C-SALT: Mining Class-Specific ALTerations in Boolean Matrix Factorization

Sibylle Hess and Katharina Morik

TU Dortmund University, Computer Science 8, Dortmund, Germany
{sibylle.hess,katharina.morik}@tu-dortmund.de,
http://www-ai.cs.uni-dortmund.de/PERSONAL/{morik,hess}.html

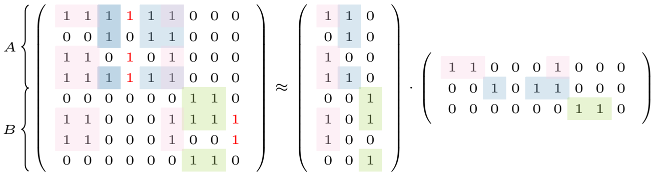
Abstract. Given labeled data represented by a binary matrix, we consider the task to derive a Boolean matrix factorization which identifies commonalities and specifications among the classes. While existing works focus on rank-one factorizations which are either specific or common to the classes, we derive class-specific alterations from common factorizations as well. Therewith, we broaden the applicability of our new method to datasets whose class-dependencies have a more complex structure. On the basis of synthetic and real-world datasets, we show on the one hand that our method is able to filter structure which corresponds to our model assumption, and on the other hand that our model assumption is justified in real-world application. Our method is parameter-free.

Keywords: Boolean matrix factorization, shared subspace learning, nonconvex optimization, proximal alternating linearized optimization

1 Introduction

When given labeled data, a natural instinct for a data miner is to build a discriminative model that predicts the correct class. Yet in this paper we put the focus on the characterization of the data with respect to the label, i.e., finding similarities and differences between chunks of data belonging to miscellaneous classes. Consider a binary matrix where each row is assigned to one class. Such data emerge from fields such as gene expression analysis, e.g., a row reflects the genetic information of a cell, assigned to one tissue type (primary/relapse/no tumor), market basket analysis, e.g., a row indicates purchased items at the assigned store, or from text analyses, e.g., a row corresponds to a document/article and the class denotes the publishing platform. For various applications a characterization of the data with respect to classes is of particular interest. In genetics, filtering the genes which are responsible for the reoccurrence of a tumor may introduce new possibilities for personalized medicine [?]. In market basket analysis it might be of interest which items sell better in some shops than others and in text analysis one might ask about variations in the vocabulary used when reporting from diverse viewpoints.

These questions are approached as pattern mining [?] and Boolean matrix factorization problems [?]. Both approaches search for factors or patterns which



$$\begin{array}{c|ccccccccc} A & 1&1&1&1&1&1&0&0&0\\ & 0&0&1&0&1&1&0&0&0\\ & 1&1&0&1&0&1&0&0&0\\ & 1&1&1&1&1&1&0&0&0\\ B & 0&0&0&0&0&0&1&1&0\\ & 1&1&0&0&0&1&1&1&1\\ & 1&1&0&0&0&1&0&0&1\\ & 0&0&0&0&0&0&1&1&0 \end{array} \approx \begin{pmatrix} 1&1&0\\ 0&1&0\\ 1&0&0\\ 1&1&0\\ 0&0&1\\ 1&0&1\\ 1&0&0\\ 0&0&1 \end{pmatrix} \cdot \begin{pmatrix} 1&1&0&0&0&1&0&0&0\\ 0&0&1&0&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&0 \end{pmatrix}$$

Fig. 1: A Boolean factorization of rank three. The data matrix on the left is composed by transactions belonging to two classes A and B. Each outer product is highlighted. Best viewed in color.

occur in both or only one of the classes. This is illustrated in Fig. 1; a data matrix is indicated on the left, whose rows are assigned to one class, A or B. While the pink outer product spreads over both classes, the blue and green products concentrate in only one of the classes. We refer to the factorizations of the first kind as common and to those of the second kind as class-specific.

The identification of class specific and common factorizations is key to a characterization of similarities and differences among the classes. Yet, what if meaningful deviations between the classes are slightly hidden underneath an overarching structure? The factorization in Fig. 1 is not exact, we can see that the red colored ones in the data matrix are not taken into account by the model. This is partially desired as the data is expected to contain noise which is supposedly filtered by the model. On the other hand, we can observe concurrence of the red ones and the pink factors – in each class.

1.1 Main Contributions

In this paper we propose a novel Boolean Matrix Factorization (BMF) method which is suitable to compare horizontally concatenated binary data matrices originating from diverse sources or belonging to various classes. To the best of the authors knowledge, this is the first method in the field of matrix factorizations of any kind, combining the properties listed below in one framework:

1. the method can be applied to compare any number of classes or sources,
2. the factorization rank is automatically determined; this includes the number of outer products, which are common among multiple classes, but also the number of discriminative outer products occurring in only one class,
3. in addition to discriminative rank-one factorizations, more subtle characteristics of classes can be derived, pointing out how common outer products deviate among the classes.

While works exist which approach one of the points 1 or 2 (see Sec. 2.2), the focus on subtle deviations among the classes as addressed in point 3 is entirely new. This expands the applicability of the new method to datasets where deviations among the classes have a more complex structure.

2 Preliminaries

We identify items $\mathcal{I} = \{1, \ldots, n\}$ and transactions $\mathcal{T} = \{1, \ldots, m\}$ by a set of indices of a binary matrix $D \in \{0,1\}^{m \times n}$. This matrix represents the data, having $D_{ji} = 1$ iff transaction j contains item i. A set of items is called a *pattern*.

We assume that the data matrix is composed of various sources, identified by an assignment from transactions to classes. Denoting by $[A^{(a)}]_a$ the matrix vertically concatenating the matrices $A^{(a)}$ for $a \in \{1, \ldots, c\}$, we write

$$D = \left[D^{(a)}\right]_a, \; Y = \left[Y^{(a)}\right]_a \text{ and } V^T = \left[V^{(a)T}\right]_a. \tag{1}$$

The $(m_a \times n)$-matrix $D^{(a)}$ comprises the $m_a < m$ transactions belonging to class a. Likewise, we explicitly notate the class-related $(m_a \times r)$- and $(n \times r)$-dimensional parts of the $m \times r$ and $n \times rc$ factor matrices Y and V as $Y^{(a)}$ and $V^{(a)}$. These factor matrices are properly introduced in Sec. 2.3.

We often employ the function θ_t which rounds a real value $x \geq t$ to one and $x < t$ to zero. We abbreviate $\theta_{0.5}$ to θ and denote with $\theta(X)$ the entry-wise application of θ to a matrix X. We denote matrix norms as $\| \cdot \|$ for the Frobenius norm and $| \cdot |$ for the entry-wise 1-norm. We express with $x^{m \times n}$ the $(m \times n)$-dimensional matrix having all entries equal to x. The operator $\circ$ denotes the Hadamard product. Finally, we denote with log the natural logarithm.

2.1 Boolean Matrix Factorization in Brief

Boolean Matrix Factorization (BMF) assumes that the data $D \in \{0,1\}^{m \times n}$ originates from a matrix product with some noise, i.e.,

$$D = \theta(YX^T) + N, \tag{2}$$

where $X \in \{0,1\}^{n \times r}$ and $Y \in \{0,1\}^{m \times r}$ are the factor matrices of rank r and $N \in \{-1,0,1\}^{m \times n}$ is the noise matrix. The Boolean product conjuncts r matrices; the outer products $Y_{\cdot s} X_{\cdot s}^T$ for $1 \leq s \leq r$. We use θ to denote the Boolean conjunction in terms of elementary algebra. Each outer product is defined by a pattern, indicated by $X_{\cdot s}$, and a set of transactions using the pattern, indicated by $Y_{\cdot s}$. Correspondingly, X is called the pattern and Y the usage matrix.

Unfortunately, solving X and Y from Eq. (2), if only the data matrix D is known, is generally not possible. Hence, surrogate tasks are formulated in which the data is approximated by a matrix product according to specific criteria. The most basic approach is to find the factorization of given rank which minimizes the residual sum of absolute values $|D - \theta(YX^T)|$. This problem, however, cannot be approximated within any factor in polynomial time (unless $\mathbf{NP} = \mathbf{P}$) [**?**].

BMF has a very popular relative, called Nonnegative Matrix Factorization (NMF). Here, a nonnegative data matrix $D \in \mathbb{R}_+^{m \times n}$ is approximated by the product of nonnegative matrices $X \in \mathbb{R}_+^{n \times r}$ and $Y \in \mathbb{R}_+^{m \times r}$. NMF tasks often involve minimizing the Residual Sum of Squares (RSS) $\frac{1}{2}\|D - YX^T\|^2$ [**?**]. Minimizing the RSS subject to binary matrices X and Y introduces the task of binary matrix factorization [**?**].

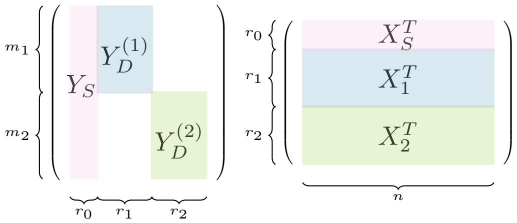


Fig. 2: A Boolean product identifying common (pink) and class-specific outer products (blue and green). Best viewed in color.

2.2 Related Work

If the given data matrix is class-wise concatenated (cf. Eq. (1)), a first approach for finding class-defining characteristics is to separately derive factorizations for each class. However, simple approximation measurements as discussed in Sec. 2.1 are already nonconvex and have multiple local optima. Due to this vagueness of computed models, class-wise factorizations are not easy to interpret; they lack a view on the global structure. Puzzling together the (parts of) patterns defining (dis-)similarities of classes afterwards, is non-trivial.

In the case of nonnegative, labeled data matrices, measures such as Fisher's linear discriminant criterion are minimized to derive weighted feature vectors, i.e., patterns in the binary case, which discriminate most between classes. This variant of NMF is successfully implemented for classification problems such as face recognition [**?**] and identification of cancer-associated genes [**?**].

For social media retrieval, Gupta et al. introduce Joint Subspace Matrix Factorization (JSMF) [**?**]. Focusing on the two-class setting, they assume that data points (rows of the data matrix) emerge not only from discriminative but also from common subspaces. JSMF infers for a given nonnegative data matrix and ranks r_0, r_1 and r_2 a factorization as displayed in Fig. 2. Multiplicative updates minimize the weighted sum of class-wise computed RSS. In Regularized JSNMF (RJSNMF), a regularization term is used to prevent that shared feature vectors swap into discriminative subspaces and vice versa [**?**]. The arising optimization problem is solved by the method of Lagrange multipliers. Furthermore, a provisional method to determine the rank automatically is evaluated. However, this involves multiple runs of the algorithm with increasing rank of shared and discriminative subspaces, until the approximation error barely decreases. A pioneering extension to the multi-class case is provided in [**?**].

Miettinen [**?**] transfers the objective of JSMF into Boolean algebra, solving

$$\min_{X,Y} \sum_{a\in\{1,2\}} \frac{\mu_a}{2} \left| D^{(a)} - \theta\left(\begin{bmatrix} Y_S^{(a)} & Y_D^{(a)} \end{bmatrix} \begin{bmatrix} X_S^T \\ X_a^T \end{bmatrix}\right)\right|$$

for binary matrices D, X and Y, and normalizing constants $\mu_{1/2}^{-1} = |D^{(2/1)}|$. A variant of the BMF algorithm Asso [**?**] governs the minimization. A provisional determination of ranks based on the Minimum Description Length (MDL) principle is proposed, computing which of the candidate rank constellations yields the lowest description length. The description length captures model complexity and data fit, and is hence suitable for model order selection [**?**,**?**].

Budhatoki and Vreeken [**?**] pursue the idea of MDL to derive a set of pattern sets, which characterizes similarities and differences of groups of classes. Identifying the usage of each pattern with its support in the data, the number of derived patterns equates the rank in BMF. In this respect, their proposed algorithm DiffNorm automatically determines the ranks in the multi-class case. However, the posed constraint on the usage often results in vast amount of returned patterns.

In the two-class nonnegative input matrices case, Kim et al. improve over RJSNMF by allowing small deviations from shared patterns in each class [**?**]. They found that shared patterns are often marginally altered according to the class. In this paper, we aim at finding these overlooked variations of shared patterns together with strident differences among multiple classes, combining the strengths of MDL for rank detection and the latest results in NMF.

2.3 (Informal) Problem Definition

Given a binary data matrix composed from multiple classes, we assume that the data has an underlying model similar to the one in Fig. 1. There are common or shared patterns (pink) and class-specific patterns (blue and green). Furthermore, there are class-specific patterns, which align within a subset of the classes where a pattern is used (the red ones). We call such aligning patterns class-specific alterations and introduce the matrix V to reflect these.

Definition 1. *Let* $X \in \{0,1\}^{n\times r}$ *and* $V \in \{0,1\}^{n\times cr}$*. We say the matrix* V *models* class-specific alterations of X *if* $\|X \circ V^{(a)}\| = 0$ *for all* $1 \leq a \leq c$*, and* $\|V^{(1)} \circ \ldots \circ V^{(c)}\| = 0$.

Similar to the data decomposition denoted in Eq. (2), we assume that data emerges from a Boolean matrix product; yet, we now consider multiple products, one for each class, which are defined by the class-wise alteration matrix V, its pattern matrix, usage and the noise matrix $N = [N^{(a)}]_a$, such that for $1 \leq a \leq c$

$$D^{(a)} = \theta\left(Y^{(a)}(X + V^{(a)})^T\right) + N^{(a)}. \tag{3}$$

Given a class-wise composed binary data matrix, we consider the task to filter the factorization, defined by X, Y and V, from the noise.

3 The Proposed Method

We build upon the BMF algorithm Primp, which combines recent results from numerical optimization with MDL in order to return interpretable factorizations of a suitably estimated rank [**?**]. The employed description length f reflects the size of the data encoded by a code table as known from algorithms Slim and Krimp [**?**,**?**]. Determining a smooth function F, bounding the description length from above, and a function ϕ to penalize non-binary values, locally minimizing matrices of the relaxed objective $F(X,Y)+\phi(X)+\phi(Y)$ are derived. Rounding the local minimizers to binary matrices according to the description length, yields the final result and decides over the rank of the factorization.

The numerical optimization is performed by *Proximal Alternating Linearized Minimization* (PALM) [**?**]. That are alternatingly invoked *proximal mappings* with respect to ϕ from the gradient descent update with respect to F (cf. lines 6,8 and 10 in Algorithm 1). The proximal mapping of ϕ returns a matrix satisfying the following minimization criterion:

$$\operatorname{prox}_{\phi}(X) \in \arg\min_{\hat{X}} \left\{ \frac{1}{2}\|X-\hat{X}\|^2 + \phi(\hat{X}) \right\}.$$

Loosely speaking, X is given a little push into a direction minimizing ϕ. We choose $\phi(X)=\sum_{i,j}\Lambda(X_{ij})$ to penalize non-binary matrix-entries by an entry-wise application of the function Λ. Correspondingly, the prox-operator is computed entry-wise $\operatorname{prox}_{\alpha\phi}(X)=(\operatorname{prox}_{\alpha\Lambda}(X_{ji}))_{ji}$, where

$$\Lambda(x)=\begin{cases}-|1-2x|+1 & x\in[0,1]\\ \infty & x\notin[0,1].\end{cases},\quad \operatorname{prox}_{\alpha\Lambda}(x)=\begin{cases}\max\{0,x-2\alpha\} & x\leq 0.5\\ \min\{1,x+2\alpha\} & x>0.5.\end{cases}$$

Notice, the proximal mapping ensures that factor matrices always attain values between zero and one. For further information on prox-operators, see, e.g., [**?**].

The step sizes of the gradient descent updates are computed by the Lipschitz moduli of partial gradients (cf. lines 5, 7 and 9 in Algorithm 1). Assuming that the infimum of F and ϕ exists and ϕ is proper and lower continuous, PALM generates a nonincreasing sequence of function values which converges to a critical point of the relaxed objective.

3.1 C-Salt

In order to capture class-defining characteristics in the framework of Primp, few extensions have to be made. We pose two requirements on the interplay between usage and class-specific alterations of patterns: class-specific alterations ought to fit very well to the corresponding class but as little as possible to other classes. We introduce a regularizing function to penalize nonconformity to this request.

$$\begin{aligned}S(Y,V) &= \sum_{s=1}^{r}\sum_{a=1}^{c}\left(\left|Y_{\cdot s}^{(a)}\right|\left|V_{\cdot s}^{(a)}\right| - {Y^{(a)}}^{T}D^{(a)}V_{\cdot s}^{(a)}\right) + \sum_{b\neq a}{Y_{\cdot s}^{(b)}}^{T}D^{(b)}V_{\cdot s}^{(a)}\\ &= \sum_{a=1}^{c}\operatorname{tr}\left(\left({Y^{(a)}}^{T}(1^{m_a\times n}-2D^{(a)})+Y^{T}D\right)V^{(a)}\right).\end{aligned}$$

We extend the description length of Primp such that class-specific alterations are encoded in the same way as patterns; by standard codes, assigning item

Algorithm 1 C-Salt($D = [D^{(a)}]_a; \mathit{\Delta}_r = 10, \gamma = 1.00001$)

```
1: $(X_K, V_K, Y_K) \leftarrow (\emptyset, \emptyset, \emptyset)$
2: for $r \in \{\mathit{\Delta}_r, 2\mathit{\Delta}_r, 3\mathit{\Delta}_r, \ldots\}$ do
3:     $(X_0, V_0, Y_0) \leftarrow$IncreaseRank$(X_K, V_K, Y_K, \mathit{\Delta}_r)$    ▷ Append random columns
4:     for $k = 0, 1, \ldots$ do    ▷ Select stop criterion
5:         $1/\alpha_k \leftarrow \gamma M_{\nabla_X F}(V_k, Y_k)$
6:         $X_{k+1} \leftarrow \mathrm{prox}_{\alpha_k \phi}(X_k - \alpha_k \nabla_X F(X_k, V_k, Y_k))$
7:         $1/\nu_k^{(a)} \leftarrow \gamma M_{\nabla_V^{(a)} F}(X_{k+1}, Y_k)$    ▷ $1 \leq a \leq c$
8:         $V_{k+1}^{(a)} \leftarrow \mathrm{prox}_{\nu_k^{(a)} \phi}\left(V_k^{(a)} - \nu_k^{(a)} \nabla_V^{(a)} F(X_{k+1}, V_k^{(a)}, Y_k)\right)$    ▷ $1 \leq a \leq c$
9:         $1/\beta_k \leftarrow \gamma M_{\nabla_Y F}(X_{k+1}, V_{k+1})$
10:        $Y_{k+1} \leftarrow \mathrm{prox}_{\beta_k \phi}(Y_k - \beta_k \nabla_Y F(X_{k+1}, V_{k+1}, Y_k))$
11:    $(X, V, Y) \leftarrow$ Round$(f, X_k, V_k, Y_k)$    ▷ Try thresholds from finite set
12:    if $r - r(X, V, Y) > 1$ then return $(X, V, Y)$ end if
```

$i \in \mathcal{I}$ a code of length $u_i = -\log\left(|D_{\cdot i}|/|D|\right)$. The objective function f adds the description length to the specificity-regularizer

$$f(X,V,Y) = -\sum_{s:|Y_{\cdot s}|>0}\left((|Y_{\cdot s}|+1)\cdot\log\left(\frac{|Y_{\cdot s}|}{|Y|+|N|}\right) + X_{\cdot s}^T u + \sum_a {V_{\cdot s}^{(a)}}^T u\right)$$
$$-\sum_{i:|N_{\cdot i}|>0}\left((|N_{\cdot i}|+1)\cdot\log\left(\frac{|N_{\cdot i}|}{|Y|+|N|}\right) + u_i\right) + S(Y,V).$$

This determines the relaxed objective $F(X,V,Y) + \phi(X) + \phi(V) + \phi(Y)$, where

$$F(X,V,Y) = \frac{1}{2}\left(\mu\sum_{a=1}^{c}\|D^{(a)} - Y^{(a)}(X+V^{(a)})^T\|^2 + G(X,V,Y) + S(Y,V)\right),$$

$\mu = 1 + \log(n)$ and G is defined as stated in Appendix A. F has Lipschitz continuous gradients and is suitable for PALM.

Algorithm 1 details C-Salt, which largely follows the framework of Primp [**?**]. C-Salt has as input the data D and two parameters, for which default values are given, which rarely need to be adjusted in practice. Further information about the robustness and significance of these parameters is provided in 1. For step-wise increased ranks, PALM optimizes the relaxed objective (lines 4-10). Note that the alternating minimization of more than two matrices corresponds to the extension of PALM for multiple blocks, discussed in [**?**]. The required gradients and Lipschitz moduli are stated in Appendix A. Subsequently, a rounding procedure returns the binary matrices $X_{t_1} = \theta_{t_1}(X_K)$, $V_{t_1} = \theta_{t_1}(V_K)$ and $Y_{t_2} = \theta_{t_2}(Y_K)$ for thresholds $t_1, t_2 \in \{0.05k \mid k \in \{0, 1, \ldots, 20\}\}$ minimizing f. Thereby, the validity of Definition 1 is ensured by setting unsuitable values in V to zero. Furthermore, *trivial* outer products covering fewer than two transactions or items are removed. The number of remaining outer products defines the rank $r(X, V, Y)$. If the gap between the number of possibly and actually modeled outer products is larger than one, the current factorization is returned (line 12).

4 Experiments

The experimental evaluations concern the following research questions:

1. Given that the data matrix is generated as stated by the informal problem definition in Sec. 2.3, does C-Salt find the original data structure?
2. Is the assumption that real-world data emerge as stated in Eq. (3) reasonable, and what effect has the modeling of class-specific alterations on the results?

We compare against the algorithms Dbssl, the dominated approach proposed in [**?**], and Primp[1]. The first question is approached by a series of synthetic datasets, generated according to Eq. (3). To address the second question, we compare on real-world datasets the RSS, computed factorization ranks and visually inspect derived patterns. Furthermore, we discuss an application in genome analysis where none of the existing methods provides the crucial information.

For C-Salt and Primp we use as stop criterion a minimum average function decrease (of last 500 iterations) of 0.005 and maximal $k \leq 10,000$ iterations. We use the Matlab/C implementation of Dbssl which has been kindly provided by the authors upon request. Setting the minimum support parameter of the employed FP-Growth algorithm proved tricky. Choosing the minimum support too low results in a vast memory consumption (we provided 100GiB RAM); setting it too high yields too few candidate patterns. Hence, this parameter varies between experiments within the range {2,. . . ,8}.

C-Salt is implemented for GPU, as is Primp. We provide the source code of our algorithms together with the data generating script [2].

4.1 Measuring the Quality of Factorizations

For synthetic datasets, we compare the computed models against the planted structure by an adaptation of the micro-averaged F-measure. We assume that generated matrices $X^\star, V^\star, Y^\star$ and computed models X, V, Y have the same rank r. Otherwise, we attach columns of zeros to make them match. We compute one-to-one matchings $\sigma_1 : \{1,\ldots,r\} \to \{1,\ldots,r\}$ between outer products of computed and generated matrices by the Hungarian algorithm [**?**]. The matching maximizes $\sum_{s=1}^{r} F^{(a)}_{s,\sigma_1(s)}$, where

$$F^{(a)}_{S,T} = 2\frac{\mathrm{pre}^{(a)}_{S,T} \cdot \mathrm{rec}^{(a)}_{S,T}}{\mathrm{pre}^{(a)}_{S,T} + \mathrm{rec}^{(a)}_{S,T}},$$

for selections of columns S and T. $\mathrm{pre}^{(a)}_{S,T}$ and $\mathrm{rec}^{(a)}_{S,T}$ denote precision and recall w.r.t. the denoted column selection. Writing $X^{(a)} = X + V^{(a)}$, we compute

$$\mathrm{pre}^{(a)}_{S,T} = \frac{\left|(Y^\star_{\cdot S} \circ Y_{\cdot T})^{(a)}\,(X^\star_{\cdot S} \circ X_{\cdot T})^{(a)^T}\right|}{\left|Y^{(a)}_{\cdot T} X^{(a)^T}_{\cdot T}\right|}, \quad \mathrm{rec}^{(a)}_{S,T} = \frac{\left|(Y^\star_{\cdot S} \circ Y_{\cdot T})^{(a)}\,(X^\star_{\cdot S} \circ X_{\cdot T})^{(a)^T}\right|}{\left|Y^{\star\,(a)}_{\cdot S} X^{\star\,(a)^T}_{\cdot S}\right|}.$$

[1] http://sfb876.tu-dortmund.de/primp

[2] http://sfb876.tu-dortmund.de/csalt

We calculate then precision and recall such that planted outer products with indices $R = (1, \dots, r)$ are compared to outer products of the computed factorization with indices $\sigma_1(R) = (\sigma_1(1), \dots, \sigma_1(r))$. The corresponding F-measure is the micro F-measure, which is identified by $F^{(a)}_{R,\sigma_1(R)}$.

Since class-specific alterations of patterns, reflected by the matrix V, are particularly interesting in the scope of this paper, we additionally state the recall of $V^\star$, denoted by rec_V. Therefore, we compute a maximum matching σ_2 between generated class alterations $V^\star$ with usage $Y^\star$ and computed patterns $X_V = [X\ V]$ (setting V to the $(n \times cr)$ zero matrix for other algorithms than C-Salt) with usage $Y_V = [Y \dots Y]$ (concatenating c times). The recall $\mathrm{rec}^{(a)}_{R,\sigma_2(R)}$ is then computed with respect to the matrices $V^\star, Y^\star, X_V$ and Y_V. Furthermore, we compute the class-wise factorization rank $r^{(a)}$ as the number of nontrivial outer products, involving more than only one column or row. Outer products where solely one item or one transaction is involved yield no insight for the user and are therefore always discarded. In following plots, we indicate averaged measures over all classes

$$F = \frac{1}{c}\sum_a F^{(a)}_{R,\sigma_1(R)}, \quad \mathrm{rec}_V = \frac{1}{c}\sum_a \mathrm{rec}^{(a)}_{R,\sigma_2(R)} \text{ and } r = \frac{1}{c}\sum_a r^{(a)}.$$

Therewith, the size of the class is not taken into account; the discovery of planted structure is considered equally important for every class. F-measure and recall have values between zero and one. The closer both approach one, the more similar are the obtained and planted factorizations.

4.2 Synthetic Data Generation

We state the synthetic data generation as a procedure which receives the matrix dimensions $(m_a)_a$ ($m = \sum_a m_a$) and n, the factorization rank $r^\star$, matrix $C \in \{0,1\}^{c \times r}$ and noise probability p as input. The matrix C indicates for each pattern in which classes it is used.

```
GenerateData($n, (m_a)_a, r^\star, C, p$)
 1. Draw the $(n \times r^\star)$ and $(m \times r^\star)$ matrices $X^\star$, ${V^{(a)}}^\star$ and $Y^\star$ uniformly random
    from the set of all binary matrices subject to
      – each column $X^\star_{\cdot s}(Y^\star_{\cdot s})$ has at least $n/100(m/100)$ uniquely assigned bits,
      – the density is bounded by $|X^\star_{\cdot s}| \leq n/10$ and $|{Y^{(a)}_{\cdot s}}^\star| \leq C_{sa} m_a/10$
      – ${V^{(a)}}^\star$ models class-specific alterations of $X^\star$ and $\left|\sum_{a=1}^{c} {V^{(a)}_s}^\star\right| \leq 2/3\,|X^\star_{\cdot s}|$
 3. Set $D^{(a)}$, flipping every bit of $\theta\left({Y^{(a)}}^\star (X^\star + {V^{(a)}}^\star)^T\right)$ with probability $p$.
```

By default, the parameters $r^\star = 24$, $m_a = m/2$, where m and n are varied as described in Sec. 4.3, $p = 0.1$, and depending on the number of classes we set

$$C_2 = \left[\begin{pmatrix} 1\,0\,1 \\ 1\,1\,0 \end{pmatrix}\right]_{\frac{r^\star}{3}}, \quad C_3 = \left[\begin{pmatrix} 1\,1\,0\,0 \\ 1\,0\,1\,0 \\ 1\,0\,1\,1 \end{pmatrix}\right]_{\frac{r^\star}{4}}, \quad C_4 = \left[\begin{pmatrix} 1\,1\,0\,0\,0 \\ 1\,0\,1\,0\,0 \\ 1\,0\,1\,1\,0 \\ 1\,0\,1\,1\,1 \end{pmatrix}\right]_{\frac{r^\star}{5}}.$$

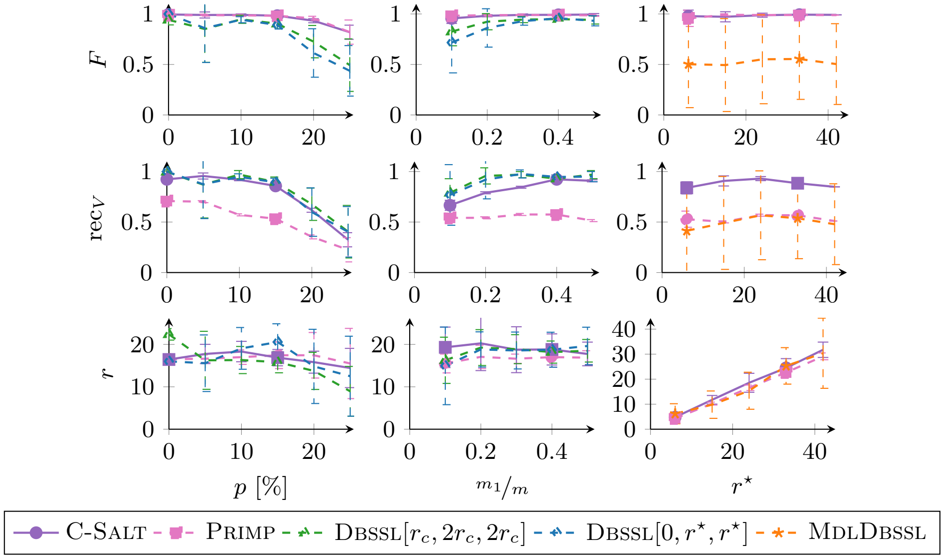


Fig. 3: Variation of noise (left column), class distribution m_1/m (middle column) and the rank (right column). The F-measure, recall of the matrix V (both the higher the better) and the class-wise estimated rank of the calculated factorization is plotted against the varied parameter. Best viewed in color.

4.3 Synthetic Data Experiments

We plot for the following series of experiments the averaged F-measure, recall rec_V, and the rank (cf. Sec. 4.1), against the parameter varied when generating the synthetic data (see Sec. 4.2). Error bars have length 2σ. For every experiment, we generate eight matrices: two for each combination of dimensions $(n, m) \in \{(500, 1600), (1600, 500), (800, 1000), (1000, 800)\}$.

Fig. 3 contrasts the results of C-Salt, Primp and Dbssl in the two-class setting. For Dbssl, we consider two instantiations if the rank $r^\star$ is fixed. Both correctly reflect the number of planted specific and common patterns, yet the one rates class-specific alterations as separate patterns and the other counts every pattern with its class-specific alteration as a class-specific pattern. In the experiments varying the rank, we employ the MDL-based selection of the rank proposed for Dbssl. The input candidate constellations of class-specific and common patterns are determined according to the number of planted patterns, i.e., candidate rank constellations are a combination of $r_0 \in r^\star/3 \pm \{5, 0\}$ and $r_1 = r_2 \in \{kr^\star/3 \mid k \in \{1, 2, 4\}\}$.

Fig. 3 shows the performance measures of the competing algorithms when varying three parameters: noise p (left column), ratio of transactions in each class m_1/m (middle column) and rank $r^\star$ (right column). We observe an overall high F-measure of C-Salt and Primp. Both Dbssl instantiations also obtain

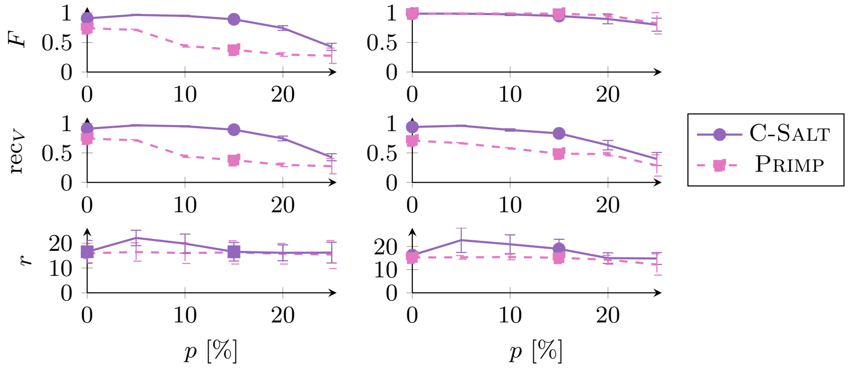


Fig. 4: Variation of noise for generated data matrices with three (left) and four classes (right). The F-measure, recall of the matrix V (both the higher the better) and the class-wise estimated rank of the calculated factorization (between 16 and 24 can be considered correct) is plotted against the varied parameter. Best viewed in color.

high F-values, but only at lower noise levels and if one class is not very dominant over the other. C-Salt and Primp differ most notably in the discovery of class specific alterations measured by rec$_V$. C-Salt shows a similar recall as Dbssl if the noise is varied but a lower recall if classes are imbalanced. The ranks of returned factorizations by all algorithms lie in a reasonable interval, considering that class-specific alterations can also be interpreted as unattached patterns. Hence, a class-wise averaged rank between 16 and 24 is legitimate. When varying the number of planted patterns, the MDL selection procedure of the rank also yields correct estimations for Dbssl. However, the F-measure and recall of $V^\star$ decrease to 0.5 if the rank is not set to the correct parameters for Dbssl.

Fig. 4 displays the results of Primp and C-Salt when varying the noise for generated class-common and class-specific factorizations for three and four classes. The plots are similar to Fig. 3. The more complex constellations of class-overarching outer products, which occur when more than two classes are involved, do not notably affect the ability to discover class-specific alterations by C-Salt and the planted factorization by Primp and C-Salt.

4.4 Real-World Data Experiments

We explore the algorithms' behavior by three interpretable text-datasets depicted in Table 1. The datasets are composed by two classes to allow a comparison to Dbssl. The dimensions m_1 and m_2 describe how many documents belong to the first, respectively second class. Each document is represented by its occurring lemmatized words, excluding stop words. The dimension n reflects the number of words which occur in 20 documents at least. From the 20 News-

Table 1: Comparison of the amount of derived class-specific (r_1, r_2) and class-common patterns (r_0), the overall rank $r = r_0 + r_1 + r_2$ and the RSS of the BMF (scaled by 10^4) for real-world datasets. Values in parentheses correspond to factorizations where outer products with less than four items or transactions are discarded. The last two columns summarize characteristics of the datasets: number of rows belonging to the first and second class (m_1, m_2), number of columns (n) and density $d = |D|/(nm)$ in percent.

	Space-Rel				Politics				Movie			
	C-Salt	Primp	Dbssl1	Dbssl2	C-Salt	Primp	Dbssl1	Dbssl2	C-Salt	Primp	Dbssl1	Dbssl2
r	29(28)	30(30)	40(7)	18(6)	41(40)	30(30)	57(20)	42(15)	26(25)	30(27)	27(4)	12(4)
r_0	4(3)	8(8)	19(1)	7(1)	10(10)	8(8)	16(2)	5(0)	25(25)	29(27)	21(1)	6(0)
r_1	9(9)	8(8)	13(4)	8(4)	19(18)	15(15)	27(14)	18(11)	1(0)	1(0)	3(1)	3(1)
r_2	16(16)	14(14)	8(2)	3(1)	12(12)	7(7)	14(4)	19(4)	0(0)	0(0)	3(2)	3(3)
RSS	76(77)	76(76)	73(79)	76(79)	119(119)	122(122)	110(122)	116(123)	320(320)	319(319)	315(318)	316(318)
	m_1	m_2	n	d[%]	m_1	m_2	n	d[%]	m_1	m_2	n	d[%]
	622	980	2244	2.27	936	775	2985	2.64	998	997	4442	3.68

group corpus[3], we compose the *Space-Rel* dataset by posts from`sci.space` and `talk.religion.misc`, and the *Politics* dataset from `talk.politics.mideast` and `talk.politics.misc`. The *Movie* dataset is prepared from a collection of 1000 negative and 1000 positive movie reviews[4].

We consider two instantiations of Dbssl: Dbssl1 is specified by $r_0 = r_1 = r_2 = 30$ and Dbssl2 by $r_0 = r_1 = r_2 = 15$. For a fair comparison, we set a maximum rank of 30 for C-Salt and Primp. Therewith, the returned factorizations have a maximum rank of 90 for Dbssl1, 45 for Dbssl2, 30 for Primp and 60 for C-Salt. Note that C-Salt has the possibility to neglect X and use mainly V to reflect $cr = 60$ class-specific outer products. In practice, we consider patterns $V_{\cdot s}^{(a)} + X_{\cdot s}$ as individual class-specific patterns if $|V_{\cdot s}^{(a)}| > |X_{\cdot s}|$.

Table 1 shows the number of class-specific and common patterns, and the resulting RSS. Since outer products involving only a few items or transactions either provide little insight or are difficult to interpret, we also state in parentheses the values concerning *truncated factorizations*, i.e., outer products reflecting less than four items or transactions are discarded (glossing over the truncating of singletons, which is performed in both cases).

The untruncated factorizations obtained from Dbssl generally obtain a low RSS. However, when we move to the more interesting truncated factorizations, Dbssl suffers (the rank shrinks to less than a third for factorizations of Dbssl2). On the 20 News datasets this leads to a substantial RSS increase; C-Salt and Primp provide the lowest RSS in this case. We also observe, that the integration of the matrix V by C-Salt empowers the derivation of more class-specific factorizations than Primp. Nevertheless, both algorithms describe the Movie dataset only by class-common patterns. We inspect these results more closely in the next section, showing that mining class-specific alterations points at exclusively derived class characteristics, especially for the Movie dataset.

[3] `http://qwone.com/~jason/20Newsgroups/`

[4] `http://www.cs.cornell.edu/People/pabo/movie-review-data/`

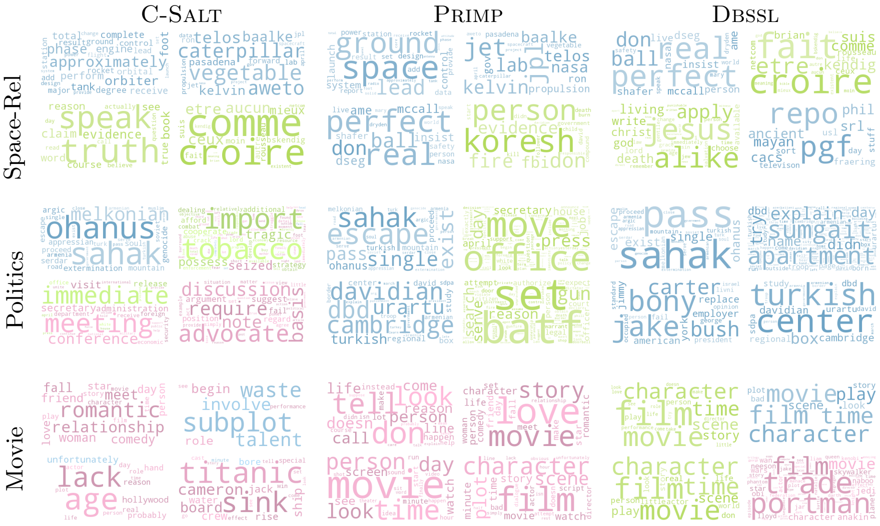


Fig. 5: Illustration of a selection of derived topics for the 20 News and Movie datasets. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^T D_{\cdot i}$) and the color its class affiliation: pink words are class-common, blue words belong to the first and green words to the second class. Best viewed in color.

4.5 Illustration of Factorizations

Let us inspect the derived most prevalent topics in the form of word clouds. Fig. 5 displays for every algorithm the top four topics, whose outer product spans the largest area. Class-common patterns are colored pink whereas class-specific patterns are blue or green. Class-specific alterations within topics become apparent by differently colored words in one word cloud. We observe that the topics displayed for the 20-News data are mostly attributed to one of the classes. The topics are generally interpretable and even comparable among the algorithms (cf. the first topic in the Politics dataset). Here, class-specific alterations of C-Salt point at the context in which a topic is discussed, e.g., the press release from the white house after a conference or meeting took place, whereby the latter may be discussed in both threads (cf. the third topic for the Politics dataset).

The most remarkable contribution of class-specific alterations is given for the movie dataset. Generally, movie reviews addressing a particular genre, actors, etc., are not exclusively bad or good. Primp and C-Salt derive accordingly only common patterns. Here, C-Salt can derive the decisive hint which additional words indicate the class membership. We recall from Table 1 that DBSSL returns in total four truncated topics for the Movie dataset. Thus, the displayed topics for the Movie dataset represent all the information we obtain from DBSSL. In addition, the topics display a high overlap in words, which underlines the reasonability of our assumption that minor deviations of major and common patterns can denote the sole class-distinctions.

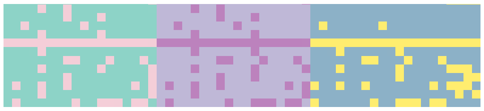


Fig. 6: Transposed usage matrix returned by C-Salt on the genome dataset. Class-memberships are signalized by colors.

Table 2: Average size and empirical standard deviation of patterns ($\cdot 10^3$) and class-specific alterations ($\cdot 10^3$).

$\lvert X \rvert$	$\lvert V^{(N)} \rvert$	$\lvert V^{(T)} \rvert$	$\lvert V^{(R)} \rvert$
10.7 ± 96	2.1 ± 2.5	3.6 ± 4.8	3.8 ± 6.6

4.6 Genome Data Analysis

The results depicted in the previous section are qualitatively easy to assess. We easily identify overlapping words and filter the important class characteristics from the topics at hand. In this experiment, the importance or meaning of features is unclear and researchers benefit from any summarizing information which is provided by the method, e.g., the common and class-specific parts of a pattern. We regard the dataset introduced in [**?**] representing the genomic profile of 18 Neuroblastoma patients. For each patient, samples are taken from three classes: *normal* (N), *primary tumor* (T) and *relapse tumor cell* (R). The data denotes loci and alterations taking place with respect to a reference genome. Alterations denote nucleotide variations such as $A \rightarrow C$, insertions ($C \rightarrow AC$) and deletions ($AC \rightarrow A$). One sample from each of the classes N and T is given for every patient ($m_N = m_T = 18$), one patient lacks one and another has three additional relapse samples ($m_R = 20$), resulting in $m = 56$ samples. We convert the alterations into binary features, each representing one alteration at one locus (position on a chromosome). The resulting matrix has $n \approx 3.7$ million columns.

C-Salt returns on the genome data a factorization of rank 28, of which we omit sixteen patterns solely occurring in one patient. Fig. 6 depicts the usage of the remaining twelve outer products, being almost identical for each class. Most notably, all derived patterns are class-common and describe the genetic background of patients instead of class characteristics. Table 2 summarizes the average length of patterns and corresponding class-specific alterations. We see that the average pattern reflects ten thousands of genomic alterations and that among the class-specific alterations, the ones which are attributed to relapse samples are highest in average. These results correspond to the evaluation in [**?**].

The information provided by C-Salt can not be extracted by existing methods. Primp yields only class-common patterns whose usage aligns with patients, regardless of classes. Running Primp separately on each class-related part $D^{(a)}$ yields factorizations of rank zero – the genomic alignments between patients can not be differentiated from noise for such few samples. However, using the framework of Primp to minimize the RSS without any regularization, yields about 15 patterns for each part $D^{(a)}$. The separately mined patterns overlap over the classes in an intertwined fashion. The specific class characteristics are not easily perceived for such complex dependencies and would require further applications of algorithms which structure the information from the sets of vast amounts of features.

5 Conclusion

We propose C-Salt, an explorative method to simultaneously derive similarities and differences among sets of transactions, originating from diverse classes. C-Salt solves a Boolean Matrix Factorization (BMF) by means of numerical optimization, extending the method Primp [?] to incorporate classes. We integrate a factor matrix reflecting class-specific alterations of outer products from a BMF (cf. Definition 1). Therewith, we capture class characteristics, which are lost by unsupervised factorization methods such as Primp. Synthetic experiments show that a planted structure corresponding to our model assumption is filtered by C-Salt (cf. Fig. 3). Even in the case of more than two classes, C-Salt filters complex dependencies among them (cf. Fig 4). These experiments also show that the rank is correctly estimated. On interpretable text data, C-Salt derives meaningful factorizations which provide valuable insight into prevalent topics and their class specific characteristics (cf. Table 1 and Fig 5). An analysis of genomic data underlines the usefulness of our new factorization method, yielding information which none if the existing algorithms can provide (cf. Sec. 4.6).

Acknowledgments Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, project C1 `http://sfb876.tu-dortmund.de`.

A Functions, Gradients and Lipschitz-Moduli

The functions, required by Algorithm 1, are stated in relation to $N = [N^{(a)}]_a$, as defined in Eq. (3). F, stated in Sec. 3.1, and its gradients are defined by

$$\begin{aligned}
G(X,V,Y) &= -\sum_{s=1}^{r}(|Y_{\cdot s}|+1)\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) + |X^T u| + \sum_{a=1}^{c}|{V^{(a)}}^T u| + |Y|,\\
\nabla_X F(X,V,Y) &= -\mu\sum_{a=1}^{c}{N^{(a)}}^T Y^{(a)} + u(0.5)^{1\times n},\\
\nabla_V^{(a)} F(X,V,Y) &= -\mu {N^{(a)}}^T Y^{(a)} + u(0.5)^{1\times n} + \nabla_V^{(a)} S(Y,V),\\
\nabla_Y^{(a)} F(X,V,Y) &= -\mu N^{(a)} X - \frac{1}{2}\left(\log\left(\frac{|Y_{\cdot s}|{+}1}{|Y|{+}r}\right) - 1\right)_{js} + \nabla_Y^{(a)} S(Y,V),\\
\nabla_V^{(a)} S(Y,V) &= D^T Y + (1^{m_a\times n} - 2D^{(a)})^T Y^{(a)}\\
\nabla_Y^{(a)} S(Y,V) &= D^{(a)}\left(\sum_{b\neq a} V^{(b)}\right) + (1^{m_a\times n} - D^{(a)})V^{(a)}.
\end{aligned}$$

The Lipschitz moduli are $M_{\nabla_X F}(Y,V) = \mu\|YY^T\|$, $M_{\nabla_V^{(a)} F}(X,Y) = \mu\|Y^{(a)}{Y^{(a)}}^T\|$ and $M_{\nabla_Y^{(a)} F}(X,V) = \mu\|(X{+}V^{(a)})(X{+}V^{(a)})^T\|{+}m_a$, $M_{\nabla_Y F}(X,V) = \|(M_{\nabla V^{(a)} F}(X,Y))_a\|$.

References

1. J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.
2. S. K. Gupta, D. Phung, B. Adams, T. Tran, and S. Venkatesh. Nonnegative shared subspace learning and its application to social media retrieval. In *KDD*, pages 1169–1178, 2010.
3. S. K. Gupta, D. Phung, B. Adams, and S. Venkatesh. Regularized nonnegative shared subspace learning. *DAMI*, 26(1):57–97, 2013.
4. S. K. Gupta, D. Phung, B. Adams, and S. Venkatesh. A matrix factorization framework for jointly analyzing multiple nonnegative data sources. In *Data Mining for Service*, pages 151–170. Springer, 2014.
5. S. Hess, K. Morik, and N. Piatkowski. The primping routine—tiling through proximal alternating linearized minimization. *DAMI*, 31(4):1090–1131, 2017.
6. H. Kim, J. Choo, J. Kim, C. K. Reddy, and H. Park. Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In *KDD*, pages 567–576, 2015.
7. H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
8. P. Miettinen. On finding joint subspace boolean matrix factorizations. In *SDM*, pages 954–965, 2012.
9. P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *TKDE*, 20(10):1348–1362, 2008.
10. P. Miettinen and J. Vreeken. Model order selection for boolean matrix factorization. In *KDD*, pages 51–59, 2011.
11. S. Nikitidis, A. Tefas, and I. Pitas. Projected gradients for subclass discriminant nonnegative subspace learning. *IEEE transactions on cybernetics*, 44(12):2806–2819, 2014.
12. O. Odibat and C. K. Reddy. Efficient mining of discriminative co-clusters from gene expression data. *Knowledge and information systems*, 41(3):667–696, 2014.
13. N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3), 2014.
14. A. Schramm, J. Köster, Y. Assenov, K. Althoff, M. Peifer, E. Mahlow, A. Odersky, D. Beisser, C. Ernst, A. Henssen, H. Stephan, C. Schröder, L. Heukamp, A. Engesser, Y. Kahlert, J. Theissen, B. Hero, F. Roels, J. Altmüller, P. Nürnberg, K. Astrahantseff, C. Gloeckner, K. De Preter, C. Plass, S. Lee, H. Lode, K. Henrich, M. Gartlgruber, F. Speleman, P. Schmezer, F. Westermann, S. Rahmann, M. Fischer, A. Eggert, and J. Schulte. Mutational dynamics between primary and relapse neuroblastomas. *Nature Genetics*, 47(8):872–877, 2015.
15. A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *SDM*, volume 6, pages 393–404, 2006.
16. K. Smets and J. Vreeken. Slim: Directly mining descriptive patterns. In *SDM*, pages 236–247, 2012.
17. J. Vreeken, M. van Leeuwen, and A. Siebes. Characterising the difference. In *KDD*, pages 765–774, 2007.
18. Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *TKDE*, 25(6):1336–1353, 2013.
19. Z. Zhang, C. Ding, T. Li, and X. Zhang. Binary matrix factorization with applications. In *ICDM*, pages 391–400, 2007.