BFA and BMF: What is the Difference

Alexander A. Frolov
Institute of Higher Nervous Activity
Russian Academy of Sciences, Moscow, Russia
Email: aafrolov@mail.ru

Dušan Húsek
Institute of Computer Science
Acad. of Sci. of the Czech Rep., Prague, Czech Republic
Email: dusan@cs.cas.cz

Ajith Abraham, Pavel Y. Polyakov
IT4Inovations
VSB Technical University Ostrava, Czech Republic
Email: abraham.ajith@gmail.com, pavel.mipt@mail.ru

Hana Řezanková
Department of Statistics and Probability
University of Economics, Prague, Czech Republic
Email: rezanka@vse.cz

***Abstract*—Studied are differences of two approaches to binary data dimension reduction. The first one is Boolean Matrix Factorization and the second one is Expectation Maximization Boolean Factor Analysis. The two BMF methods are used for comparison. First is M8 method from the BMDP statistical software package. The second is the BMF method, as suggested by Belohlavek and Vychodil (BVA2). These two are compared to Expectation Maximization Boolean Factor Analysis extended with binarization step developed here. Generated (Bars problem) and mushroom dataset are used for(experiments. In particular, under scrutiny was the reconstruction ability of the computed factors and the information gain as the measure of dimension reduction. In addition, presented are some general remarks on all the methods being compared.**

***Index Terms*—Dimension reduction, statistics, data mining, Boolean factor analysis, Boolean matrix factorization, information gain, likelihood-maximization, bars problem.**

I. Introduction

A fundamental problem in many data-analysis tasks is to find a suitable representation of the data. A useful representation typically makes latent structure in the data explicit, and often reduces the dimensionality of the data so that further computational methods can be applied. Dimensionality reduction methods are able to transform a high-dimensional space of attributes to a lower-dimensional space. There exists high demand for such transformation in many areas of human activity (such as engineering, computer science, biology or economics) where one is facing a problem of efficient processing of large datasets. So dimension reduction methods are crucial part of our life.

Some linear methods were promising and still are very useful for dimension reduction; let us mention Singular Value Decomposition [2] or latent Dirichlet allocation (LDA) [3].

The main problem of unconstrained matrix factorization methods lies in the fact that resulting representations are hard to interpret. To solve this problem Non-negative matrix factorization (NMF) method (see [4], [5]) was developed for finding more natural representation. Given a non-negative data matrix $\mathbf{X}$, NMF finds an approximate factorization $\mathbf{X} = \mathbf{WH}$ into non-negative matrices $\mathbf{W}$ and $\mathbf{H}$. The non-negativity constraints make the representation purely additive (allowing no subtractions), in contrast to many other linear representations such as principal component analysis (PCA) and independent component analysis (ICA) [6]. These (and other) matrix decomposition methods allow the matrices $\mathbf{W}$ and $\mathbf{H}$ to contain arbitrary real numbers.

However, if the input matrix $\mathbf{X}$ is binary, it is natural to require that $\mathbf{W}$ and $\mathbf{H}$ are also binary. In this paper, we consider the matrix decomposition problem created by this requirement. In this case, the combination operation of matrices $\mathbf{W}$ and $\mathbf{H}$ is the Boolean matrix product (i.e., the matrix product in the semiring of Boolean $\wedge$ and $\vee$).

However as we stated, what makes the dimension reduction process successful is the finding of latent structure of the data. What we want to demonstrate here is that better results can be achieved when using a model based statistical procedure for Boolean factor analysis.

The paper is organized as follows. In Section II, we firstly describe the problem of Boolean matrix factorization (BMF) and Boolean factor analysis (BFA), and secondly we briefly describe the three solving methods, Expectation-Maximization Boolean Factor Analysis (EMBFA) which is a BFA method, and then two BMF methods, i.e. methods which we compare in this paper. Section III contains the description of two measures used for estimating the methods performance. Then Section IV contains description of databases used. Finally the experimental comparison of the methods is presented in Section V. Section VI contains discussion on further issues and concludes the paper.

II. BMF and BFA methods

Boolean Matrix Factorization [1] implies presentation of binary matrix of observed dataset $\mathbf{X}$ in the form

$$\mathbf{X} = \mathbf{S} \otimes \mathbf{F}, \tag{1}$$

where each row of binary $M \times N$ matrix $\mathbf{X}$ is an observed pattern, each row of binary $L \times N$ matrix $\mathbf{F}$ is a representation of a factor in the signal space and each row of binary $M \times L$ matrix $\mathbf{S}$ is a set of factor scores defining which factors are mixed in the patterns. Boolean matrix product $\otimes$ means that

978-1-4673-5119-5/12/$31.00

each component of matrix $\mathbf{X}$ is obtained as $x_{mj} = \bigvee_{i=1}^{L} S_{mi} f_{ij}$. The method implies identification of a minimal set of factors that provide representation of the observed data in the form (1). The number of such factors is called the Boolean rank of $\mathbf{X}$. Since the combinatorial problem of $\mathbf{X}$ rank identification is NP complete [7] existing methods give reasonable, but not obligatory optimal solutions. Optimal solution gives a brute force search which is not suitable for high dimensional data. On other side the classical linear methods could not take into account non-linearity of Boolean summation and therefore are inadequate for this task.

A. Formal Concepts Boolean Matrix Factorization

Recently authors Belohlavek and Vychodil in [1] revealed a tight relationship between BMF and formal concept analysis [8] and developed two simple greedy algorithms solving this task.

1) Algorithm BVA1: This algorithm, named as "Algorithm 1" in [1], utilizes formal concepts of $\mathbf{X}$ as factors. Recall that a formal concept of $\mathbf{X}$ [8] is any pair $\langle C; D\rangle$ of sets $C \subseteq 1, \ldots, m$ (rows, objects) and $D \subseteq 1, \ldots, n$ (columns, attributes) satisfying the following property: D is the set of all attributes j for which $x_{ij} = 1$ for every object $i \in C$ and, vice versa, C is the set of all objects i for which $x_{ij} = 1$ for every attribute $j \in D$. Formal concepts are very well understood by domain experts. Geometrically, they are, up to permuting rows and columns, just maximal rectangles full of 1s in the matrix X. If a set $\mathfrak{I}$ of formal concepts is to be used as a set of factors of $\mathbf{X}$, the corresponding matrices $\mathbf{S}_{\mathfrak{I}}$ and $\mathbf{F}_{\mathfrak{I}}$ are defined the following way: column l of $\mathbf{S}_{\mathfrak{I}}$ is just the characteristic vector of C_l and row l of $\mathbf{F}_{\mathfrak{I}}$ is just the characteristic vector of D_l, where $\langle C_l; D_l\rangle$ is the lth formal concept in $\mathfrak{I}$. It is proved in [8] that using such factors is optimal in that the Boolean rank of $\mathbf{X}$ may be achieved by using formal concepts as factors. In algorithm BVA1, one first computes all the formal concepts of $\mathbf{X}$. The algorithm proceeds in a greedy way: In every step, it selects the concept that covers the largest number of entries with 1 in $\mathbf{X}$ that were not covered by the previously selected concepts ($\langle C; D\rangle$ covers $X_{ij} = 1\rangle$ if $i \in C$ and $j \in D$, i.e. the rectangle corresponding to $\langle C; D\rangle$ spans over the entry $\langle i; j\rangle$).

2) Algorithm BVA2: This algorithm, named as "Algorithm 2" in [1], utilizes formal concepts of $\mathbf{X}$ as factors the same way as algorithm BVA1. However, algorithm BVA2 avoids the necessity to compute all the concepts of $\mathbf{X}$ and browse through them during the greedy selection. Instead, the algorithm computes the candidate factors, i.e. concepts of $\mathbf{X}$, on demand the following, greedy way. Each time a new factor is needed, one looks at the columns of $\mathbf{X}$ and selects the one concept generated by a column which covers most of the yet uncovered 1s in $\mathbf{X}$. Such a concept corresponds to a narrow but high rectangle in the data. Then one tries to see if such rectangle may be extended to a wider (and thus not so high) rectangle by adding some attribute and deleting the objects so that one still has a rectangle. If so, one selects the best such rectangle, i.e. covering most of the yet uncovered 1s in $\mathbf{X}$. One repeats the process of extension until no such extension yields a better rectangle. This way one obtains the new factor and eventually a set $\mathbf{F}$ of formal concepts—the factors of $\mathbf{X}$.

For our computer comparison we used the second faster algorithm BVA2 only.

B. Boolean Matrix Factorization; the M8 Procedure in BMDP

The second approach of Boolean matrix factorization is implemented in statistical package BMDP [9], which was originally developed at the UCLA for biomedical applications. The method even if it is solving BMF task is called Boolean factor analysis and the procedure is denoted as M8. The M8 procedure is effective enough despite being based on brute force search approach. The algorithm behind M8 works as follows. To compute L factors of $\mathbf{X}$ (and thus the corresponding $M \times L$ and $L \times N$ matrices $\mathbf{S}$ and $\mathbf{F}$), the algorithm starts $k < L$ candidate rows of $\mathbf{F}$ (candidate factor loadings). These are either supplied by the user or computed from $\mathbf{X}$ using a heuristic based on inclusion of the columns of $\mathbf{X}$. From this set of k vectors of factor loadings, the algorithm computes k vectors of factor scores (candidate columns of $\mathbf{S}$); from the k vectors of scores, the algorithm tries to find better k vectors of factor loadings, etc. until no change occurs or three such cycles are completed. Such tuning of factor loadings and scores is called refinement (the details are too technical to be included here). The algorithm then iteratively adds further factors as follows. Suppose l factors have been obtained. Then, one adds new factor $l + 1$, refines the loadings and scores of all the factors as above, adds new factor $l + 2$ and refines again. Then the l^th factor is removed and the remaining factors are refined. Consequently, the process is repeated, i.e. two new factors are added, one is removed, etc. For example, starting with $k = 2$ factors, we obtain 2, 3, 4, 3, 4, 5, 4, 5, 6, 7, 6, 7, 8, etc. factors. The process stops when the required number L of factors is obtained the second time. For example, with $k = 2$ and $L = 6$, one computes 2, 3, 4, 3, 4, 5, 4, 5, 6, 7, 6 factors and the last six ones are the final factors output by the algorithm. By default, $k = L - 2$ but k may be set by a user. A new factor is added based on the matrix describing the error committed by the factors obtained so far. In particular, one uses the column of $\mathbf{X}$ which contains the largest number of 1s uncovered by the previously computed factors.

C. Boolean Factor Analysis based on the Expectation-Maximization method

The Expectation-Maximization method for Boolean Factor Analysis was developed for analysis of data of statistical origin [10]. Similar to BMF, in terms of BFA, each observation is a binary row vector $\mathbf{x} = [x_1, \ldots, x_N]$, each common factor $\mathbf{f}_i = [f_{i1}, \ldots, f_{iN}]$ is a binary row vector of dimension N. Unlike BMF, BFA is a statistical method that assumes the existence of a probabilistic generative model. The parameters of the generative model are $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \ldots, L, j = 1, \ldots, N)$. Parameter p_{ij} is the probability of the presence of jth attribute in an observation due to the ith factor. For attributes constituting the factor, i.e., for attributes with $f_{ij} = 1$,

the probability p_{ij} is high, and for the other attributes (with $f_{ij} = 0$), it is zero. Thus the contribution of ith common factor is defined by binary row vector $\mathbf{f}'_i = [f'_{i1}, \ldots, f'_{iN}]$ which is a distorted version of the vector of factor loadings $\mathbf{f}_i$. Factor distortion implies that entries of $\mathbf{f}_i$ having value equal to One can change their values to Zero with probability $1 - p_{ij}$ but none of the entries of $\mathbf{f}_i$ equal to Zero can change value to One.

Parameter q_j is the probability of the presence of the jth attribute in an observation due to specific factor η_j. It is assumed that each specific factor influences only one attribute, in contrast to common factor $\mathbf{f}_i$, which influences more than one attribute. The contribution of all specific factors is defined by a binary row vector $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_N]$.

As a result, any observation $\mathbf{x}$ can be presented in the form

$$x_j = [\bigvee_{i=1}^{L} s_i \wedge f'_{ij}] \vee \eta_j,$$

where $\mathbf{s} = [s_1, \ldots, s_L]$ is a binary row vector of factor scores of dimension L, L being the total number of factors.

Parameter π_i $(i = 1, \ldots, L)$ is the probability that the ith factor appears in an observation.

We assume that factors are distorted independently of other factors and specific factors, factor's components are distorted independently of other components, and specific factors are independent of each other and of the common factors.

In contrast to BMF, which is aimed to find exact or approximate decomposition of a given dataset, the aim of BFA is to find the parameters of a generative model $\boldsymbol{\Theta}$ and factor scores for all patterns of the dataset. Moreover, it is supposed that the factors found could also be detected in any arbitrary pattern $\mathbf{x}$, if generated by the same BFA model. Note that in the case $p_{ij} = f_{ij}$ and $q_j = 0$ BFA provides the exact decomposition of a given dataset equivalent to BMF solution given by (1).

The EMBFA maximizes the likelihood of the observed data by maximizing the free energy

$$\mathcal{F} = \sum_{m=1}^{M} \left\{ \sum_{\mathbf{s}} g_m(\mathbf{s}) \log(P(\mathbf{x}_m|\mathbf{s}, \boldsymbol{\Theta}) P(\mathbf{s}|\boldsymbol{\Theta})) + H(g_m(\mathbf{s})) \right\},$$

where $g_m(\mathbf{s})$ is the expected distribution of factor scores for the mth pattern, $H(g_m(\mathbf{s}))$ is the Shannon entropy of $g_m(\mathbf{s})$,

$$P(\mathbf{x}|\mathbf{s}, \boldsymbol{\Theta}) = \prod_{j=1}^{N} P(x_j|\mathbf{s}, \boldsymbol{\Theta}),$$

where

$$P(x_j|\mathbf{s}, \boldsymbol{\Theta}) = x_j - (2x_j - 1)(1 - q_j) \prod_{i=1}^{L} (1 - p_{ij})^{s_i}, \qquad (2)$$

$$P(\mathbf{s}|\boldsymbol{\Theta}) = \prod_{i=1,L} \pi_i^{s_i} (1 - \pi_i)^{1 - s_i}.$$

The iterations of EM alternatively increase $\mathcal{F}$ with respect to the distributions g_m, while holding $\boldsymbol{\Theta}$ fixed (the E-step), or with respect to parameters of the model $\boldsymbol{\Theta}$, while holding g_m fixed (the M-step).

At the E-step, the distributions g_m maximizing $\mathcal{F}$ are calculated according to the following equation

$$g_m(\mathbf{s}|\boldsymbol{\Theta}) = \frac{P(\mathbf{x}_m|\mathbf{s}, \boldsymbol{\Theta}) P(\mathbf{s}|\boldsymbol{\Theta})}{\sum_{\mathbf{s}} P(\mathbf{x}_m|\mathbf{s}, \boldsymbol{\Theta}) P(\mathbf{s}|\boldsymbol{\Theta})}.$$

The obtained distributions g_m provide the expected likelihood of the observed data over the factor scores for the given set of parameters of the generative model.

At the M-step, π_i can be obtained as

$$\pi_i = (1/M) \sum_{m=1}^{M} s_{mi},$$

where

$$s_{mi} = \sum_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta}) s_i.$$

Respectively, p_{ij} and q_j can be obtained by steepest ascent maximization of $\mathcal{F}$:

$$\Delta p_{ij} = \gamma_{ij} \frac{\partial \mathcal{F}}{\partial p_{ij}}, \qquad \Delta q_j = \gamma_j \frac{\partial \mathcal{F}}{\partial q_j}, \qquad (3)$$

where γ_{ij} and γ_j are learning rates,

$$\frac{\partial \mathcal{F}}{\partial p_{ij}} = \sum_{m=1}^{M} \sum_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta}) P(x_{mj}|\mathbf{s}, \boldsymbol{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s}, \boldsymbol{\Theta})}{\partial p_{ij}}$$

$$\frac{\partial \mathcal{F}}{\partial q_j} = \sum_{m=1}^{M} \sum_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta}) P(x_{mj}|\mathbf{s}, \boldsymbol{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s}, \boldsymbol{\Theta})}{\partial q_j}$$

and according to (2)

$$\frac{\partial P(x_{mj}|\mathbf{s}_m, \boldsymbol{\Theta})}{\partial p_{ij}} = (x_{mj} - P(x_{mj}|\mathbf{s}_m, \boldsymbol{\Theta})) \frac{s_{mi}}{1 - p_{ij}}$$

$$\frac{\partial P(x_{mj}|\mathbf{s}_m, \boldsymbol{\Theta})}{\partial q_j} = (x_{mj} - P(x_{mj}|\mathbf{s}_m, \boldsymbol{\Theta})) \frac{1}{1 - q_j}.$$

At each iteration cycle of step M, we put $p_{ij} = 0$ if

$$p_{ij} < 1 - \prod_{l \neq i} (1 - \pi_l p_{lj}),$$

where the right side of the inequality is the probability that the jth attribute appears in the pattern due to other factors besides $\mathbf{f}_i$.

In our computer experiments, we set the learning rates in (3) to be

$$\gamma_{ij} = p_{ij}(1 - p_{ij})/(M\pi_i), \qquad \gamma_j = q_j(1 - q_j)/M.$$

The iterative procedure (3) at each step M continues until $\sum_{ij} |\Delta p_{ij}|/LN$ become smaller than $\epsilon_2 = 10^{-3}$.

The obtained values of p_{ij}, q_j and π_i are used as the input for the next E–step. EM iterative procedure terminates once values $\sum_{ij} |\Delta p_{ij}|/LN$ remained smaller than $\epsilon_1 = 10^{-3}$ where Δp_{ij} is the change of the model parameters p_{ij} comparing to the previous E–step.

After the convergence of the procedure, the resulting values s_{mi} are the estimates of the factor scores which are not binary but gradual. To satisfy the generative model, we binarized those values. The binarization threshold was chosen to maximize the BFA information gain [11], [12] (see below).

We restricted the EMBFA algorithm to the case of sparse scores, when only a small number of factors (no more than three) are supposed to be mixed in the observed patterns. In this case, summation over $\mathbf{s}$ in the above formulas is reduced to

$$\sum_{\mathbf{s}}(\dots) = (\dots)_{\mathbf{s}=0} + \sum_{i}(\dots)_{\mathbf{s}=\mathbf{s}_i} + \sum_{i<j}(\dots)_{\mathbf{s}=\mathbf{s}_{ij}} + \sum_{i<j<k}(\dots)_{\mathbf{s}=\mathbf{s}_{ijk}}, \quad (4)$$

where $\mathbf{s}_i$ is the vector of factor scores with all zeros except s_i, $\mathbf{s}_{ij}$ is the vector of factor scores with all zeros except s_i and s_j, and $\mathbf{s}_{ijk}$ is the vector of factor scores with all zeros except s_i, s_j and s_k. An increase of the number of terms in (4) leads to a considerable rise in computational complexity.

To start the EM procedure we set $\pi_i = 1/L$, where L is expected number of factors that have to be set in advance; we also initialized $q_j = 0$ and p_{ij} with random values uniformly distributed in the range from 0.2 to 0.7.

In terms of BMF, matrix of factor scores $\mathbf{S}$ whose rows are vectors of factor scores $\mathbf{s}_i$ $(i = 1, \dots, M)$ is the object-factor matrix $\mathbf{S}$. To estimate factor-attribute matrix $\mathbf{F}$ we binarized probabilities p_{ij} assuming that $f_{ij} = 1$ if $p_{ij} \geq p_{th}$ and $f_{ij} = 0$ if $p_{ij} < p_{th}$ where p_{th} is the binarization threshold. The binarization threshold was chosen to maximize the coverage quality (see below).

III. Estimation of the methods performance

To compare the efficiency of the three methods we used two measures for estimating their performance: information gain and coverage quality. The first measure is based on statistics of the input data and relates to BFA while the second one relates to BMF.

A. Information gain

Information gain is a general information theoretic measure of BFA efficiency, which is a difference of two entropies. The first is the entropy of a dataset when its hidden factor structure is unknown and the second is the entropy when it is revealed and taken into account [11].

If factor structure of the signal space is unknown, then representing the jth component of vector $\mathbf{X}$ requires $h(p_j)$ bits of information, where $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is Shannon function and p_j is probability of the jth component to take One. Representing the whole dataset requires

$$H_0 = M\sum_{j=1}^{N} h(p_j)$$

bits of information. If the hidden factor structure of the signal space is detected that is all generative model parameters and all factor scores in the dataset are found, then representing the whole dataset requires

$$H = H_1 + H_2$$

bits of information. Here

$$H_1 = M\sum_{i=1}^{L} h(\pi_i)$$

defines information required to represent factor scores and

$$H_2 = \sum_{m=1}^{M}\sum_{j=1}^{N} h(P(x_{mj}|\mathbf{\Theta}, \mathbf{S}_m)) \quad (5)$$

defines information required to represent all patterns of the dataset when factor scores are given. In (5) $P(x_{mj}|\mathbf{\Theta}, \mathbf{S}_m)$ is the probability of the jth component of mth signal $\mathbf{x}_m$ to take the value x_{mj}. This probability is given by (2).

We defined the relative information gain as

$$G = (H_0 - H)/H_0.$$

As shown in [11], information gain decreases when both the noise in dataset increases and the errors in BFA solution increases. Thus it is a reliable measure of BFA quality and propriety of BFA to given dataset at all.

B. Coverage quality

According to (1), the product $\mathbf{S} \otimes \mathbf{F}$ should approximately cover the input matrix $\mathbf{X}$. The error between $\mathbf{X}$ and its coverage by $\mathbf{S} \otimes \mathbf{F}$ is a natural measure of BMF quality. The error $E(\mathbf{X}, \mathbf{S} \otimes \mathbf{F})$ may be seen as being the sum of two components, E_u corresponding to 1s in $\mathbf{X}$ that are 0s in $\mathbf{S} \otimes \mathbf{F}$ (uncovered) and E_o corresponding to 0s in $\mathbf{X}$ that are 1s in $\mathbf{S} \otimes \mathbf{F}$ (overcovered):

$$E(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) = E_u(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) + E_o(\mathbf{X}, \mathbf{S} \otimes \mathbf{F})$$

with

$$E_u(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) = |\{\langle i, j\rangle | x_{ij} = 1, (\mathbf{S} \otimes \mathbf{F})_{ij} = 0\}|$$

$$E_u(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) = |\{\langle i, j\rangle | x_{ij} = 0, (\mathbf{S} \otimes \mathbf{F})_{ij} = 1\}|$$

Note that in the BMDP manual on the M8 method [9], E_u and E_o are called the positive and negative discrepancy, respectively. For convenience, we use the functions measuring a kind of a relative error of $\mathbf{S} \otimes \mathbf{F}$ with respect to $\mathbf{X}$, and

$$Q(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) = 1 - \frac{E(\mathbf{X}, \mathbf{S} \otimes \mathbf{F})}{||\mathbf{X}||}$$

which may be thought of as measuring coverage quality. Clearly, $Q(\mathbf{X}, \mathbf{S} \otimes \mathbf{F}) = 1$ if and only if $\mathbf{X} = \mathbf{S} \otimes \mathbf{F}$ (exact decomposition). Furthermore, $Q(\mathbf{X}, \mathbf{S} \otimes \mathbf{F})$ decreases with increasing error, i.e. with increasing $E(\mathbf{X}, \mathbf{S} \otimes \mathbf{F})$.

IV. Datasets used

We compare the efficiency of methods using the artificial signals which are random mixtures of horizontal and vertical bars and with the Mushroom dataset taken from the UCI Machine Learning Repository [13]

Fig. 1. **A** Sixteen vertical and horizontal bars in 8-by-8 pixel images. **B** Examples of images in the standard bars problem. Each image contains two bars on average.

A. Bars Problem

Bars Problem (BP) was introduced in [14] and in various modifications has been considered in many papers (see [15] for references) as a benchmark for learning of objects from complex patterns. In this problem, each pattern of the dataset is n-by-n binary pixel image containing several of $L = 2n$ possible (one-pixel wide) bars (Fig. 1). Pixels belonging and not belonging to the bar take values 1 and 0, respectively. For each image each bar could be chosen with a probability C/L, where C is the mean number of bars mixed in an image. In the point of intersection of vertical and horizontal bars, pixel takes the value 1. The Boolean summation of pixels belonging to different bars simulates the occlusion of objects. The task is to recognize all bars as individual objects on the basis of a dataset containing M images consisting of bar mixtures. In most papers where the BP was used as benchmark C was set to 2 and $n = 8$.

In terms of BFA, bars are factors. Factor loadings $f_{ij}, j = 1, \ldots, N$ take value One for pixels constituting ith bar and value Zero for pixels not constituting it, $N = n^2$ is a total number of pixels in an image. Each image is Boolean superposition of factors. Factor scores take values One or Zero dependently on bar presence or absence in the image. Thus, the bars problem is a special case of BFA. We consider the case of homogeneously distributed noise in images both in the form of factor distortion and in the form of specific factors. Particularly, we put $p_{ij} = pf_{ij}$ and $q_j = q$. This means that pixels constituting bars can take Zero with equal probabilities p and pixels can take One with equal probabilities q due to specific factors.

B. Mushroom dataset

The Mushroom dataset consists of 8125 objects and 23 nominal attributes (for example, attribute "class" with values "edible" and "poisonous", or attribute "cap-shape" taking values such as "bell", "conical" or "convex"). We transformed this dataset to a Boolean matrix by nominal scaling, i.e. by replacing a nominal attribute y with k values $v_1, \ldots, v_k$ by k Boolean attributes $y_{v_1}, \ldots, y_{v_k}$ in such a way that at ith row, the value of the column corresponding to y_{v_j} is 1 if and only if the value of the attribute y at ith row in the original dataset is equal to v_j.

V. Experiments

In this section, we compare the efficiency of the three methods for Boolean Factor Analysis and Matrix Factorization. These methods are compared according to two criteria: information gain G and coverage quality Q. Initially, the methods are compared in solving BP.

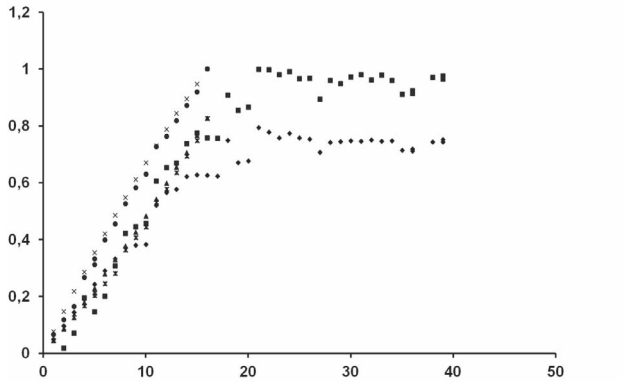

Fig. 2. Dependency of gain G and coverage Q on the number of found factors for the case when noise is absent ($p = 1$ and $q = 0$). ♦ – G(EMBFA), ■ – Q(EMBFA), ▲ – G(BVA2), × – Q(BVA2), ∗ – G(M8), ● – Q(M8).

The dependency of G and Q on the number of found factors for the case when noise is absent ($p = 1$ and $q = 0$) is depicted in Fig. 2. For all methods both criteria initially grow proportionally to the number of found factors. Since factors are assumed to be uniformly distributed in the dataset with equal probabilities $\pi_i = 1/8$, finding of each new factor provides the fixed increment of both indices G and Q. When all 16 bars are found, BVA2 and M8 stop because complete coverage is achieved. In this case, G and Q reach their maxima which correspond to the exact solution of BP. In EMBFA like in M8 the total number of searched factors L must be assigned in advance. As shown in Fig. 2, EMBFA provides maximal G and Q only when $L = 21$, that is when assigned number of factors exceeds the actual number of factors, which is the number of bars. In this case, besides the bars some false factors (not bars) are found. As a result, EMBFA slightly loses to BVA2 and M8 in information gain. Another and more important cause of its loss is the omission of some factors in the observations of the dataset. Recall that this EMBFA implementation is supposing that not more than three factors are mixed in a pattern. However, in the generative model under consideration, the number of mixed factors K has the binomial distribution $B(K, C/L, L)$, where $C = 2$ and $L = 16$. According to this distribution, 13% of patterns containing more than three factors are generated. Since only three factors can be identified in these pattern, 9% of the onevalued scores are expected to be identified as zerovalued scores. The portion of onevalued scores missed by EMBFA amounts to 10% that is close to the theoretical estimation obtained.

Fig. 3 illustrates the quality of methods for the case when common factors were undistorted ($p = 1$) but specific factors were added ($q = 0.3$). When the number of found factors L is less than the number of bars, for all methods coverage quality again increases almost proportionally to the found factors. For EMBFA and BVA2 this is accompanied by the increase of G but for M8 G remains to be close to zero. This is explained

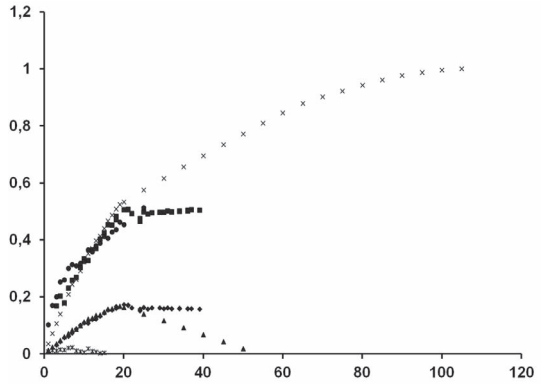

Fig. 3. Dependency of gain G and coverage Q on the number of found factors for the case when common factors were not distorted (p = 1) and specific factors were present (q = 0.3). ♦ – G(EMBFA), ■ – Q(EMBFA), ▲ – G(BVA2), × – Q(BVA2), ∗ – G(M8), ● – Q(M8).

by the fact that EMBFA and BVA2 found mostly true bars but M8 mostly false factors (not bars). For EMBFA the gain reaches maximum at the point when the assigned number of searched factors L slightly exceeds the number of true factors. As mentioned above, this occurs because all true factors can be revealed by this method only when L exceeds the actual number of true factors. For BVA2 the gain reaches maximum when the number of (correctly) found factors L equals to the number of bars. When L increases the gain decreases due to the influence of found false factors. Recall that in BVA2 and M8 there is no notion of specific factors, that is why specific factors appeared in observations when q = 0.3 are treated as common ones. For EMBFA when the number of assigned searched factors L continues to increase the gain does not decrease because specific factors are treated correctly.

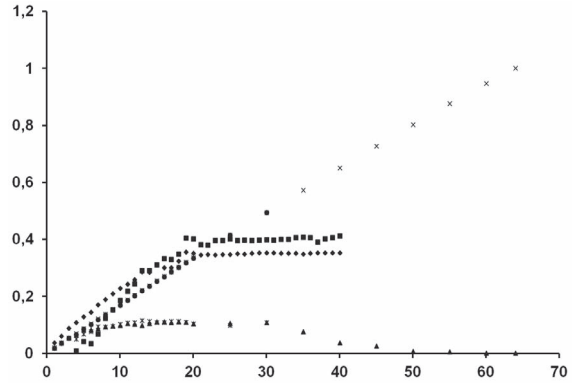

Fig. 4. Dependency of gain G and coverage Q on the number of found factors for the case when common factors were distorted (p = 0.6) and specific factors were absent (q = 0). ♦ – G(EMBFA), ■ – Q(EMBFA), ▲ – G(BVA2), × – Q(BVA2), ∗ – G(M8), ● – Q(M8).

Fig. 4 illustrates the quality of methods for the opposite case, that is when common factors were distorted (p = 0.6) but specific factors were absent (q = 0). In this case, both BVA2 and M8 much lose EMBFA in information gain for all numbers of found factors. It means that the most factors found by these methods are false. As for the previous cases, all true factors were found by EMBFA when the number of assigned searched factors L slightly exceeds their actual number L = 16. When the number of found factors continues to increase, information gain decreases for both BVA2 and M8 but not for EMBFA. In contrast, coverage quality continues to increase for BVA2 and M8 but not for EMBFA. However, the coverage increases in this case only on account of false factors and, therefore, it does not serve revealing the latent regular structure of the dataset. Thus, thus only efficiency of EMBFA happened to be insensitive to a noise in the input data.

Note that for L = 19, when both indices Q and g for EMBFA reach maxima, coverage quality for EMBFA exceeds those for both BVA2 and M8 despite of the fact that these methods are claimed to maximize Q for any given L. This gives a good example when greedy algorithms used by these methods actually provide reasonable but not optimal solutions. Note also that sometimes EMBFA converges to zero solution (as shown in Fig. 4 in our computation it happened in case of L = 4). To overcome this problem it is enough to restart it with another seed of initial values p_{ij}.

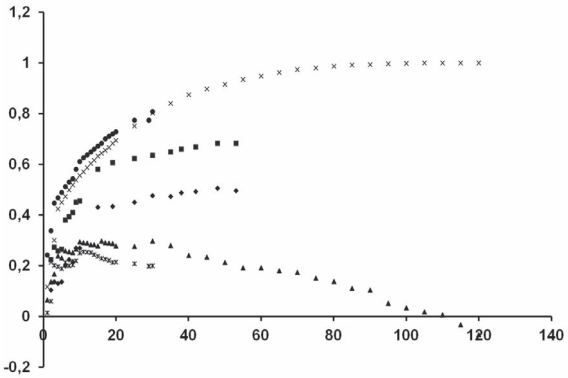

Fig. 5. Dependency of gain G and coverage Q on the number of found factors is depicted for Mushroom dataset. ♦ – G(EMBFA), ■ – Q(EMBFA), ▲ – G(BVA2), × – Q(BVA2), ∗ – G(M8), ● – Q(M8).

The performance of the methods applied to the Mushroom dataset is shown in Fig. 5. All the methods demonstrates similar results until the number of found factors L is less than 10. If L continues to increase the gain obtained by BVA2 and M8 decreases, but for EMBFA it continues to increase. Following the results obtained for artificial signals one can expect that the increase of coverage quality provided by BVA2 and M8 for L > 10 is explained only by the false factors. Unlike BVA2 and M8, information gain obtained by EMBFA increases monotonically. The drop of the increasing rate at L = 20 may be interpreted as the end of finding true factors.

VI. Discussion

The performance of both BMF and BFA methods were compared for artificial and real datasets. Since for artificial dataset its hidden factor structure is known in advance it is possible to estimate exactly the efficiency of methods in

revealing this structure. As artificial signals we used the superposition of horizontal and vertical bars at the grid of 8-by-8 pixels. Revealing the factor structure of these signals (solving the Bars Problem, BP) is a common benchmark [16], [15] to clarify strengths and weaknesses of BFA methods. We showed that both BVA2 and M8 methods are perfect in solving BP when noise in signals is absent. This is not surprising because in this case the search of factors is reduced to the dataset Boolean factorization. But both these methods were developed specially to perform this task. In contrast, EMBFA was developed for Boolean factor analysis based on given statistical generative model. As a result, EMBFA slightly loses on BVA2 and M8 in performance when noise-free signals are analyzed. To the contrary, both BVA2 and M8 lose on EMBFA when input signals are noisy. M8 is not able to reveal the hidden factor structure of the dataset for both kinds of noise: i.e. for factor distortion and specific factors appearance, while BVA2 is unable to do this only in the case of factor distortion.

To estimate the method‘s performance we used two indices: information gain G and coverage quality Q. The first index relates to BFA and the second one to BMF. As shown with artificial signals when the number of found factors L increases, these indices change sometimes identically, sometimes oppositely. Their identical change relates to the case when found factors are true and the reverse change to the case when they are false. This gives the heuristic criterion for identification of true and false factors in real datasets. For BVA2 and M8 applied to Mushroom dataset, both G and Q increase until L reaches 10. In this case, the factors found by all three methods are very similar. When $L > 10$, Q increases for all methods but for EMBFA G increases and for BVA2 and M8 it decreases. This can be interpreted as finding true factors by EMBFA and false factors by BVA2 and M8. Note that there is no notion “false factor” in methods of matrix factorization and therefore also no substantiated criterion for terminating the factors searching too. Contrary to this in BFA such criterion exists. It relates to the maximum of information gain: it is reasonable to continue the procedure of factors search until G increases and stop it when G reaches maximum or stops to increase. For the Mushroom datasets it increases until $L = 20$. Thus, one can expect that EMBFA found 20 but BVA2 and M8 only 10 factors which are suitable for further analysis by experts. However, it seems that the best strategy to reveal the latent structure of the dataset is to apply to its analysis all the methods and to compare the results to find the proper interpretation.

In spite of the fact that both BVA2 and M8 were developed to maximize coverage quality for every given number of factors during the run of algorithm, EMBFA is comparable with these methods even in this index. Thus, EMBFA could be used also for Boolean matrix factorization. The disadvantage of EMBFA is that sometimes it converges to unreasonable zero solution. But this problem can be easily overcome by restart with another initial seed of parameters.

ACKNOWLEDGMENT

This research has been partly funded by project GACR P202/10/0262, by the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme ’Research and Development for Innovations’ funded by Structural Funds of the European Union and state budget of the Czech Republic and by long-term strategic development financing of the Institute of Computer Science (RVO:67985807).

REFERENCES

[1] R. Belohlavek and V. Vychodil, “Discovery of optimal factors in binary data via a novel method of matrix decomposition,” *Journal of Computer and System Sciences*, vol. 76, no. 1, pp. 3–20, 2010.

[2] G. Golub and W. Kahan, “Calculating the singular values and pseudo-inverse of a matrix,” *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, vol. 2, no. 2, pp. pp. 205–224, 1965.

[3] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, p. 2003, 2003.

[4] D. Lee, H. Seung *et al.*, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[5] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994. [Online]. Available: http://dx.doi.org/10.1002/env.3170050203

[6] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis*. Wiley & Sons, New York, 2001.

[7] P. Miettinen, T. Mielikainen, A. Gionis, G. Das, and H. Mannila, “The discrete basis problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 10, pp. 1348–1362, 2008.

[8] B. Ganter, R. Wille, and R. Wille, *Formal concept analysis*. Springer Berlin, 1999.

[9] [Online]. Available: {http://www.statistical-solutions-software.com/bmdp-statistical-software/boolean-factor-analysis/}

[10] A. A. Frolov, D. Husek, and P. Y. Polyakov, “Boolean factor analysis by expectation-maximization method,” in *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, ser. Advances in Intelligent Systems and Computing, M. Kudelka, J. Pokorny, V. Snasel, and A. Abraham, Eds. Springer Berlin Heidelberg, 2013, vol. 179, pp. 243–254.

[11] A. Frolov, D. Husek, and P. Polyakov, “Estimation of Boolean factor analysis performance by informational gain,” in *ADVANCES IN INTELLIGENT WEB MASTERING-2, PROCEEDINGS*, ser. Advances in Intelligent and Soft Computing, Snasel, V. and Szczepaniak, P.S. and Abraham, A. and Kacprzyk, J, Ed., vol. 67, 2010, pp. 83–94, 6th Atlantic Web Intelligence Conference, Prague, CZECH REPUBLIC, SEP, 2009.

[12] ——, “New measure of Boolean factor analysis quality,” in *Adaptive and Natural Computing Algorithms*, ser. Lecture Notes in Computer Science, A. Dobnikar, U. Lotric, and B. Ster, Eds. Springer Berlin / Heidelberg, 2011, vol. 6593, pp. 100–109.

[13] A. Asuncion and D. J. Newman, “UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences,” 2007. [Online]. Available: {http://www.ics.uci.edu/~mlearn/MLRepository.html}

[14] P. Foldiak, “Forming sparse representations by local anti-hebbian learning,” *Biological Cybernetics*, vol. 64, p. 165170, 1990.

[15] J. Lücke and M. Sahani, “Maximal causes for non-linear component extraction,” *The Journal of Machine Learning Research*, vol. 9, pp. 1227–1267, 2008.

[16] M. W. Spratling, “Learning image components for object recognition,” *Journal of Machine Learning Research*, vol. 7, pp. 793–815, 2006.