prefix codes for 1 and 0. Then, using these codes, and in a fixed order, we transmit the value of every cell of $\mathbf{E}$. The total number of bits required by this approach, which we refer to as Naïve XOR, is given by

$$L_n(\mathbf{E}) = \log mn + |\mathbf{E}|\, l_1 + (mn - |\mathbf{E}|) l_0 \ , \qquad (11)$$

where the lengths of the codes for 1 and 0 respectively are

$$l_1 = -\log p_1^{\mathbf{E}} \quad \text{and} \quad l_0 = -\log(1 - p_1^{\mathbf{E}}) \ .$$

By this approach, we consider every cell in $\mathbf{E}$ independently, yet, importantly, with regard to $p_1^{\mathbf{E}}$. This means that L_n is not strictly monotonic with regard to the number of 1s in $\mathbf{E}$, as once $p_1^{\mathbf{E}} > (1 - p_0^{\mathbf{E}})$, adding 1s to $\mathbf{E}$ decreases its cost. In practice, however, this is not a problem, as it only occurs if $\mathbf{A}$ is both extremely large and extremely dense, yet contains so little structure that encoding it in factors costs more bits than one gains. Besides a pathological case, this situation could be avoided by spending 1 extra bit to indicate whether we are encoding $\mathbf{E}$ or its Boolean inverse—a cost that is dwarfed by the total number of bits to describe $\mathbf{E}$.

5.2.4 *Encoding* $\mathbf{E}$*: Typed Exclusive-Or*

Our last refinement is to differentiate between noise in the part of $\mathbf{E}$ that falls within the modeled part of $\mathbf{A}$, i.e., the 1s we have modeled but do not occur in $\mathbf{A}$, $\mathbf{E}^- = \mathbf{E} \wedge (\mathbf{B} \circ \mathbf{C})$, and those 1s that are part of $\mathbf{A}$ but not included in the model, i.e., $\mathbf{E}^+ = \mathbf{E} \ominus (\mathbf{B} \circ \mathbf{C})$. Trivially, this gives us $\mathbf{E} = \mathbf{E}^+ \vee \mathbf{E}^-$. We refer to this approach as Typed XOR.

We encode each of these two parts analogously to Naïve XOR, but can transmit the number of 1s in the additive and subtractive parts in respectively $\log(mn - |\mathbf{B} \circ \mathbf{C}|)$ and $\log |\mathbf{B} \circ \mathbf{C}|$ bits. We define the probability of a 1 in $\mathbf{E}^+$ as $p_1^+ = \left|\mathbf{E}^+\right| / (mn - |\mathbf{B} \circ \mathbf{C}|)$, and similarly for $\mathbf{E}^-$, $p_1^- = \left|\mathbf{E}^-\right| / |\mathbf{B} \circ \mathbf{C}|$. When we combine this, we can calculate the number of bits required to encode $\mathbf{E}$ by the Typed XOR encoding as

$$L_x(\mathbf{E}) = L(\mathbf{E}^+) + L(\mathbf{E}^-) \qquad (12)$$

where

$$\begin{aligned} L(\mathbf{E}^+) = \log(mn - |\mathbf{B} \circ \mathbf{C}|) + \left|\mathbf{E}^+\right| l_1^+ \\ + (mn - |\mathbf{B} \circ \mathbf{C}| - \left|\mathbf{E}^+\right|) l_0^+ \end{aligned}$$

and

$$L(\mathbf{E}^-) = \log(|\mathbf{B} \circ \mathbf{C}|) + \left|\mathbf{E}^-\right| l_1^- + (|\mathbf{B} \circ \mathbf{C}| - \left|\mathbf{E}^-\right|) l_0^-$$

respectively give the number of bits to encode the additive and subtractive parts of $\mathbf{E}$. We calculate l_1^+, l_0^+, l_1^-, l_1^- analogous to how we calculate l for Naïve XOR.

Like Naïve XOR, in general this encoding is monotonically increasing for larger error $|\mathbf{E}|$. However, it is more efficient than its naïve cousin when the noise is not uniformly distributed over the modeled and not modeled parts of $\mathbf{A}$. That is, when the probability of a 1 being part of a true pattern being recorded as a 0 is not equal to the probability of a true 0 being recorded as a 1. Clearly, in many situations this may be the case.

By making a choice for one of the four encoding strategies of $\mathbf{E}$, we can now calculate the total compressed size $L(\mathbf{A}, H)$ for a dataset $\mathbf{A}$ and its factorization. As out of the four strategies, Typed XOR compresses $\mathbf{E}$ most efficiently given the available information, we expect it to be the best choice for identifying the correct model order. In Section 6 we will empirically evaluate performance, but first we discuss the complexity of finding the factorization that minimizes $L(\mathbf{A}, H)$.

5.3 Computational Complexity

Finding the minimum description length Boolean matrix factorization is a computationally hard task. To start with, the shortest encoding corresponds to the Kolmogorov complexity, which is non-computable. But even when we try to minimize some given encoding, like one of the above, the problem does not necessarily become easy. In particular, we cannot have a polynomial-time algorithm that minimizes the description length in *any* given encoding.

PROPOSITION 1. *Unless* $\mathrm{P} = \mathrm{NP}$*, there exists no polynomial-time algorithm that, given an encoding length function* L *and an* n*-by-*m *Boolean matrix* $\mathbf{A}$*, finds Boolean matrices* $\mathbf{B}$ *and* $\mathbf{C}$ *such that* $L(\mathbf{A}, (\mathbf{B}, \mathbf{C}))$ *is minimized.*

PROOF. Consider function $L(\mathbf{A}, H) = L(H) + L(\mathbf{E})$ that has $L(H) < L(\mathbf{E})$ for any $\mathbf{E}$ with $|\mathbf{E}| > 0$. Furthermore, let $L(H)$ be such that if $\mathbf{B}$ is n-by-k and $\mathbf{B}'$ is n-by-k' with $k < k'$, then $L(\mathbf{B}) < L(\mathbf{B}')$ (and same for $\mathbf{C}$). (Such encoding could, for example, encode every value of $\mathbf{B}$ and $\mathbf{C}$ with one bit, and then send more than $2nm$ bits for each 1 in $\mathbf{E}$.) Then clearly, if $\mathbf{B}$ and $\mathbf{C}$ are such that $\mathbf{A} = \mathbf{B} \circ \mathbf{C}$ and they are as small as possible, they also minimize $L(\mathbf{A}, H)$. But if such $\mathbf{B}$ is n-by-k, then we know that $\mathrm{rank}_B(\mathbf{A}) = k$. As finding the Boolean rank of $\mathbf{A}$ is NP-complete problem, if a polynomial-time algorithm can minimize L, it must be that $\mathrm{P} = \mathrm{NP}$. □

The encoding used in the above proof is, of course, something we would not use in practice. Also, Proposition 1 does not say that *all* encoding lengths would be hard to minimize (indeed, it is not very hard to come up with an encoding that is trivial to minimize). What Proposition 1 does say, however, is that if we can minimize some encoding length in polynomial time, that must be because of some specific properties of the encoding (or because $\mathrm{P} = \mathrm{NP}$).

With MDL in general, and our encodings specifically, we have the problem that the total encoded size is not monotonic with regard to addition and removal of factors—or, moving of 1s from $\mathbf{E}$ into the factors. Adding a factor always costs bits, but depending on how many 1s it replaces, and how much error it introduces, it may or not lead to a decrease of the total encoded size. Another problem with the more efficient XOR encodings is that introducing error at one iteration influences the cost of introducing error later on: a locally optimal decision may turn out to be quite bad. Moreover, there is no usable structure that we can exploit for a direct search for the MDL-optimal BMF.

Intuitively, however, it would make a good heuristic to iteratively minimize the MDL cost by finding the factor that will save us most bits. This means that we are essentially trying to move as many 1s possible from $\mathbf{E}$ into a new factors in $\mathbf{B}$ and $\mathbf{C}$, while keeping the complexity of those factors in mind. (In fact, the MDL optimal factorization can be the same as the zero-error factorization, the latter known to be NP-hard to compute in general.) Basically, this strategy means minimizing the error, while favoring factors with short description lengths. If we drop the latter, non-monotonic requirement, our strategy is to minimize error, which is the same strategy Asso follows to solve BMF heuristically, and is one reason why we choose to employ Asso here.

5.4 Merging MDL and Asso

The most straightforward way to decide the model order, given an encoding method, is to compute the BMF for each value of k, and to report the k at which $L(\mathbf{A}, H)$ was found to be minimal. This naïve

strategy, however, may be very slow in practice, as, if we do not set a maximum k ourselves, we would have to re-start $\min(n, m)$ times—with n and m in the order of thousands, this would mean a significant computational task. But here the hierarchical nature of Asso comes to help: instead of having to re-start every time, we can start by computing the first factor, compute its description length, add the second factor, compute the new encoded length, and so on. To compute $L(\mathbf{A}, H)$, we can use the fact that at any point we know the cost of encoding the previous $k-1$ factors, and thus only have to compute the cost for the new factor. Further, as Asso calculates per step how much it reduces the error, we can use this information when encoding the error matrix.

As we minimize the error, we cannot guarantee that the description length is a convex function with respect to k. Nevertheless, if we assume the cost to be approximately convex we can implement an early-stopping criterion: stop the algorithm if compression has not improved during the last c steps.

The benefits of using Asso with MDL are not restricted to only selecting the model order. Recall that Asso requires a user-provided parameter t to generate the candidate factors. Selecting the value for this parameter can be a daunting task. Here MDL also helps: by computing the decompositions for different values of t, we can select the pair (k, t) that minimizes the total description length. Hence, we can use MDL not only for BMF model order selection in general, but also for BMF model selection.

6. EXPERIMENTS

In this section we experimentally evaluate how well our description length functions identify the correct model orders. While we naturally also investigate the factors discovered at these orders, note that this is not specifically the topic of this paper; rather, our main concern is the model order selection.

We implemented the scoring models and the Asso algorithm in Matlab/C, and provide the source code for research purposes together with the generator for the synthetic data.[2]

6.1 The Other Option: Cross Validation

We selected 10-fold cross-validation (CV) as our benchmark method. As we are not trying to predict missing values, hold-out data are columns of the original data. While CV is an appealing and simple approach, it has one major drawback in the context of BMF: generalizing the column factors to the hold-out data is NP-hard. Namely, finding the best possible way to express the given column with the factors is called Basis Usage Problem, and is known to be NP-hard even to approximate better than with super-polylogarithmic factor [20]. To solve this problem, we used the same greedy process used by Asso (see Section 4.2). The overall error for each k and t was averaged over the folds, and the parameters that gave the least average error were selected.

Notice that we cannot use Owen and Perry's bi-cross-validation approach [28], as their method to generalize the learned factors to test data does not apply to BMF.

6.2 Synthetic Data

We start our experimental analysis in a controlled environment by using synthetic data. For these experiments, we generate n-by-m matrices, in which we plant k random itemsets, of random cardinality, and random frequency. We also add noise to the matrix by flipping values from 0 to 1, and vice-versa, with respective probabilities of n^+ and n^-.

[2] `http://www.mpi-inf.mpg.de/~pmiettin/src/mdl4bmf.zip`

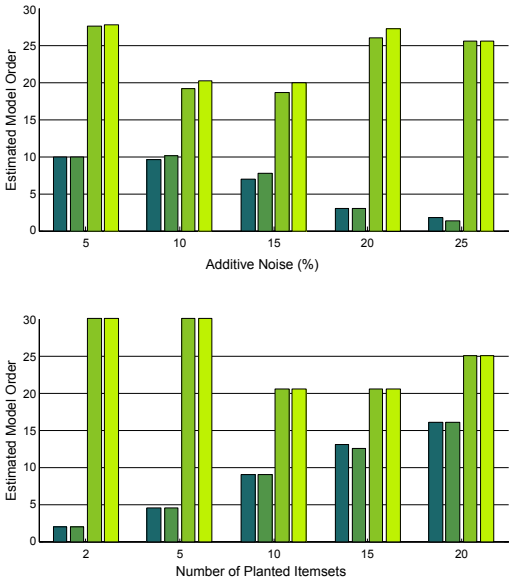

Figure 1: Model order estimates for varying amounts of additive noise (top, true $k = 10$) and varying true model order (bottom). The bars represent the estimates of, from left to right, Typed XOR, Naïve XOR, Naïve Indices, and Naïve Factor. All values averaged over 5 datasets.

First, we investigate sensitivity to noise. To this end, for n^+ from 5% to 25% in steps of 5%, we generate 5 matrices of 8000-by-100 planting 10 random itemsets, each of random cardinality between 4 and 6, and random frequency between 10% and 40%. We fix n^- to 5%. Note that at the highest noise levels, more than twice as many 1s are due to noise than to structure. For each of these datasets, we run Asso for k from 1 to 30, and t from 0.1 to 0.9 in increments of 0.025. For each of the $30 \times 32 = 960$ factorizations this gives us per dataset, we calculate $L(\mathbf{A}, H)$ for each encoding, and per encoding we record the values for k associated with the minimal encountered description length. For the same datasets, we also estimate k by running Asso with 10-fold cross-validation.

In this experiment, as well as in all other experiments, cross-validation always always estimates k to full-rank. Hence, we draw the conclusion that CV does not give good estimates in combination with Asso, and is not a good choice for solving the BMF model order selection problem, and do not further report on it (but see Section 7 for discussion why CV fails).

In the top graph of Figure 1 we plot the average k per encoding strategy for each level of noise. We see that Naïve Factor and Naïve Indices perform equal, and over-estimate k strongly. Naïve XOR and Typed XOR do provide very good estimates: exact at 5% and 10%, still very good at 15%, and underestimate when the large majority of the 1s in the data are due to noise at 20% and 25%.

While under-estimation is clearly preferred to over-estimation—we do not want to model noise—yet it does raise the question whether the effect is due to the scoring function or the search strategy; by iteratively minimizing error, Asso may simply not consider any H of correct k that remotely resemble the true model, and hence, making it impossible to detect the correct model order.

To see what is the case, we manually compare the encoded sizes of the true model to that of the best model found by Asso. The results are clear: for low noise, Asso finds models that compress as well as the true model, sometimes even better, e.g., by not to mod-

eling damaged parts of a structure. Unsurprisingly, the discovered itemsets match the underlying model very well.

For high noise levels, on the other hand, we see that Asso returns models that compress worse, typically requiring 10% more bits. The discovered itemsets, however, do consist of combinations of the true itemsets; specifically, those with relatively large overlap in items and rows. So, while the true itemsets are in fact detected, by the iterative minimization of error, Asso does not report them separately. Hence, it is clear that the scoring function performs very well; even for the highest levels of noise, the true model compresses much better, and hence, if it (or any model remotely resembling it) would be considered, the model order would be identified correctly.

Next, we experiment for varying model orders. We again generate matrices of 8000-by-100, and plant random itemsets of cardinality between 2 and 10 and frequencies between 10% and 40%, fixing n^+ to 10%, and n^- to 5%. We experiment with datasets in which we plant 2, 5, 10, 15, and 20 itemsets. The bottom graph of Figure 1 shows the averaged results for each of these datasets. We see again that the Factor and Indices encodings overestimate, and may dismiss these as good encodings for the model order selection problem. The much more efficient XOR encodings, on the other hand, do consistently give very good estimates. Only for 20 planted itemsets we see a slight under-estimation, of which inspection shows is again due to overlap (due to chance, and the small number of columns, i.e., 100) between itemsets, which makes Asso's greedy strategy group some of these together. (Also, for $k = 10$, we note that Naïve Indices is either correct *or* strongly overestimates.)

These two experiments show that Naïve XOR and Typed XOR provide highly accurate BMF model order estimates, even for large amounts of noise and densely populated matrices. Further experiments show these encodings also identify k accurately for datasets of varying density, numbers of columns, and data rows. As over all experiments, Typed XOR gives the best results, we will use this encoding for the remainder of this section.

6.3 Real Data

Now that we know that the model order is accurately identified for synthetic data, we proceed to real data. We first consider 7 datasets, most of which are publicly available. The 859-by-3933 *Abstracts* dataset represents the words for all abstracts of the accepted papers at the ICDM conference up to 2007, where the words have been stemmed and stop words removed [17]. The 6980-by-19 *DBLP* dataset contains records of in which of the 19 conferences the 6980 authors had published. The dataset is collected from the DBLP database[3] and it is pre-processed as in [20]. *Dialect* is a 1334-by-506 presence data of dialectical linguistic properties in 506 Finnish municipalities [7]. The *DNA Amplification* data, of 4590-by-392, contains information on DNA copy number amplifications. Such copies activate oncogenes and are hallmarks of nearly all advanced tumors [26]. Amplified genes represent attractive targets for therapy, diagnostics and prognostics. This dataset exhibits a banded structure [11]. The *Mammals* presence data[4] consists of presence records of 124 European mammals within 2183 geographical areas of 50×50 kilometers [24]. *Newsgroups* is a subset of the *20Newsgroups* dataset,[5] containing, for 400 posts from 4 newsgroups[6], the usage of 800 words. Finally, the 501-by-139 *Paleo* dataset consists of fossil records per location.[7]

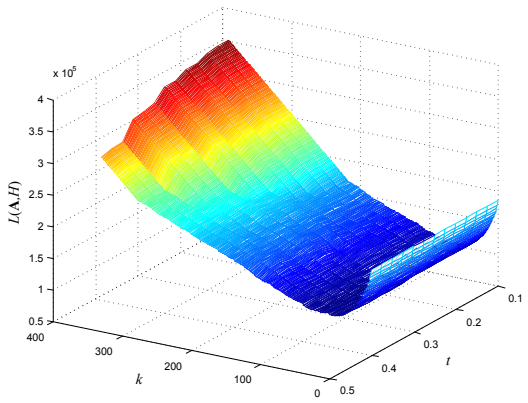

Figure 2: A 3D plot of the number of bits (y-axis), for varying values of k (x-axis) and t (z-axis) for the *DNA* dataset. Minimal description length attained at $k = 57$ and $t = 0.325$.

Before we discuss the identified model orders, we look at the sensitivity to Asso's parameter t. In Figure 2, we plot a typical example of the space of total encoded lengths, $L(\mathbf{A}, H)$. For the *DNA* data, we show all $L(\mathbf{A}, H)$ for every considered k and t. The figure shows the landscape to be a valley, with extreme high values for overly complex and overly simplistic models, and the model order is to be found in the distinct minimum around $k = 57$; which is the value for k at which $L(\mathbf{A}, H)$ for Asso is minimized. Further, the plot tells us that t does not (strongly) influence the detected model order k. Over all considered datasets the landscape is of similar shape, with the exception of the highly noisy synthetic data. Hence, in practice, we can do with a coarse sweep over t.

In Figure 3 we show, for each of the datasets, the total encoded size per k, fixing t to the value at which the minimum description length was found. The identified model orders (i.e., values for k at which the minimum was reached) are given in each figure. As the plots show, our description length function is close-to-convex for Asso's greedy heuristic search. This means that the early-stop criterion c can validly be employed. It also suggests that binary search might be a valid search strategy for non-hierarchical algorithms.

As mentioned above, the purpose of these experiments is to assess the quality of our model-order selection approach, not that of BMF or Asso per se. Nevertheless, we have to analyze whether the selected model orders make sense, and can best do this by investigating the factors Asso discovers at the identified model orders.

For the *DBLP* dataset, our method proposes $k = 4$. This yields four disjoint sets of conferences as row factors: (1) SIGMOD, VLDB, ICDE, (2) SODA, FOCS, STOC, (3) KDD, PKDD, ICDM, and (4) ICML, ECML. These four factors clearly correspond to four different field of computer science, namely (1) databases, (2) theoretical computer science, (3) data mining, and (4) machine learning. The suggested number of factors for *Newsgroups*, 17, might seem high given that the data is about four newsgroups. But it would be overly simplistic to assume that each newsgroup could be represented with just one (or even two) topic. Instead, there are some general topics (i.e., factors containing words appearing across the data, such as 'question'), and three to four subtopics per newsgroup. The estimate for *DNA*,

[3] http://www.informatik.uni-trier.de/~ley/db/

[4] Available for research purposes from the Societas Europaea Mammalogica at http://www.european-mammals.org

[5] http://people.csail.mit.edu/jrennie/20Newsgroups/

[6] The authors are grateful to Ata Kabán for pre-processing the data; the pre-processing in explained in [20]

[7] NOW public release 030717, available from http://www.helsinki.fi/science/now/ [10].

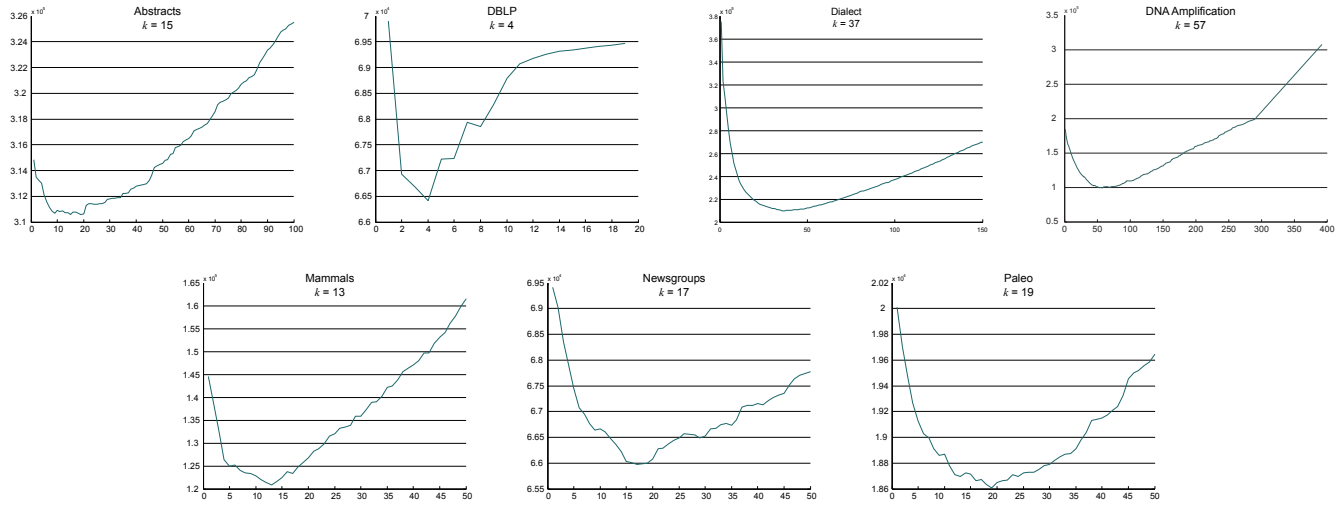

Figure 3: MDL score per k for, from left to right: (top row) *Abstracts*, *DBLP*, *Dialect*, *DNA*, (bottom row) *Mammals*, *Newsgroups*, and *Paleo*. Value for t fixed per dataset to value at which the minimal description length is found over all t and k.

$k = 57$, returns factors that model the known chromosomal regions (or, bands [11]) of interest very well [26], only missing some of the smallest itemsets. What comes to *Paleo* ($k = 19$), Tatti et al. [34] computed its normalized correlation dimension to be 15. While this normalized correlation dimension has no direct connection to BMF, we consider it interesting that these two methods designed for Boolean data give rather similar results. Even more so, as Tatti et al. [34] report that explaining 90% of the variance with PCA requires 79 factors.

We also checked the error curves for these data sets. In most of the cases, error decreases smoothly as k increases, and hence we cannot find any 'elbow' that would suggest the model order.

We also experimented with *NSFAbstracts* dataset. The data[8] contains 4894 terms used in 12 841 project abstracts. The preprocessing is explained in [20]. The resulting data is extremely sparse (0.9%).The model selection procedure resulted to $t = 0.8$ and $k = 1848$. Here, k is of course too large for humans to interpret the factors. Yet, it seems that the data indeed requires quite large k, as the model cost decreases steeply when k is increased. Furthermore, using PCA, 1848 factors explain only about 69% of the variance, and in order to explain at least 90% of the variance, PCA needs to have 3399 factors. While these numbers are not directly comparable, they do indicate that the data has very complex structure (e.g., very little overlap between the rows). This can, at least partly, explain also the need for high k with BMF.

7. DISCUSSION

The experiments show our approach to solve the BMF model order selection problem by MDL works very well. By the MDL principle, and hence by employing the most efficient encoding for the error matrix, Typed XOR, we find highly accurate estimates.

We also applied cross-validation. While intuitively appealing, it failed to produce any reasonable results. Why was that? There are (at least) two possible reasons. First, generalizing the learned factors to new columns is a hard problem, as we already mentioned. But it might be that cross-validation fails even if we could generalize in an optimal way. The problem is that when we generalize to new columns (or rows), we do not have to use all factors. The consequence is most obvious in hierarchical decompositions, such as those returned by Asso: adding a new factor reduces the error on training data, but it will never increase the error on test data (if it would, we would not use it to explain the test data). This is not special to Boolean factorizations, as similar behavior has been reported with PCA [6].

Of our error matrix encoding strategies, the inefficient Naïve Factor and Naïve Indices do not work well in practice—except when noise is low and a model closely resembling the true model is offered. Naïve XOR and Typed XOR, on the other hand, consistently provide highly accurate model order estimates. Neither overestimate model complexity, and naturally provide underestimation in high-noise situations; highly desirable properties for model (order) selection. As it is the most efficient encoding, our overall recommendation is to use Typed XOR to encode $\mathbf{E}$.

Experiments on real data show that meaningful factors are discovered at the selected model orders, as well as that the score is near-convex. This means that, even for the heuristic BMF algorithm we employed in our experiments, we can confidently employ a greedy early-stopping criterion.

We did not investigate data-to-model codes in this paper. Such codes are very efficient, and hence we expect these to work badly with Naïve Factors and Naïve Indices, while not providing significant benefits compared to the current encoding when used with Typed XOR. Space constraints, however, prevent us from exploring these hypotheses here.

Finally, we note that our encoding length functions are easily adaptable for different variations of BMF, such as Boolean column subset-selection [20] and dominated BMF [21].

8. CONCLUSION

We proposed a general solution to the model order selection problem for Boolean matrix factorization. By the Minimum Description Length principle, we formulate the number of bits required to lossless encode the model and the error it introduces, and report the order of the model for which this sum is minimized. We empirically evaluated its performance in combination with the Asso BMF algorithm. The experiments show that the correct model orders are reliably selected for varying amounts of noise and model orders, and

[8] `http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html`

moreover, that the models at which the MDL score is minimized consist of meaningful factors.

Future work includes the development of good heuristics that optimize the MDL score directly, instead of the error, when finding Boolean matrix factorizations of a given rank, as well as further analysis of the complexity of problem.

Acknowledgements

Jilles Vreeken is supported by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

9. REFERENCES

[1] R. Belohlavek and V. Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comp. Sys. Sci.*, 76(1):3–20, 2010.

[2] R. Cattell. The scree test for the number of factors. *Multivar. Behav. Res.*, 1:245–276, 1966.

[3] V. Chandola and V. Kumar. Summarization – compressing data into an informative representation. *Knowl. Inf. Syst.*, 12(3):355–378, 2007.

[4] R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Trans. Inform. Theory*, 51(4):1523–1545, 2005.

[5] T. Cover and J. Thomas. *Elements of Information Theory, 2nd ed.* John Wiley and Sons, 2006.

[6] C. T. dos S. Dias and W. J. Krzanowski. Model selection and cross validation in additive main effect and multiplicative interaction models. *Crop Sci.*, 43:865–873, 2003.

[7] S. M. Embleton and E. S. Wheeler. Finnish dialect atlas for quantitative studies. *J. Quant. Ling.*, 4(1–3):99–102, 1997.

[8] C. Faloutsos and V. Megalooikonomou. On data mining, compression and Kolmogorov complexity. *Data Min. Knowl. Disc.*, 15:3–20, 2007.

[9] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. UAI'93*, pages 1022–1027, 1993.

[10] M. Fortelius et al. Neogene of the old world database of fossil mammals (NOW), 2003. http://www.helsinki.fi/science/now/.

[11] G. C. Garriga, E. Junttila, and H. Mannila. Banded structure in binary matrices. In *Proc. KDD'08*, pages 292–300, 2008.

[12] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Proc. DS'04*, pages 278–289, 2004.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU Press, 1996.

[14] P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[15] H. Heikinheimo, J. Vreeken, A. Siebes, and H. Mannila. Low-entropy set selection. In *Proc. SDM'09*, 2009.

[16] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proc. KDD'04*, pages 206–215, 2004.

[17] K.-N. Kontonasios and T. De Bie. An information-theoretic approach to finding noisy tiles in binary databases. In *Proc. SDM'10*, 2010.

[18] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1993.

[19] H. Lu, J. Vaidya, and V. Atluri. Optimal Boolean matrix decomposition: Application to role engineering. In *Proc. ICDE'08*, pages 297–306, 2008.

[20] P. Miettinen. *Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms*. PhD thesis, University of Helsinki, 2009.

[21] P. Miettinen. Sparse Boolean matrix factorizations. In *Proc. ICDM'10*, pages 935–940, 2010.

[22] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008.

[23] T. P. Minka. Automatic choice of dimensionality for PCA. In *Proc. NIPS'00*, pages 598–604, 2001.

[24] A. J. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, P. J. H. Reijnders, F. Spitzenberger, M. Stubbe, J. B. M. Thissen, V. Vohralik, and J. Zima. *The Atlas of European Mammals*. Academic Press, 1999.

[25] S. D. Monson, N. J. Pullman, and R. Rees. A survey of clique and biclique coverings and factorizations of $(0, 1)$-matrices. *Bulletin of the ICA*, 14:17–86, 1995.

[26] S. Myllykangas, J. Himberg, T. Böhling, B. Nagy, J. Hollmén, and S. Knuutila. DNA copy number amplification profiling of human neoplasms. *Oncogene*, 25(55), 2006.

[27] D. S. Nau, G. Markowsky, M. A. Woodbury, and D. B. Amos. A mathematical analysis of human leukocyte antigen serology. *Math. Biosci.*, 40:243–270, 1978.

[28] A. B. Owen and P. O. Perry. Bi-cross-validation of the SVD and the nonnegative matrix factorization. *Ann. Appl. Stat.*, 3(2):564–594, Jun 2009.

[29] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[30] V. Pestov. An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21(2-3):204–213, 2008.

[31] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(1):465–471, 1978.

[32] M. Schmidt, O. Winther, and L. Hansen. Bayesian non-negative matrix factorization. In *Proc. ICA'09*, volume 5411 of *LNCS*, pages 540–547, 2009.

[33] A. Streich, M. Frank, D. Basin, and J. Buhmann. Multi-assignment clustering for Boolean data. In *Proc. ICML'09*, pages 969–976, 2009.

[34] N. Tatti, T. Mielikainen, A. Gionis, and H. Mannila. What is the dimension of your binary data? In *Proc. ICDM'06*, pages 603–612, 2006.

[35] N. Tatti and J. Vreeken. Finding good itemsets by packing data. In *Proc. ICDM'08*, pages 588–597, 2008.

[36] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: Finding a minimal descriptive set of roles. In *Proc. SACMAT*, pages 175–184, 2007.

[37] J. Vreeken and A. Siebes. Filling in the blanks – KRIMP minimisation for missing data. In *Proc. ICDM'08*, pages 1067–1072, 2008.

[38] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: Mining itemsets that compress. *Data Min. Knowl. Discov.*, 19(2):176–193, 2010. 10.1007/s10618-010-0202-x.

[39] C. Wallace. *Statistical and inductive inference by minimum message length*. Springer-Verlag, 2005.

[40] J. Wang and G. Karypis. On efficiently summarizing categorical databases. *Knowl. Inf. Syst.*, 9(1):19–37, 2006.

[41] K. Yeomans and P. Golder. The Guttman–Kaiser criterion as a predictor of the number of common factors. *The Statistician*, 31(3):221–229, 1982.

[42] M. Zhu and A. Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Comp. Stat. & Data Anal.*, 51(2):918–930, 2006.