A Mathematical Theory of Making Hard Decisions: Model Selection and Robustness of Matrix Factorization with Binary Constraints

DISSERTATION

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Sibylle Charlotte Heß

Dortmund
2018

Tag der mündlichen Prüfung:	25. Juni 2019
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr. Katharina J. Morik
	Prof. Dr. Arno P.J.M. Siebes

ABSTRACT

One of the first and most fundamental tasks in machine learning is to group observations within a dataset. Given a notion of similarity, finding those instances which are outstandingly similar to each other has manifold applications. Recommender systems and topic analysis in text data are examples which are most intuitive to grasp. The interpretation of the groups, called *clusters*, is facilitated if the assignment of samples is definite. Especially in high-dimensional data, denoting a degree to which an observation belongs to a specified cluster requires a subsequent processing of the model to filter the most important information. We argue that a good summary of the data provides hard decisions on the following question: how many groups are there, and which observations belong to which clusters? In this work, we contribute to the theoretical and practical background of clustering tasks, addressing one or both aspects of this question. Our overview of state-of-the-art clustering approaches details the challenges of our ambition to provide hard decisions. Based on this overview, we develop new methodologies for two branches of clustering: the one concerns the derivation of nonconvex clusters, known as *spectral clustering*; the other addresses the identification of *biclusters*, a set of samples together with similarity defining features, via Boolean matrix factorization. One of the main challenges in both considered settings is the robustness to noise. Assuming that the issue of robustness is controllable by means of theoretical insights, we have a closer look at those aspects of established clustering methods which lack a theoretical foundation. In the scope of Boolean matrix factorization, we propose a versatile framework for the optimization of matrix factorizations subject to binary constraints. Especially Boolean factorizations have been computed by intuitive methods so far, implementing greedy heuristics which lack quality guarantees of obtained solutions. In contrast, we propose to build upon recent advances in nonconvex optimization theory. This enables us to provide convergence guarantees to local optima of a relaxed objective, requiring only approximately binary factor matrices. By means of this new optimization scheme PAL-Tiling, we propose two approaches to automatically determine the number of clusters. The one is based on information theory, employing the minimum description length principle, and the other is a novel statistical approach, controlling the false discovery rate. The flexibility of our framework PAL-Tiling enables the optimization of novel factorization schemes. In a different context, where every data point belongs to a pre-defined class, a characterization of the classes may be obtained by Boolean factorizations. However, there are cases where this traditional factorization scheme is not sufficient. Therefore, we propose the integration of another factor matrix, reflecting class-specific differences *within* a cluster. Our theoretical considerations are complemented by empirical evaluations, showing how our methods combine theoretical soundness with practical advantages.

Contents

1 INTRODUCTION **1**
1.1 Roadmap 5

2 MATRIX FACTORIZATION WITH BINARY CONSTRAINTS **9**
2.1 Notation 9
2.2 Nonnegative Matrix Factorization 10
2.2.1 NMF and Clustering 11
2.2.2 Algorithms for NMF 12
2.3 One-Sided Clustering 14
2.3.1 k-Means 15
2.3.2 Kernel k-Means 17
2.3.3 Graph Cuts and Factorizing Graph Laplacians 18
2.4 Two-Sided Clustering 21
2.4.1 Checkerboard Clustering 22
2.4.2 Plaid Model 24
2.4.3 Block Diagonal Model and Bipartite Graph Cuts 26
2.4.4 Binary Matrix Factorization 27
2.4.5 Boolean Matrix Factorization 29

3 OPTIMIZING SUBJECT TO BINARY CONSTRAINTS **33**
3.1 Greedy Approach 33
3.2 Alternating Minimization 35
3.3 Orthogonal Nonnegative Relaxation 36
3.4 Spectral Relaxation 37
3.5 Nonnegative Relaxation with Nonbinary Penalization 39

4 A SPECTRAL APPROACH TO DENSITY BASED CLUSTERING **41**
4.1 Spectral Clustering and the Robustness Issue 43
4.2 Spectral Averagely-Dense Clustering 43
4.2.1 Dense Clusters and Projected Eigenvectors 44
4.3 Experiments 47

4.3.1 Experiments on Synthetic Data . . . 47
4.3.2 Real-World Data Experiments . . . 52
4.4 Discussion . . . 54

5 PROXIMAL OPTIMIZATION FOR BMF 55
5.1 Approximating the Boolean Product . . . 57
5.2 Proximal Alternating Linearized Minimization . . . 58
5.2.1 Convergence Results . . . 59
5.2.2 Kurdyka-Łojasiewicz Property . . . 61
5.3 Nonbinary Penalization within PALM . . . 63
5.3.1 A Binary Proximal Mapping . . . 63
5.3.2 The Proposed BMF Framework: PAL-Tiling . . . 65
5.4 Discussion . . . 68

6 BMF RANK SELECTION BY MDL 71
6.1 MDL Principle . . . 72
6.1.1 MDL for Pattern Mining . . . 72
6.1.2 MDL for BMF . . . 75
6.2 Minimizing the Description Length in PAL-Tiling . . . 76
6.2.1 Panpal . . . 77
6.2.2 Primp . . . 78
6.3 Experiments . . . 81
6.3.1 Experiments on Synthetic Data . . . 82
6.3.2 Real-World Data Experiments . . . 91
6.3.3 Qualitative Inspection of Mined Tiles . . . 93
6.4 Discussion . . . 97

7 BMF RANK SELECTION BY FDR 99
7.1 False Discoveries in BMF . . . 100
7.1.1 Tiles in Bernoulli Matrices . . . 101
7.1.2 Rejecting the Rejection of Null Hypotheses . . . 103
7.2 Theoretical Comparison of Proposed Bounds . . . 105
7.3 Algorithmic Integration of FDR Control . . . 109
7.4 Experiments . . . 110
7.4.1 Experiments on Synthetic Data . . . 110
7.4.2 MovieLens Experiments . . . 112
7.5 FDR Control in Clustering and Pattern Mining . . . 113
7.6 Discussion . . . 114

8 MINING CLASS-SPECIFIC ALTERATIONS WITH BMF **117**
8.1 Integrating Class Labels into BMF 119
8.2 Mining Class-Specific Alterations 120
8.2.1 C-Salt 121
8.3 Experiments 124
8.3.1 Experiments on Synthetic Data 125
8.3.2 Real-World Data Experiments 128
8.3.3 Genome Data Analysis 132
8.4 Discussion 133

9 CONCLUSIONS **135**
9.1 Impact and Future Directions 138

A PROOFS OF CHAPTER 2 **141**

B PROOFS OF CHAPTER 5 **143**

LIST OF FIGURES **154**

LIST OF ALGORITHMS **155**

BIBLIOGRAPHY **157**

Nomenclature

$A, B, C, \ldots$	Matrices
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	Random variables
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	Sets
$\mathbb{1}^{m \times n}$	Set of all binary $m \times n$ partition matrices; every row is a unit vector
$\mathbb{R}, \mathbb{R}_+$	The set of all real values, respectively the set of all nonnegative values
$A_{\mathcal{J}\mathcal{I}}$	Submatrix of A, containing only the rows in $\mathcal{J}$ and columns in $\mathcal{I}$
D	Data matrix
I	Identity matrix
I_A	The diagonal matrix $\operatorname{diag}(A\mathbf{1})$
$Y^\dagger$	Moore-Penrose inverse $\left(Y^\top Y\right)^{-1} Y^\top$
$\operatorname{diag}(x)$	Diagonal matrix whose diagonal corresponds to vector x
$\mathbf{1}, \mathbf{0}$	Constant one or zero matrix: $\mathbf{1}_{ji} = 1$, $\mathbf{0}_{ji} = 0$
$\overline{A}$	The Boolean complement matrix $\mathbf{1} - A$ for a binary matrix A
$\mathcal{N}_\epsilon(j)$	The set of all points lying in the ϵ-ball around point j
$\mu(A)$	The average of all elements in the matrix A
θ, θ_t	Heaviside step function, $\theta_t(x) = 1$ if $x > t$ and $\theta_t(x) = 0$ otherwise, $\theta = \theta_{0.5}$
$\lVert \cdot \rVert$	Entrywise 2-norm (Frobenius norm)
$\langle \cdot, \cdot \rangle$	The Frobenius inner product

$\lvert \cdot \rvert$	Entrywise 1-norm
$\circ$	Hadamard product
$\odot$	Boolean product

CHAPTER 1

Introduction

Making decisions is hard – generally in life, but for learning systems as well. Finding the optimal (sequence of) choices leading to the best outcome offers in most cases far too many possibilities. One may think of simple examples such as computing the optimal move in a chess game, deciding over the winner in a sports bet, or finding the optimal gift for a special someone. With respect to machine learning, these examples are prototype applications of binary decision making in the three main fields: chess for *reinforcement learning*; placing a bet for *supervised learning*; and giving recommendations for *unsupervised learning*. Across these fields, the requirement to make binary *yes* or *no* decisions is handled differently. Famous applications of reinforcement learning exploit the possibility to let a computer run simulations of its own, e.g., play a game against itself (Silver et al., 2017). In this variant, the vast discrete search space is traversed during idle times. A compressed model maintains the gained knowledge and is used to swiftly make decisions during a game. The compressed model is often based on a classification model. Classification is an instance of supervised learning. In this branch of machine learning, determining if an object belongs to one class or another is usually based on a relaxed model, e.g., reflecting the probabilities of possible outcomes. With regard to our example, a bet should be made on the side which is most likely to win. While reinforcement learning basically relies on knowledge gained by self-made experience (finding out the best strategy in a game by playing against itself again and again), clustering relies on the experience gathered by others. For instance, given the shopping history of a huge collection of costumers, a clustering model can be used to recommend a gift which is liked by specific costumers having a similar shopping history. Of those three examples, we focus on the latter in this work.

As the name suggests, what exactly we are aiming for is unclear in unsupervised learning. It comprises tasks where the goal is to explore and to find characteristics

and recurring patterns, setting some instances apart from others. We denote a set of observations which belongs together in some sense as a cluster. A set of clusters, the clustering, is supposed to represent the key concepts found in the data. Concept learning is an umbrella term for tasks which involve finding a set of objects which can usefully be grouped together (Morik et al., 1993). Being developed during the 1990s, concept learning has been formalized with respect to what was a hot topic back then: inductive logic programming (Muggleton, 1992). In this framework, a theory has been established around the *what* and *how* of hypothesis selection. In particular, the possibilities to control the complexity of learning representations have been explored (Kietz and Wrobel, 1992). What has been valid then has become even more crucial now, and thus, we take up some of the most important requirements for the definition of suitable clusterings.

COMPLETENESS AND NON-REDUCIBILITY Given a set of derived hypotheses (a model), *completeness* requires that all necessary information provided by the data is derivable from the model. *Non-reducibility* states that there is no redundancy among these hypotheses: there is no smaller set of hypotheses satisfying completeness. These requirements are similar to those defining the basis of a vector space. Given a set of data points, the subspace spanned by those data points has a set of vectors called the *basis*. Basis vectors are complete in the sense that every data point lies in the span of the basis vectors. Thus, we can reconstruct every data point as a linear combination of the basis vectors. A basis is furthermore non-reducible due to the linear independence of its vectors. The concept of linear independence implies that none of the basis vectors is a linear combination of the others and thus, every point in the spanned space is uniquely represented as a linear combination of basis vectors. We focus here on cluster models which approximately implement the constraints of completeness and non-reducibility. Completeness is not a strong requirement for cluster algorithms because an aggregation of data to groups does not need to pertain all the information provided in the data. Errors in measurements or, generally, noisy information which is not relevant for the task at hand should be filtered out. Therefore, we aim at finding a lower-dimensional subspace than the space spanned by all data points, maintaining the most characteristic traits of the data. This mechanism is incorporated by matrix factorizations, which we employ as a standard reference framework. It approximates a given data matrix by the product of two matrices. One matrix represents the basis vectors of the derived subspace and the other matrix returns the coefficients of the linear combination, derived by a projection of the data points onto the derived subspace. In that sense, matrix factorization

derives complete and non-reducible cluster models, as it finds a set of basis vectors of a (low-dimensional) subspace, approximating the space spanned by all data points.

INTERPRETABILITY Especially in explorative data mining, where the model is supposed to provide insight into the data, we need to be able to interpret the information encoded in the model. In inductive logic programming, *interpretability* is achieved by means of concepts, which have some similarities with clusters as they group terms according to a subsumption relation and relations between the terms (Kietz and Morik, 1994). This addresses one aspect of interpretability, which is the reason why we require that assignments of points to clusters are binary. There are plausible reasons to consider fuzzy memberships or probabilistic cluster assignments as well. The most prominent example of such methods is probably latent Dirichlet allocation (Blei et al., 2003). However, understanding the implications of such models typically requires a post-processing step and doesn't come naturally. As an example, how are we supposed to interpret a clustering stating that a single instance belongs with probability 0.4 to cluster one and with probability 0.6 to cluster two? Let us recall our gift recommendation example. What could we learn from such an assignment? Should we buy a gift from cluster one or cluster two? Would maybe none of the clusters provide suitable gifts or both? We argue that a more helpful cluster indication provides the user with clear information, recommending to buy a gift from one of the clusters, both or none: we should be making hard decisions.

EXPLAINABILITY The inductive logic programming algorithm MYCIN is one of the prototypes of *explainable* artificial intelligence methods (Fagan et al., 1980). Given a set of hand-coded rules, this algorithm lays out its reasoning together with the suggested diagnosis. The inductive logic programming system MOBAL shows not only the derivation from rules and facts, but also the induction from facts (Morik et al., 1993). The demand for the ability to provide explanations for particular results acquired by an algorithm, gained momentum since the European Union urged the disclosure of any automated decision-making, made on a solely algorithmic basis since 2018 (cf. Doshi-Velez and Kim (2017) and references therein). Explainability is related to and often also referred to as interpretability. We explicitly distinguish between these terms here and refer with explainability to the transparency of the method, whereas interpretability concerns the way information is presented to the practitioner. In short, while interpretability refers to understanding *what* is returned by an algorithm, explainability refers to understanding *why* the result is derived. We distinguish two ways to address the explainability of clustering models. The one is to provide characteristics of (local)

optima of the objective function. Those characteristics could be necessary conditions which are formulated in an understandable way, e.g., as a rule. The other concerns a reasoning for the performed model selection. Model selection refers here particularly to the derived number of clusters. If we can explain under what circumstances the number of derived clusters is suitable, trust towards algorithmic based decisions can be established in an unsupervised setting where no direct feedback is provided.

PREFERENCE BIAS Bias in inductive logic programming generally addresses the influence a specific learning method has on the result. *Preference bias* more specifically hints at the bias emerging from the way the search space is traversed: a bias which occurs among other things when a method focuses on some hypotheses more than others or when there are possibilities to prune some of the hypotheses. This is a relevant aspect for clustering algorithms, since clustering objectives are by and large nonconvex and have multiple local optima. Therefore, the applied optimization procedure has a big impact on the result. This is a general issue in optimization: there is a relatively new field emerging around the question how a particular setting of learning parameters in gradient descent procedures influences the probability to end up in local optima of a certain kind (Hennig and Kiefel, 2012; Hennig, 2015). Although these efforts pose only the beginning in creating a characterization of results generated with numerical optimization methods, it also points out a long-term advantage of algorithms which follow generic optimization methods. Insights in optimization theory are transferable to particular instances of the optimization methods, enabling the development of even more robust and suitable algorithms by incorporating newer findings, e.g., adding noise to the gradient (Jin et al., 2017; Hennig, 2013).

Another aspect of bias addresses the problem of approximation in hard clustering tasks. Most of the popular clustering problems are not only NP-hard but also NP-hard to approximate within a given factor. As a result, efficient methods which are supposed to approximate solutions of the given problem return for some example data sets solutions which diverge from actual optima of the objective (unless NP=P). Since a guaranteed approximation of the objective is not possible under the assumption that NP$\neq$P, the question is how to suitably relax the objective. Theoretical soundness and insight into solutions of the relaxing methods are therefore relevant. Whenever possible, the search space of the relaxed objective should be suitably restricted such that the learner is pointed towards directions where suitable optima are more likely to be found.

An important factor determining the suitability of clustering models is the number of expected clusters. We need to find a trade-off between the desired low dimensionality of the subspace and its approximation to the space spanned

by the data. If the model summarizes the data too coarsely, then only little if any knowledge can be gained. On the other hand, if the model does not generalize enough, then we run the risk of reflecting structure where there is none. A common way around this problem is to let the user decide the number of clusters. However, this stands in contrast to the explorative setting, where the user often knows little about the key characteristics of the data, not to mention the number of clusters. Therefore, finding ways to automatically determine the number of clusters is important.

The requirements of completeness, non-reducibility and interpretability determine the overall framework of this thesis: matrix factorizations with binary constraints. Within this framework, we propose clustering methods which are explainable and for which we discuss and study the circumstances influencing the preference bias. Popular instances of matrix factorization encompass Principal Component Analysis (PCA), Singular Value Decomposition (SVD) and eigendecomposition. We will see that a whole branch of clustering, probably the most widely known and used, has an objective related to either of the popular eigendecompositions subject to binary constraints. Furthermore, these objectives have applications in fields which are not directly related to clustering, such as hashing and combinatorial optimization problems (Ding et al., 2008; Mukherjee et al., 2015). Not only in this respect, being able to make hard decisions in the standard framework of matrix factorizations and the standard method of optimization has applications far outside the scope of clustering. We take a first step towards creating a mathematical theory of making hard decisions.

1.1 Roadmap

BACKGROUND The background material is covered in Chapters 2 and 3. In Chapter 2, we discuss the broad spectrum of clustering tasks where the approximation error is minimized according to the Frobenius norm subject to binary constraints. We unveil under which circumstances this optimization task defines the objectives of k-means, spectral clustering and various subspace clustering objectives. A unified formalism of these objectives makes similarities and differences between the tasks apparent, obtaining a taxonomy of hard clustering optimization problems. In Chapter 3, we review existing optimization approaches, generic to objective characteristics. Considering what approaches exist for the optimization of discussed algorithms, we distinguish between the relaxations relying either on singular value decomposition, eigendecomposition and orthonormal nonnegative decompositions, and optimizations relying on a greedy approach, alternating min-

imization and a nonbinary penalization. Each of these approaches corresponds to one prototype clustering. We discuss in detail this particular clustering instance and how this approach is applicable to related problems.

THE SPECTACL OF NONCONVEX CLUSTERING When it comes to clustering nonconvex shapes, two paradigms are used to find the most suitable clustering: minimum cut and maximum density. The most popular algorithms incorporating these paradigms are spectral clustering and DBSCAN. Both paradigms have their pros and cons. While minimum cut clusterings are sensitive to noise, density-based clusterings have trouble handling clusters with varying densities. Furthermore, spectral clustering involves a peculiar discretization step based on k-means clustering, a choice which apparently works well in practice but which lacks a theoretical or intuitive explanation. In contrast, DBSCAN is very sensitive to its parameter setting and the related preference bias is not well understood. In Chapter 4, we propose SpectACl (Hess et al., 2019): a method combining the advantages of both approaches, while solving the mentioned drawbacks. Therewith, we also pose a step toward a unified (nonconvex) clustering formalism, contributing to the discussion about similarities between DBSCAN and spectral clustering (Schubert et al., 2018). Our method is easy to implement, similar to spectral clustering. However, unlike spectral clustering, we demonstrate the fundamental soundness of applying k-means for discretization in SpectACl: we show that the application of k-means in SpectACl results in an optimization of an upper bound of the objective function. Through experiments on synthetic and real-world data, we demonstrate that our approach provides robust and reliable clusterings.

PROXIMAL ALTERNATING MINIMIZATION FOR BMF From Chapter 5 on, we focus on the clustering of binary data by Boolean matrix factorization. Boolean matrix factorization decomposes the data matrix into two binary factor matrices, whose product is performed in Boolean algebra. The restriction to only binary factor matrices enables the interpretation of cluster descriptions by binary vectors. Choosing the matrix multiplication to be performed in Boolean algebra enables a maximally free choice of cluster assignments. Since one plus one equals one in Boolean algebra, the optimization with respect to Boolean operations enables the derivation of nonexhaustive, overlapping clusters. However, the optimization of Boolean matrix factorizations is particularly difficult due to the nonlinearity of the matrix product and the explicit representation of overlapping clusters. To this end, only heuristic and greedy approaches have been proposed, resulting in an undesired preference bias where the first cluster is prone to cover most of the data already. In Chapter 5 we propose a generic optimization procedure based on latest results in nonconvex and nonsmooth optimization (Hess et al., 2017). Our main

contribution is the derivation of a prox operator which embodies the penalization of nonbinary factor matrices. We provide therewith a universal solution to the matrix factorization problem with respect to binary constraints. Piatkowski (2018) recently extended this optimization scheme from binary to integer constraints in order to estimate integer exponential families.

RANK SELECTION BY MDL The parameter which influences the result most in Boolean matrix factorization is the number of expected clusters. On toy datasets, even the greedy approaches usually work sufficiently well if the number of clusters is correctly set. In Chapter 6, we explore the direct minimization of description lengths of the resulting factorization. This approach is well known for model selection and data compression, but not for finding suitable factorizations via numerical optimization. We demonstrate the superior robustness of the new approach in the presence of several kinds of noise and types of underlying structure. Moreover, our general framework can work with any cost measure having a suitable real-valued relaxation. Thereby, no convexity assumptions have to be met. The experimental results on synthetic data and image data show that the new method identifies interpretable patterns which explain the data almost always better than the competing algorithms. The work discussed in this chapter has been cited as a unifying approach, bringing together the fields of low-rank models and pattern mining (Pfahler et al., 2017; Ramírez, 2018; Holzinger et al., 2017).

RANK SELECTION BY FDR Model selection by means of the minimum description length is information-theoretically founded and can be used to deliver empirically satisfactory results. Yet, there are no guarantees that any of the returned clusters do not actually arise from noise, i.e., are false discoveries. In Chapter 7, we propose and discuss the usage of the false discovery rate in the unsupervised Boolean matrix factorization setting (Hess et al., 2018). We prove two bounds on the probability that a found cluster is constituted of random Bernoulli-distributed noise. Each bound exploits a specific property of the factorization which minimizes the approximation error—yielding new insights on the minimizers of Boolean matrix factorization. This does not only contribute to the explainability of Boolean matrix factorization results, but also leads to improved algorithms by replacing heuristic rank selection techniques with a theoretically well-based approach. Our empirical demonstration shows that both bounds deliver excellent results in various practical settings.

MINING CLASS-SPECIFIC ALTERATIONS IN BINARY DATA In Chapter 8, we take a peek into applications of interpretable Boolean matrix factorizations in a supervised setting. 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 clusters which are either specific to one or common over the classes, we propose the concept of class-specific alterations (Hess and Morik, 2017). Therewith, we are able to derive clusters which are common to all classes together with its class-related deviations. Opposed to general classification methods, we are less interested in prediction than in explainability of the results. The model is designed to derive the properties which create the uniformity of a group and which discriminate among the classes within a group. Therewith, we broaden the applicability of our proposed method C-Salt 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 a real-world application.

CONCLUSION We conclude our findings in Chapter 9 and discuss further directions. We discuss recent advances in nonconvex optimization theory, having applications within our proximal minimization framework for BMF. Furthermore, we discuss how this optimization scheme is possibly applied to other matrix factorizations subject to binary constraints. At least, we discuss possible adaptations to automatically determine the rank, transferring our results from the Boolean to the real-valued setting.

CHAPTER 2

Matrix Factorization with Binary Constraints

Cholesky decomposition, principal component analysis and eigendecomposition – those are famous examples involving matrix factorization, which most machine learners and data miners have probably used at least once in their life. For practical applications, it is sufficient to know that these factorizations compute a certain kind of embedding into a subspace where the information relevant to the task at hand is maintained. There are built-in functions computing the required decomposition in an efficient manner for all main programming languages. A little bit less known are the general optimization problems whose solutions return the desired decompositions. Even less known is the fact that these optimization problems are transformed into the most widely used clustering objectives by a restriction to binary values of one or more factor matrices.

2.1 Notation

A formal discussion of the state of clustering requires some notation. We visually distinguish between sets, matrices and vectors by displaying sets in calligraphy style $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots$, matrices in uppercase mode $X, Y, Z, \ldots$ and vectors and constants as lowercase letters x, y, z, where we highlight vectors in bold if we want to avoid confusion with constants. We use $\mathbb{R}$ and $\mathbb{R}_+$ to denote the set of real and nonnegative real values. Accordingly, $\mathbb{N}$ denotes the set of natural numbers. Furthermore, we state the set of all $n \times m$ partition matrices as $\mathbb{1}^{m \times n}$. That is, $A \in \mathbb{1}^{m \times n}$ if and only if A is a binary matrix and every row of A has exactly one entry equal to one.

We write $\mathbf{1}$, respectively $\mathbf{0}$, to represent a matrix having all elements equal to 1, respectively 0. The dimension of such a matrix can always be inferred from the context. We denote with the matrix I the identity matrix. We often neglect stating the range of indices if that is clear from the context. We strive for uniformity in the use of indices, such that the range is usually deducible from the particular index. As such, we usually have $1 \leq i \leq n, 1 \leq j \leq m$ and $1 \leq s \leq r$.

We assume that our data is given by the matrix $D \in \mathbb{R}^{m \times n}$. The data represents m points $D_{j\cdot}$ for $1 \leq j \leq m$ in the n-dimensional feature space. For every point, we denote with $\mathcal{N}_\epsilon(j) = \{l | \|D_{j\cdot} - D_{l\cdot}\| < \epsilon\}$ its ϵ-neighborhood.

Throughout this work, we often employ the Heaviside step function θ_t which rounds real to binary values, i.e., $\theta_t(x) = 1$ for $x > t$ and $\theta_t(x) = 0$ otherwise. We abbreviate $\theta_{0.5}$ to θ and denote with $\theta(X) = (\theta(X_{ji}))_{ji}$ the entrywise application of θ to a matrix X. The operator $\circ$ denotes the Hadamard product which multiplies two matrices of same dimensionality elementwise. We denote matrix norms as $\|\cdot\|$ for the Frobenius norm and $|\cdot|$ for the entrywise 1-norm. These norms are equivalent for binary matrices A in the sense that $|A| = \|A\|^2$. The Frobenius inner product is defined for matrices X and Y as $\langle X, Y \rangle_F = \mathrm{tr}(X^\top Y)$. For nonnegative matrices X and Y, the Frobenius inner product equates the entrywise 1-norm of the (componentwise) Hadamard product (denoted by $\circ$), i.e., $\langle X, Y \rangle_F = |X \circ Y|$. Lastly, we remark that log denotes the natural logarithm.

2.2 Nonnegative Matrix Factorization

The objective of Nonnegative Matrix Factorization (NMF) makes only a slight alteration to renowned factorizations known as Principal Component Analysis (PCA) or truncated Singular Value Decomposition (SVD), by requiring that the factor matrices are nonnegative. Unfortunately, this seemingly small modification makes NMF an NP-hard problem (Vavasis, 2009), opposed to the polynomially solvable SVD. NMF is originally introduced by Paatero and Tapper (1994) under the name positive matrix factorization. It gained much attention since the publications from Lee and Seung, showing that the nonnegativity constraints and the resulting parts-based explanation of the data empowers the interpretability of the results. Since then, efficient computations of good approximations to the NMF problem are studied.

A formal task definition of NMF is given as follows: let $D \in \mathbb{R}_+^{m \times n}$ be the nonnegative data matrix and let r be a specified integer, we refer to as rank. The task of NMF is then to recover nonnegative factors $X \in \mathbb{R}_+^{n \times r}$ and $Y \in \mathbb{R}_+^{m \times r}$ such

that $YX^\top$ approximates D. The quality of the approximation is usually measured by means of the Frobenius norm:

$$\min_{X,Y} F(X,Y) = \frac{1}{2}\left\|D - YX^\top\right\|^2 \quad \text{s.t. } X \in \mathbb{R}_+^{n\times r}, Y \in \mathbb{R}_+^{m\times r}. \tag{NMF}$$

We refer in the following to the function $F(X,Y) = \frac{1}{2}\left\|D - YX^\top\right\|^2$ also as the Residual Sum of Squares (RSS). Another popular measurement of the approximation error is the Kullback-Leibler divergence. While the former approach (employing the Frobenius norm) is inherently related to clustering using the Euclidean distance of points, the latter is related to topic models such as probabilitistic latent semantic indexing (Gaussier and Goutte, 2005; Ding et al., 2006a). We will later discuss the particular relationships of approximations with respect to the Frobenius norm and related clustering objectives.

2.2.1 NMF and Clustering

Although initially the difference between NMF and clustering was emphasized (Lee and Seung, 1999), further research affirms inherent clustering properties (Li and Ding, 2006). In this context, columns of X equate cluster centroids and corresponding columns of Y indicate cluster membership tendencies. This view is supported by the interpretation of a factorization as a decomposition into basis vectors and their coefficients. The approximation of data point j is given by

$$D_{j\cdot} \approx Y_{j\cdot}X^\top = \sum_{s=1}^{r} Y_{js}X_{\cdot s}^\top,$$

a linear combination of the column vectors of X, whose coefficients are given by $Y_{j\cdot}$. The interpretation of this decomposition generally depends on the basis vectors, and their meaning is more easy to grasp if they reflect an archetype of the induced latent space. In this respect, constraining the rows of the coefficient matrix to probabilistic vectors ($|Y_{j\cdot}| = 1$) has proven valuable. This constraint implies that every data point is a convex combination of the basis vectors, which in turn induces a preference bias on the basis vectors to extreme data points instead of holistic representations (Thurau et al., 2012).

Another important aspect of NMF, which is important in the scope of cluster applications, is the near orthogonality of solutions. That is, the columns of the factor matrices Y and X are approximately orthogonal, implying, e.g., for Y, that the inner product $\langle Y_{\cdot s}, Y_{\cdot t}\rangle$ is close to zero for $s \neq t$. The reason why NMF produces near orthogonal factor matrices is due to its objective function. Every matrix product $YX^\top$ of rank r is computable as the sum of r outer products or

rank-1 matrices $Y_{\cdot s}X_{\cdot s}^\top$. Thereby, the approximation error returned by the function F is transformable into the sum

$$\begin{aligned}
&\Big\|D - \sum_{s=1}^{r} Y_{\cdot s}X_{\cdot s}^\top\Big\|^2 \\
&= \|D\|^2 - 2\Big\langle D, \sum_s Y_{\cdot s}X_{\cdot s}^\top\Big\rangle + \Big\langle \sum_s Y_{\cdot s}X_{\cdot s}^\top, \sum_s Y_{\cdot s}X_{\cdot s}^\top\Big\rangle \\
&= \|D\|^2 - 2\sum_s \Big\langle D, Y_{\cdot s}X_{\cdot s}^\top\Big\rangle + \sum_s \big\|Y_{\cdot s}X_{\cdot s}^\top\big\|^2 + \sum_{s\neq t}\Big\langle Y_{\cdot s}X_{\cdot s}^\top, Y_{\cdot t}X_{\cdot t}^\top\Big\rangle.
\end{aligned}$$

Here, $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product. We add $(r-1)\|D\|^2$ to the equation above, which does not affect the solution of the minimization with respect to X and Y, and obtain therewith a minimization problem which is equivalent to (NMF)

$$\min_{Y,X} \sum_{s=1}^{r}\left(\big\|D - Y_{\cdot s}X_{\cdot s}^\top\big\|^2 + \sum_{t\neq s}\Big\langle Y_{\cdot s}X_{\cdot s}^\top, Y_{\cdot t}X_{\cdot t}^\top\Big\rangle\right). \tag{2.1}$$

This formulation of the NMF objective shows that suitable factorizations are the sum of outer products which approximate the data matrix well while having as little overlap as possible with other outer products. The rightmost term $\langle Y_{\cdot s}X_{\cdot s}, Y_{\cdot t}X_{\cdot t}\rangle = (X_{\cdot s}^\top X_{\cdot t})(Y_{\cdot s}^\top Y_{\cdot t})$ vanishes if $X_{\cdot s}$ is orthogonal to $X_{\cdot t}$ or $Y_{\cdot s}$ to $Y_{\cdot t}$. Due to the nonnegativity of the matrices, this is the case if nonzero entries from two distinct columns of a factor matrix do not overlap.

The two properties to obtain convex combinations and orthogonality are satisfied when the coefficient matrix $Y \in \mathbb{1}^{m\times r}$ is a partitioning matrix. This is a central constraint for most clustering applications.

2.2.2 Algorithms for NMF

There are numerous approaches to solve the optimization problem of NMF in its various versions, employing regularizations and constraints. Here, we focus on three methods, which are particularly suited to incorporate binary on some of the factor matrices. For a more thorough overview and discussion of optimization aspects, we refer the reader to Cichocki et al. (2009) and Kim et al. (2014).

ALTERNATING MINIMIZATION The function F in objective (NMF) is nonconvex, but convex in either X or Y, if the other argument is fixed. This property ensures that the *Gauss-Seidel* scheme, also known as *block-coordinate descent*, an *alternating minimization* along one of the matrices while the other one is fixed,

returns a stationary point upon convergence. Grippo and Sciandrone (2000) have shown that the nonnegativity constraint ensures the convergence of the sequence (X_k, Y_k) generated by the update rule

$$X_{k+1} \in \arg\min_X F(X, Y_k) \tag{2.2}$$

$$Y_{k+1} \in \arg\min_Y F(X_{k+1}, Y). \tag{2.3}$$

Decisive for the applicability of this method is the computational effort required to solve the subproblems in every iteration. Since there is no analytical solution known for problem (NMF) when one of the factor matrices is fixed, solving one of the subproblems breaks down to finding a solution by numerical optimization in every iteration. Hence, practical applications of the Gauss-Seidel scheme only approximate the solution to Eqs. (2.2) and (2.3) (Wang and Zhang, 2013). Often, the minimization step is replaced by a single gradient descent update. In that respect, the optimization procedures, which we discuss in the following, are all lazy implementations of the alternating minimization.

MULTIPLICATIVE UPDATES NMF received much attention since the publication of the easily implementable multiplicative update algorithm by Lee and Seung (2001). The update rules are defined as elementwise multiplications, given by

$$X_{is} \leftarrow X_{is} \frac{D_{\cdot i}^\top Y_{\cdot s}}{X_{i\cdot} Y^\top Y_{\cdot s}} \tag{2.4}$$

$$Y_{js} \leftarrow Y_{js} \frac{D_{j\cdot} X_{\cdot s}}{Y_{j\cdot} X^\top X_{\cdot s}}. \tag{2.5}$$

The nonnegativity of the factor matrices is ensured during the optimization procedure since only nonnegative elements are multiplied. Yet, the above update rules are also transformable into gradient descent steps, where the stepsize is always small enough such that the nonnegativity of the factor matrices is preserved.

One of the major advantages of multiplicative updates is that the integration of some constraints, such as nonnegativity or orthogonality of factor matrices, is straightforward in this scheme. The convergence to local extremal points of the objective is in most cases easy to derive from the Karush-Kuhn-Tucker conditions. Unsurprisingly, there are many proposals to solve matrix factorizations with binary constraints by multiplicative updates.

The major drawback of multiplicative updates is the conservative choice of the stepsize, which has to be small enough such that a step into a descent direction does not leave the feasible set. This results in a very slow convergence rate and makes the approach suitable only for smaller datasets. In addition, from Eqs. (2.4) and (2.5) follows that every entry in the factor matrices becoming zero is going to stay zero until the end of the optimization. This inflexibility of the optimization scheme makes solutions likely to converge to less optimal minima, since mistakes can not be corrected.

PROXIMAL MINIMIZATION AND PROJECTED GRADIENT Instead of restricting the stepsize to persistently satisfy the nonnegativity constraint, larger stepsizes can be employed and possibly negative entries can be projected to the positive orthant. Using larger stepsizes improves the convergence rate, but the stepsize should not be too large, otherwise the iterates might be zig-zagging around the optimum. We can always fall back upon using linesearch such as the Armijo rule (Lin, 2007), but having the possibility to calculate suitable stepsizes according to a specified strategy is desirable in practice.

Assuming that we have a good strategy to efficiently determine the stepsizes α_k and β_k, the projected gradient procedure for nonnegative matrix factorization performs the following updates:

$$\begin{aligned} X_{k+1} &\leftarrow X_k - \alpha_k \nabla_X F(X_k, Y_k) \\ X_{k+1} &\leftarrow \theta_0(X_{k+1}) \circ X_{k+1} \\ Y_{k+1} &\leftarrow Y_k - \beta_k \nabla_Y F(X_{k+1}, Y_k)) \\ Y_{k+1} &\leftarrow \theta_0(Y_{k+1}) \circ Y_{k+1}. \end{aligned}$$

We employ here the Heaviside step function to perform the projection step, the operation $\theta_0(X) \circ X$ sets all nonnegative entries to zero. Projected gradient procedures work well in practice, the crucial aspect is however the determination of the stepsize in order to ensure convergence. There has been little theory for this optimization scheme until proximal methods have been researched for nonconvex problems Bolte et al. (2014). We will discuss this theory of proximal methods in detail in Chapter 5.

2.3 One-Sided Clustering

In a nutshell, hard clustering aims at grouping data points according to a notion of similarity. A fundamental concept of clustering is to find a trade-off between

TABLE 2.1: The matrix factorization objectives of one-sided clustering.

Minimize $\left\|A - ZZ^\top\right\|^2$ or maximize $\mathrm{tr}\left(Z^\top AZ\right)$, subject to $Y \in \mathbb{1}^{m\times r}$ and		
k-Means	$Z = Y\left(Y^\top Y\right)^{-1/2}$	$A = DD^\top$
Kernel k-Means	$Z = Y\left(Y^\top Y\right)^{-1/2}$	$A = K$
Normalized Cut	$Z = I_W^{1/2} Y\left(Y^\top I_W Y\right)^{-1/2}$	$A = -L_s$ or $A = I_W^{-1/2} W I_W^{-1/2}$
Ratio Cut	$Z = Y(Y^\top Y)^{-1/2}$	$A = W - I_W = -L_d$

intra-cluster similarity and inter-cluster distance; to minimize distances of points within a cluster while maximizing distances of points from distinct clusters. The definition of the term distance is deciding for the resulting clustering task.

2.3.1 k-Means

If there is one algorithm which comes to mind when thinking about clustering, it is likely the k-means algorithm (Lloyd, 1982). The objective of k-means is founded in the intuitive notion of clusters by the within-cluster point scatter, minimizing the intra-cluster similarity. Suppose we are given a partition of m points reflected by clusters $\mathcal{J}_1, \ldots, \mathcal{J}_r$. A cluster is here the set $\mathcal{J}_s \subseteq \{1, \ldots, m\}$ of its point indices. The sum of average distances of points within a cluster is then given as

$$\begin{aligned}\frac{1}{2}\sum_{s=1}^{r}\frac{1}{|\mathcal{J}_s|}\sum_{j,l\in\mathcal{J}_s}\|D_{j\cdot} - D_{l\cdot}\|^2 &= \sum_{s=1}^{r}\frac{1}{|\mathcal{J}_s|}\left(\sum_{j\in\mathcal{J}_s}\|D_{j\cdot}\|^2|\mathcal{J}_s| - \sum_{j,l\in\mathcal{J}_s}\langle D_{j\cdot}, D_{l\cdot}\rangle\right). \\ &= \sum_{s=1}^{r}\sum_{j\in\mathcal{J}_s}\|D_{j\cdot}\|^2 - \sum_{j\in\mathcal{J}_s}\left\langle D_{j\cdot}, \frac{1}{|\mathcal{J}_s|}\sum_{l\in\mathcal{J}_s} D_{l\cdot}\right\rangle\end{aligned}$$

We define the matrix $X \in \mathbb{R}^{n\times r}$ by setting the columns $X_{\cdot s} = \frac{1}{|\mathcal{J}_s|}\sum_{l\in\mathcal{J}_s} D_{l\cdot}$ to the centroid of all points in cluster $\mathcal{J}_s$. Therewith, we continue the transformation to

$$\begin{aligned}\frac{1}{2}\sum_{s=1}^{r}\frac{1}{|\mathcal{J}_s|}\sum_{j,l\in\mathcal{J}_s}\|D_{j\cdot} - D_{l\cdot}\|^2 &= \sum_{s=1}^{r}\sum_{j\in\mathcal{J}_s}\left(\|D_{j\cdot}\|^2 - 2\langle D_{j\cdot}, X_{\cdot s}\rangle + \|X_{\cdot s}\|^2\right) \qquad (2.6) \\ &= \sum_{s=1}^{r}\sum_{j\in\mathcal{J}_s}\left\|D_{j\cdot} - X_{\cdot s}^\top\right\|^2.\end{aligned}$$

The equation above is the starting point from which various relationships between clustering and matrix factorization under binary constraints follows. The left term in Eq. (2.6) is the mentioned within cluster point scatter and it is easy to show that a minimization of this term goes along with the maximization of the average distance of points from distinct clusters (Friedman et al., 2001).

We now introduce the binary matrix $Y \in \mathbb{1}^{m\times r}$ to indicate the cluster partition given by the sets $\mathcal{J}_s$. That is, $Y_{js} = 1$ if point $j \in \mathcal{J}_s$ and $Y_{js} = 0$ otherwise. The objective of k-means is then to find the partition matrix Y minimizing

$$\min_{Y} \sum_{s=1}^{r}\sum_{j=1}^{m} Y_{js}\big\|D_{j\cdot} - X_{\cdot s}^{\top}\big\|^2 \qquad \text{s.t.} \quad Y \in \mathbb{1}^{m\times r},\ X_{\cdot s} = \frac{1}{|Y_{\cdot s}|}Y_{\cdot s}^{\top}D. \tag{KM}$$

Problem (KM) is transferrable into a constrained nonnegative matrix factorization problem and is likewise NP-hard (Aloise et al., 2009). Since there is for every point j exactly one cluster s such that $Y_{js} = 1$, we can pull the outer sum into the norm, that is

$$\begin{aligned}\sum_{s=1}^{r}\sum_{j=1}^{m} Y_{js}\big\|D_{j\cdot} - X_{\cdot s}^{\top}\big\|^2 = \sum_{j=1}^{m}\Big\|D_{j\cdot} - \sum_{s=1}^{r} Y_{js}X_{\cdot s}^{\top}\Big\|^2 &= \sum_{j=1}^{m}\big\|D_{j\cdot} - Y_{j\cdot}X^{\top}\big\|^2 \\ &= \big\|D - YX^{\top}\big\|^2.\end{aligned}$$

The cluster centers are in matrix notation given by $X = D^{\top}Y\left(Y^{\top}Y\right)^{-1}$. The matrix $Y^{\top}Y = \operatorname{diag}(|Y_{\cdot 1}|, \ldots, |Y_{\cdot r}|)$ is diagonal, since Y has orthogonal columns. Its inverse is easily computed by inverting the elements on the diagonal. Note, that the matrix $\left(Y^{\top}Y\right)^{-1}Y^{\top} = Y^{\dagger}$ is the Moore-Penrose inverse of the matrix Y.

The clustering of k-means has multiple equivalent formulations, which we summarize in the following theorem, whose formal proof is provided in Appendix A.

THEOREM 2.1. *The following optimization problems are equivalent to objective* (KM)

$$\min_{Y} \big\|D - YX^{\top}\big\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r}, X = D^{\top}Y\left(Y^{\top}Y\right)^{-1} \tag{2.7}$$

$$\min_{Y} \big\|D - YY^{\dagger}D\big\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r} \tag{2.8}$$

$$\min_{Y,X} \big\|D - YX^{\top}\big\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r}, X \in \mathbb{R}^{n\times r} \tag{2.9}$$

$$\max_{Y} \operatorname{tr}\left(Z^{\top}DD^{\top}Z\right) \qquad s.t.\ Z = Y\left(Y^{\top}Y\right)^{-1/2}, Y \in \mathbb{1}^{m\times r} \tag{2.10}$$

$$\min_{Y} \big\|DD^{\top} - YY^{\dagger}\big\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r} \tag{2.11}$$

The various formulations of the objective of k-means give rise to multiple optimization approaches. The equivalence of Eqs. (2.7) and (2.9) establishes an alternating minimization scheme, since the optimal cluster center matrix is provided in closed from, given the cluster assignment matrix. We will discuss this procedure, known als Lloyds' minimization, more in detail in Section 2.2.2. Since Y is orthogonal and nonnegative, Eq. (2.9) is approximable by an orthogonal relaxation (cf. Section 3.3). The objective in Eq. (2.10) resembles the optimization task to find the largest eigenvectors of a symmetric real-valued matrix such as $DD^\top$ (cf. Section 3.4). This relation founds the application of a spectral relaxation (Zha et al., 2002). Eqs. (2.10) and (2.11) show that k-means can be formulated in sole dependence of the similarity between points, measured by the inner product which equates the cosine similarity between points when the data points are normalized. This is a stepping stone to the application of kernel methods, discussed in the following section.

The restriction not only to a binary but a partition matrix Y in k-means clustering has many favorable outcomes concerning its optimization. An efficient alternating minimization scheme for the more general case where Y is a binary matrix is not known. However, some applications require more flexible cluster models, allowing for overlap between clusters and outlier detection. Examples for such areas are text mining and gene analysis. A single gene is typically involved in multiple functions of an organism and hence, it should be assignable to multiple clusters. A similar argument holds for documents, addressing more than one topic. Whang et al. (2018) propose a semidefinite program to allow for a specified amount of overlap and a specified amound of outliers. Unfortunately, the resulting semidefinite program has a large amount of constraints and its optimization is quite slow. Slawski et al. (2013) study properties of exact decompositions of the form $D = YX^\top$ where $Y \in \{0,1\}^{m\times r}$ and $X \in \mathbb{R}^{n\times r}$. Based on the observation that at most 2^r binary vectors lie in the subspace spanned by the columns of D, a combinatorial algorithm is proposed whose complexity is exponential in the rank. This algorithm is extended to find approximate factorizations $D \approx YX^\top$, yet the results display a high variance.

2.3.2 Kernel k-Means

The formulation of the k-means objective in sole dependence on the similarities of data points, expressed by the inner product, enables the application of kernel methods and the derivation of nonconvex clusters. One of the drawbacks of k-means clustering is that it computes a Voronoi tesselation which determines the

cluster membership. That is, the allocated regions of adjacent clusters are separated by a line. This entails that clusters which are returned by k-means are always convex.

If the cluster regions are not linearly separable, then a transformation into a suitable, usually higher-dimensional space enables a correct identification of clusters by k-means. Let $\Phi : \mathbb{R}^n \to \mathcal{H}$ be such a transformation from the feature space to a (possibly infinite-dimensional) Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. The similarities between two points, which are reflected by the symmetric matrix $DD^\top$ in Eqs. (2.10) and (2.11) are given in the transformed space by the kernel matrix $K \in \mathbb{R}^{m \times m}$, where

$$K_{jl} = \left\langle \Phi\left(D_{j\cdot}^\top\right), \Phi\left(D_{l\cdot}^\top\right) \right\rangle_{\mathcal{H}}. \tag{2.12}$$

Since the features are only transformed within the inner product, we can employ the *kernel trick* by computing the inner product in the transformed space directly. This application has been proposed by Schölkopf et al. (1998) in the more general scope of truncated SVD or PCA. Ding et al. (2005) discuss the application of kernels in the scope of k-means. We conclude the following more generally formulated corollary from Theorem 2.1, stated for symmetric matrices.

COROLLARY 2.2. *Let $K \in \mathbb{R}^{m \times m}$ be a symmetric matrix and let $K = UU^\top$ be a symmetric decomposition of K with $U \in \mathbb{R}^{m \times m}$. The following optimization problems are equivalent:*

$$\max_Y \ \operatorname{tr}\left(Z^\top K Z\right) \qquad s.t.\ Z = Y\left(Y^\top Y\right)^{-1/2}, Y \in \mathbb{1}^{m \times r} \tag{2.13}$$

$$\min_Y \ \left\| K - YY^\dagger \right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m \times r} \tag{2.14}$$

$$\min_{Y,X} \ \left\| U - YX^\top \right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m \times r}, X \in \mathbb{R}^{m \times r} \tag{2.15}$$

We note that Eq. (2.15) states the k-means objective on the matrix U. This follows directly from substituting K with a symmetric decomposition $UU^\top = K$ in Eq. (2.13) and Theorem 2.1. The existence of such a decomposition is guaranteed because K is a symmetric and real-valued matrix. This type of matrix is decomposable into a symmetric product of real-valued matrices by Cholesky factorization, where U is an upper triangular matrix, or by eigendecomposition, where the columns of U return scaled eigenvectors of K.

2.3.3 Graph Cuts and Factorizing Graph Laplacians

The objective of kernel k-means in sole dependence of similarities between points introduces another data representation by means of a graph. The symmetric, real-valued kernel matrix has also an interpretation as a weighted adjacency matrix W

to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Every data point $D_{j\cdot}$ corresponds thereby to a node and the edges $\{j, l\} \in \mathcal{E}$ are defined by those entries where W_{jl} is larger than zero. The larger the weight of an edge, the stronger the connection between its nodes. Now, let us transfer the notion of clusters based on maximizing the inter-cluster similarity to graphs. A cluster is then a set of nodes having weak connections to nodes outside of the cluster. In other words, if we imagine to cut the edges connecting every cluster to its outside, then we strive to cut as few (strong) edges as possible. However, this formulation entails that single nodes, which are not connected, are optimal cluster candidates. In order to favor larger clusters, Hagen and Kahng (1992) propose to normalize the cut weights with the cluster size, introducing the ratio cut objective

$$\min_Y RCut(Y; W) = \sum_{s=1}^{r} \frac{Y_{\cdot s}^\top W(\mathbf{1} - Y_{\cdot s})}{|Y_{\cdot s}|} \qquad \text{s.t. } Y \in \mathbb{1}^{m \times r}.$$

The numerator $Y_{\cdot s}^\top W(\mathbf{1} - Y_{\cdot s})$ returns the cut from the nodes inside to the nodes outside of the cluster s. Shi and Malik (2000) incorporate the edge weights into the normalization term. This modification allows for larger cut weights of strongly connected clusters. The objective is called normalized cut, given as

$$\min_Y NCut(Y; W) = \sum_{s=1}^{r} \frac{Y_{\cdot s}^\top W(\mathbf{1} - Y_{\cdot s})}{Y_{\cdot s} I_W Y_{\cdot s}} \qquad \text{s.t. } Y \in \mathbb{1}^{m \times r}.$$

We write short $I_W = \text{diag}(W\mathbf{1})$ for the diagonal matrix, denoting the sum of weights to all neighbors of node j on the j-th diagonal entry. Both objectives, ratio and normalized cut, are transferable into a form which is similar to the kernel k-means objective. We transform the cut value of cluster s to

$$\begin{aligned} Y_{\cdot s}^\top W(\mathbf{1} - Y_{\cdot s}) &= \sum_{j=1}^{m} Y_{js} \left(|W_{j\cdot}| - W_{j\cdot} Y_{\cdot s}\right) = \sum_{j=1}^{m} Y_{js}(|W_{j\cdot}| Y_{js} - W_{j\cdot} Y_{\cdot s}) \\ &= Y_{\cdot s}^\top (I_W - W) Y_{\cdot s}. \end{aligned}$$

Using this transformation, the objective cut functions correspond to a trace optimization problem of the form

$$\min_Y \sum_{s=1}^{r} \frac{Y_{\cdot s}^\top W(\mathbf{1} - Y_{\cdot s})}{Y_{\cdot s}^\top J Y_{\cdot s}} = \text{tr}\left(Y^\top (I_W - W) Y \left(Y^\top J Y\right)^{-1}\right) \qquad \text{s.t. } Y \in \mathbb{1}^{m \times r},$$

where J is a diagonal matrix. If we set $J = I$ then the objective function above is equal to ratio cut and $J = I_W$ corresponds to normalized cut. We observe that the trace optimization problem is similar to the kernel k-means objective (2.13) from Theorem 2.2. Correspondingly, we derive the following equivalence of optimization problems.

TABLE 2.2: Popular graph Laplacians and corresponding eigenproblems, returning the same set of eigenvectors v and eigenvectors λ.

	Graph Laplacian	Equivalent eigenproblems
Difference Laplacian	$L_d = I_W - W$	$L_d v = \lambda v$
Symmetric Laplacian	$L_s = I - I_W^{-1/2} W I_W^{-1/2}$	$L_s I_W^{1/2} v = \lambda I_W^{-1/2} v$
Random Walk Laplacian	$L_r = I - I_W^{-1} W$	$L_r v = \lambda I_W^{-1} v$

COROLLARY 2.3. *Let $W \in \mathbb{R}^{m \times m}$ be a symmetric, real valued matrix and let $J \in \{I, I_W\}$ be an $m \times m$ diagonal matrix. The matrix $L = J^{-1} I_W - J^{-1/2} W J^{-1/2}$ is positive semidefinite. Let $\lambda > 0$ such that the matrix $\lambda I - L = UU^\top$ has a symmetric decomposition. The following optimization problems are then equivalent:*

$$\min_Y \sum_{s=1}^{r} \frac{Y_{\cdot s}^\top W (\mathbf{1} - Y_{\cdot s})}{Y_{\cdot s}^\top J Y_{\cdot s}} \qquad s.t.\ Y \in \mathbb{1}^{m \times r}, \tag{2.16}$$

$$\min_Y \operatorname{tr}\left(Z^\top L Z\right), \qquad s.t.\ Z = J^{1/2} Y \left(Y^\top J Y\right)^{-1/2}, Y \in \mathbb{1}^{m \times r}, \tag{2.17}$$

$$\min_Y \left\| -L - ZZ^\top \right\|^2, \qquad s.t.\ Z = J^{1/2} Y \left(Y^\top J Y\right)^{-1/2}, Y \in \mathbb{1}^{m \times r}, \tag{2.18}$$

$$\min_Y \left\| UU^\top - ZZ^\top \right\|^2 \qquad s.t.\ Z = J^{1/2} Y \left(Y^\top J Y\right)^{-1/2}, Y \in \mathbb{1}^{m \times r}, \tag{2.19}$$

The name spectral clustering derives from its optimization scheme using a spectral relaxation (cf. Section 3.4), based on Eq. (2.17). The matrix L from Corollary 2.3 for $J \in \{I, I_W\}$ is called graph Laplacian (Mohar et al., 1991; Chung, 1997). There are three popular definitions of a graph Laplacian, which we summarize in Table 2.2 together with the relations between their spectra. The spectrum of graph Laplacians is interesting because the multiplicity of their smallest eigenvalue, which is zero, is equal to the number of connected components in the graph. Furthermore, the eigenvectors to the eigenvalue zero indicate precisely the connected components. This property is easily understood, considering that suitably reordering the columns and rows of W results in a block-diagonal form of W. Let us consider a single connected component $\mathcal{C} \subseteq \mathcal{V}$, encompassing $|\mathcal{C}| = c$ nodes. If W is arranged such that the nodes in the connected component are represented by the first c columns and rows, then we have $W_{jl} = 0$ for every $j \in \mathcal{C}$ $(j \leq c)$ and $l \notin \mathcal{C}$ $(l > c)$; otherwise l would belong to the connected component. The resulting matrix W has a block diagonal form. Let $v \in \{0, 1\}^m$ be the vector indicating the

set $\mathcal{C}$, having the first c entries equal to one and all other entries equal zero. Then v satisfies the following relation:

$$Wv = \left(\begin{array}{ccc:c} W_{11} & \dots & W_{1c} & \\ \vdots & & \vdots & \mathbf{0} \\ W_{c1} & \dots & W_{cc} & \\ \hdashline \multicolumn{3}{c:}{\mathbf{0}} & \widehat{W} \end{array}\right)\left(\begin{array}{c} 1 \\ \vdots \\ 1 \\ \hdashline \mathbf{0} \end{array}\right) = \left(\begin{array}{c} \mid W_{1\cdot} \mid \\ \vdots \\ \mid W_{c\cdot} \mid \\ \hdashline \mathbf{0} \end{array}\right) = I_W v.$$

Subtracting $I_W v$ in the equation above yields that v is an eigenvector of the difference Laplacian with eigenvalue zero. Similarly, multiplying with I_W^{-1} from the left, yields that v is an eigenvector of the random walk Laplacian and multiplying with $I_W^{-1/2}$ from the left shows that $I_W^{1/2} v$ is an eigenvector of the symmetric Laplacian with eigenvalue zero (cf. Table 2.2).

Usually, the graph representation W is determined in a preprocessing step as a Gauss kernel matrix or the (weighted) adjacency matrix of the ϵ-neighborhood or k-nearest neighbor graph. Since the resulting edge weights depend on the graph type and parameter setting of this preprocessing step, we can generally not assume that the graph exhibits perfect clustering properties. Therefore, newer approaches of spectral clustering aim at learning the graph representation together with the resulting clustering (Bojchevski et al., 2017; Kang et al., 2018).

2.4 Two-Sided Clustering

In some application areas, such as collaborative filtering and gene expression analysis, we can not expect that clusters are identifiable based on the similarity of data points on the whole feature space. As an example, we cite the clustering of users according to movie preferences. Given a user times movie database, where each entry reflects the given rating, we most likely will not be able to find a set of users which give similar ratings on all movies. This effect can be seen as a manifestation of the *curse of dimensionality*, broadly stating that points in a high dimensional data set are likely to be approximately equidistant (Aggarwal et al., 2001; Beyer et al., 1999). A possible solution to this problem is to identify a group of users together with a small subset of the provided movies where the ratings are similar. This introduces the task of subspace clustering, the identification of the subspaces in which data points exhibit a clear cluster structure (Kriegel et al., 2009).

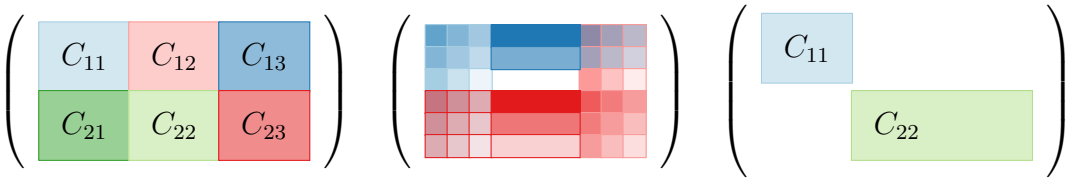

FIGURE 2.1: Variants of row- and column-partitioning biclusters: checkerboard model (left), plaid model (middle) and block-diagonal model (right). Best viewed in color.

TABLE 2.3: Overview of objective functions proposed for two-sided clustering.

	Objective
Checkerboard	$\min_{X,C,Y} \left\|D - YCX^\top\right\|^2$ s.t. $X \in \mathbb{1}^{n\times u}, Y \in \mathbb{1}^{m\times r}, C \in \mathbb{R}^{r\times u}$
Plaid	$\min_{X,Y} \left\|D - YY^\dagger D - DXX^\dagger + YCX^\top\right\|^2$ s.t. $X \in \mathbb{1}^{n\times u}, Y \in \mathbb{1}^{m\times r}, C = Y^\dagger DX$
Diagonal	$\min_{X,C,Y} \left\|D - YCX^\top\right\|^2$ s.t. $X \in \mathbb{1}^{n\times r}, Y \in \mathbb{1}^{m\times r}, C = \operatorname{diag}(C_{11}, \ldots, C_{rr})$
Binary	$\min_{X,Y} \left\|D - YX^\top\right\|^2$ s.t. $X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r}$
Boolean	$\min_{X,Y} \left\|D - Y \odot X^\top\right\|^2$ s.t. $X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r}$

Assuming that the cluster subspaces are spanned by a subset of the features, matrix factorizations for subspace clustering require not one but two binary matrices; one on the left to indicate the clustering of the data points and one on the right to indicate the feature space in which the data points cluster. A third middle matrix can be employed to scale the indicated clusters. In this case, we speak of a tri-factorization. We summarize the discussed factorizations of this section in Table 2.3. The discussed models are also known as two-mode clusters (Mechelen et al., 2004; Rosmalen et al., 2009) or biclusters (Busygin et al., 2008).

2.4.1 Checkerboard Clustering

The model of checkerboard clustering assumes that the data matrix is partitioned into a set of row clusters $\mathcal{J}_1, \ldots, \mathcal{J}_r \subseteq \{1, \ldots, m\}$ and a set of column clusters

$\mathcal{I}_1, \ldots, \mathcal{I}_u \subseteq \{1, \ldots, n\}$ such that every combination of a row- with a column-cluster creates a bicluster $(\mathcal{I}_t, \mathcal{J}_s)$. Note, that the numbers of row and column clusters are potentially different. We denote here with the bicluster set $\mathcal{B}$ the collection of all possible combinations of row- and column-clusters. The elements of the data matrix belonging to a bicluster are approximated with the mean value in the bicluster. Hence, the task of checkerboard clustering is to find an optimal partition of rows and columns such that every data entry does not differ much from the average data value in the bicluster. More formally, the objective is as follows:

$$\min_{\mathcal{B}} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(D_{ji} - \sum_{(\mathcal{J},\mathcal{I}) \in \mathcal{B}:\ (j,i) \in \mathcal{J} \times \mathcal{I}} \mu(D_{\mathcal{J}\mathcal{I}}) \right)^2 . \tag{2.20}$$

We employ the function $\mu(A)$ to denote the average value of all entries in the matrix A. This objective is transferable into a matrix factorization problem, involving a tri-factorization as outlined by the following theorem.

THEOREM 2.4. *The following optimization problems are equivalent to the objective from Eq.* (2.20) *subject to*

$$\mathcal{B} \in \left\{ \{\mathcal{I}_1, \ldots, \mathcal{I}_u\} \times \{\mathcal{J}_1, \ldots, \mathcal{J}_r\} \middle| \dot{\bigcup}_t \mathcal{I}_t = \{1, \ldots, n\},\ \dot{\bigcup}_s \mathcal{J}_s = \{1, \ldots, m\} \right\},$$

that is the set of biclusters is given as the cartesian product of a partition of the rows and the columns.

$$\min_{X,C,Y} \left\| D - YCX^\top \right\|^2 \qquad s.t.\ X \in \mathbb{1}^{n \times u},\ Y \in \mathbb{1}^{m \times r}, C \in \mathbb{R}^{r \times u} \tag{2.21}$$

$$\min_{X,Y} \left\| D - YCX^\top \right\|^2 \qquad s.t.\ X \in \mathbb{1}^{n \times u},\ Y \in \mathbb{1}^{m \times r}, C = Y^\dagger D {X^\dagger}^\top \tag{2.22}$$

$$\max_{X,Y} \operatorname{tr}\left(D^\top YCX^\top \right) \qquad s.t.\ X \in \mathbb{1}^{n \times u},\ Y \in \mathbb{1}^{m \times r}, C = Y^\dagger D {X^\dagger}^\top \tag{2.23}$$

The proof follows the techniques of expanding the Frobenius inner product into a sum and employing the convexity of matrix factorization objectives when all but one matrix is fixed. We do not explicitly state the proof of this theorem here, but it is easily adapted from the proof of Theorem 2.1.

Since the columns in Y and X are orthogonal, there is a permutation of rows and columns such that every bicluster appears as a coherent block in the factorization matrix. This is where the name checkerboard clustering comes from, the visualization by the left matrix in Figure 2.1 shows that the partition of the data matrix by biclusters results in a checkerboard pattern. There are two row clusters and three column clusters. Each intersection of row cluster s and column cluster t reflects one bicluster, which is approximated by the constant C_{st}. Such an approximation with constant biclusters goes back to Hartigan (1972), who proposes said approximation for a tree partitioning model.

The equivalence of the objectives from Eqs. (2.21) and (2.22) is attributable to Gaul and Schader (1996) and allows for an alternating minimization with respect to one matrix, while fixing the other two (Maurizio, 2001; Wang et al., 2011; Cho et al., 2004). This procedure adapts Lloyd's algorithm from k-means clustering (cf. Section 3.2). Furthermore, the orthogonal nonnegative relaxation (cf. Section 3.3) is also adapted for the optimization of tri-factorizations (Ding et al., 2006b; Yoo and Choi, 2010). The relationship between checkerboard and k-means clustering becomes apparent when determining that every feature belongs to its own cluster, that is setting $X = I_n$ in Eq. (2.22). This modification transfers the two-sided clustering of the checkerboard model into the one-sided clustering of k-means. The matrix C represents the cluster centroids in this scenario.

2.4.2 Plaid Model

The plaid model originates from a bioinformatics application in microarray data analysis. Microarrays are used to measure and reflect the gene expressions of patients. Let us say, the number of patients is m and the number of genes is n. Among a set of patients, some genes may co-regulate. That is, the genes exhibit similar expression patterns among the set of patients. Such a set of patients together with the co-regulating genes identifies a bicluster. In this case, saying that every bicluster approximates the corresponding part of the data matrix by a single aggregated value, as known from checkerboard clustering, is not enough. Some genes have generally higher or lower expression levels than other genes and the same holds for patients. Therefore, the plaid model introduces bias terms for each patient $\mu(D_{j\mathcal{I}})$ and gene $\mu(D_{\mathcal{J}i})$ to model deviations from the average

bicluster value. Denoting again with $\mathcal{B}$ the set of all possible biclusterings, we state the objective of plaid clustering as

$$\min_{\mathcal{B}} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(D_{ji} - \sum_{\substack{(\mathcal{J},\cdot)\in\mathcal{B}:\\ j\in\mathcal{J}}} \mu(D_{\mathcal{J}i}) - \sum_{\substack{(\cdot,\mathcal{I})\in\mathcal{B}:\\ i\in\mathcal{I}}} \mu(D_{j\mathcal{I}}) + \sum_{\substack{(\mathcal{J},\mathcal{I})\in\mathcal{B}:\\ (j,i)\in\mathcal{J}\times\mathcal{I}}} \mu(D_{\mathcal{J}\mathcal{I}}) \right)^2 . \quad (2.24)$$

The objective combines one-sided clusterings of the data matrix and its transposed, resulting in a plaid structure as depicted on the middle of Figure 2.1. Adding the last term in Eq. (2.24) is required as a result of the inclusion-exclusion principle, since the approximation of the data in the intersecting area is subtracted twice, once for row- and once for column-clusters.

Genes are not expected to take part in only one biological process. Hence, the biclusters are for biological applications generally not restricted to partitions of rows and columns. The algorithms proposed by Cheng and Church (2000); Lazzeroni and Owen (2002) and Turner et al. (2005) sequentially optimize the biclusters one-by-one. We call this the greedy approach, which is discussed in Section 3.1. However, assuming the set of biclusters to be restricted to row- and column partitions enables the adaptation of k-means optimization procedures. We state here the plaid optimization task with respect to partitioning biclusters in matrix factorization form.

THEOREM 2.5. *The following optimization problems are equivalent to the plaid optimization problem from Eq.* (2.24) *subject to*

$$\mathcal{B} \in \left\{ \{\mathcal{I}_1,\ldots,\mathcal{I}_u\} \times \{\mathcal{J}_1,\ldots,\mathcal{J}_r\} \middle| \dot{\bigcup_t} \mathcal{I}_t = \{1,\ldots,n\},\ \dot{\bigcup_s} \mathcal{J}_s = \{1,\ldots,m\} \right\},$$

when the search space of the row- and column-cluster indicating matrices is restricted to partition matrices $X \in \mathbb{1}^{m\times r}$ and $Y \in \mathbb{1}^{m\times r}$:

$$\min_{X,Y} \left\| D - YY^\dagger D - DXX^\dagger + YCX^\top \right\|^2 \qquad s.t.\ C = Y^\dagger D {X^\dagger}^\top \quad (2.25)$$

$$\min_{X,Y} \left\| A - YY^\dagger A \right\|^2 \qquad s.t.\ A = D - DXX^\dagger \quad (2.26)$$

$$\min_{X,Y} \left\| A - AXX^\dagger \right\|^2 \qquad s.t.\ A = D - YY^\dagger D \quad (2.27)$$

Cho et al. (2004) propose the alternating optimization based on the equality of Eqs. (2.25), (2.26) and (2.27). We observe that the last two objectives are equivalent to the k-means clustering given in Eq. (2.8) if the matrix A is fixed. Correspondingly, an optimization performing alternating updates with respect to Y based on Eq. (2.26) and with respect to X based on Eq. (2.27) converges to local minima of the plaid optimization problem.

2.4.3 Block Diagonal Model and Bipartite Graph Cuts

The models discussed so far allow for varying numbers of row- and column-clusters, being arbitrarily combined to create a bicluster. However, a special interpretation is given for tri-factorizations approximating the data matrix as known from checkerboard clustering (cf. Eq. (2.21)) when biclusters represent a one-to-one relationship of row- and column-clusters. This task is also known under the name of constant biclustering (Madeira and Oliveira, 2004). The corresponding matrix factorizations follow from the equivalences of checkerboard objectives in Theorem 2.4 and the observation that a one-to-one correspondence between row- and column-clusters implies that the middle scaling matrix C is diagonal.

COROLLARY 2.6. *Let the set of possible biclusters be given as*

$$\mathcal{B} = \left\{ (\mathcal{I}_1, \mathcal{J}_1), \ldots, (\mathcal{I}_r, \mathcal{J}_r) \middle| \dot{\bigcup}_s \mathcal{I}_s = \{1, \ldots, n\},\ \dot{\bigcup}_s \mathcal{J}_s = \{1, \ldots, m\} \right\}.$$

and denote with the matrix W the symmetric $(m+n) \times (m+n)$ matrix defined as

$$W = \begin{pmatrix} \mathbf{0} & D \\ D^\top & \mathbf{0} \end{pmatrix}.$$

The following optimization problems are equivalent to the objective of Eq. (2.20)*, where $\mathcal{B}$ is defined as above, if the search space of $X \in \mathbb{1}^{n \times r}$ is restricted to partitioning matrices:*

$$\min_{X,C,Y} \left\| D - YCX^\top \right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m \times r}, C = \operatorname{diag}(\mathbf{c}), \mathbf{c} \in \mathbb{R}^r \quad (2.28)$$

$$\min_{X,Y} \left\| D - YCX^\top \right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m \times r}, C = \operatorname{diag}(\mathbf{c}), \mathbf{c}_s = \frac{Y_{\cdot s}^\top D X_{\cdot s}}{|Y_{\cdot s}||X_{\cdot s}|} \quad (2.29)$$

$$\max_{X,Y} \operatorname{tr}\left(D^\top YCX^\top\right) \qquad s.t.\ Y \in \mathbb{1}^{m \times r}, C = \operatorname{diag}(\mathbf{c}), \mathbf{c} \in \mathbb{R}^r \quad (2.30)$$

$$\max_{Z} \operatorname{tr}\left(Z^\top W Z\right) \qquad s.t.\ Z = YC^{1/2}, Y \in \mathbb{1}^{(n+m) \times r}, C = \operatorname{diag}(\mathbf{c}), \mathbf{c} \in \mathbb{R}^r \quad (2.31)$$

The tri-factorization $YCX^\top$ from Eq. (2.28) boils down to the sum of r outer products $Y_{\cdot s}X_{\cdot s}$, which are scaled by the diagonal entry C_{ss}. We call the corresponding tasks from Corollary 2.6 block-diagonal clustering for the reason that a suitable reordering of rows and columns displays a block-diagonal factorization, as shown on the right in Figure 2.1.

Based on the equivalence of Eq. (2.28) and (2.29) which goes back to Mirkin et al. (1995), Han et al. (2017) propose an alternating minimization for block-diagonal clustering. Other optimization schemes emerge from the interpretation of block-diagonal models as bipartite graph cuts. In this view, the data matrix

indicates a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ having $|\mathcal{V}| = n + m$ nodes corresponding to the union of features and samples, where edge weights are given via the weighted adjacency matrix W. If W is defined as in Corollary 2.6, then the eigenvalues and eigenvectors of W correspond to the singular values and vectors of D. To see that, assume that $(y, x) \in \mathbb{R}^{n+m}$ is an eigenvector of W with eigenvalue λ; it holds that

$$\begin{aligned} \begin{pmatrix} \mathbf{0} & D \\ D^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \lambda \begin{pmatrix} y \\ x \end{pmatrix} \; &\Leftrightarrow Dx = \lambda y \;\wedge\; D^\top y = \lambda x \\ &\Leftrightarrow DD^\top y = \lambda^2 y \wedge D^\top y = \lambda x. \end{aligned}$$

The equation above yields that eigenvalues of W correspond to the singular values of D. The eigenvector (y, x) of W is composed of the left- and right-singular vectors y and x of D. Note, that this relationship still holds if the weight matrix is normalized as known from spectral clustering. Therefore, the eigenvectors of the $(n + m) \times (n + m)$ similarity matrix W are efficiently computed by the singular vectors of the $m \times n$ matrix D.

The computation of W's eigenvectors is relevant for a spectral relaxation, as originally proposed by Zha et al. (2001) and Dhillon (2001). Newer methods minimizing bipartite graph cuts discuss how to learn the graph representation of W, respectively D, together with the optimal clustering (Nie et al., 2017). This trend has also been observed with respect to spectral clustering (cf. Section 2.3.3). The spectral relaxation emerges from the fact that the trace maximization in Eq. (2.31) yields the eigenvectors to the largest eigenvalues of W if the matrix Z is only required to be real-valued and orthonormal. We discuss this relationship more in Section 3.4. Here, the eigenvectors are given by the singular vectors of D, thus we could also speak of a singular value relaxation. The relationship of block-diagonal clustering and SVD becomes also apparent when we regard the tri-factorization in Eq. (2.28). Relaxing the partitioning constraints of X and Y to orthogonality constraints for real-valued matrices X and Y results in the optimization problem of truncated SVD. In this case, the singular values having the largest absolute values are denoted on the diagonal of matrix C.

2.4.4 Binary Matrix Factorization

A somewhat special case of biclustering arises if the data matrix is binary. This affects applications of collaborative filtering (a movie either is watched or not), text analysis (word occurrences are binary) or genome data analysis (considering, e.g. mutations). In this situation, a decomposition into binary matrices befits the

interpretability of the result. We state the optimization problem of binary matrix factorization in its general form as

$$\min_{X,Y} \left\| D - YX^\top \right\|^2 \qquad \text{s.t. } X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r}. \tag{BiMF}$$

The requirement that both factor matrices have orthogonal columns as known from block-diagonal clustering is too strict for most of the relevant applications. A movie might be watched by multiple user groups and correspondingly, groups of words and documents or mutations and patients do not follow a one-to-one relationship. Li (2005) shows that alternating minimization is possible nevertheless, if one of the matrices indicates a partition. We summarize the objectives equivalent to (BiMF) for the partially orthogonal case in the following theorem. We introduce here the set $\Theta(A)$ as the set of all binary matrices which result from thresholding a real-valued matrix A at one half, where the binary value to which one half is rounded, is undetermined.

THEOREM 2.7. *Define* $\Theta(A)$ *as the set*

$$\Theta(A) = \{B | B_{ji} = \theta(A_{ji}) \text{ for } A_{ji} \neq 1/2, B_{ji} \in \{0,1\} \text{ for } A_{ji} = 1/2\}$$

The following optimization problems are equivalent:

$$\min_{X,Y} \left\| D - YX^\top \right\|^2 \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \mathbb{1}^{m\times r} \tag{2.32}$$

$$\min_{Y} \left\| D - YX^\top \right\|^2 \qquad s.t.\ X \in \Theta\left(D^\top Y \left(Y^\top Y\right)^{-1}\right), Y \in \mathbb{1}^{m\times r} \tag{2.33}$$

$$\max_{X,Y} \operatorname{tr}\left(Y^\top (2D - \mathbf{1})X\right) \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \mathbb{1}^{m\times r} \tag{2.34}$$

Koyutürk and Grama (2003) propose the subsequent optimization of clusters, aiming for the optimization of the objective in Eq. (2.33), where row-clusters do not overlap. Therefore, the algorithm PROXIMUS is proposed, optimizing a rank-one factorization of the general objective (BiMF) via alternating minimization. We will discuss this greedy approach more in detail in Section 3.1. Shen et al. (2009) remark that the results of PROXIMUS are highly sensitive to the initialization. They propose an initialization based on a relaxation of Eq. (2.34), allowing binary matrices to have entries between zero and one.

Zhang et al. (2007, 2010) aim at solving the general problem (BiMF). They discuss a relaxation of binary to nonnegative matrices and derive an optimization scheme to determine suitable thresholds to discretize NMF solutions. In addition, they propose a multiplicative update algorithm for the optimization of (NMF) with an integrated penalization term for nonbinary values. A follow-up paper discusses the application of these multiplicative updates for symmetric binary matrix factorizations (Zhang et al., 2013). We discuss this penalization approach more in detail in Section 3.5.

2.4.5 Boolean Matrix Factorization

The attempt to allow for more overlap between clusters of binary data is is naturally incorporated if the factorization is computed in Boolean algebra.

DEFINITION 2.8 (BOOLEAN ALGEBRA). Let $\mathcal{A}$ be a set, let $\oplus$ and $\odot$ be two binary relations and $\overline{\cdot}$ an unary relation on the set A. The structure $(A, \oplus, \odot, \overline{\cdot})$ is called a Boolean algebra if the following requirements hold

1. $(A, \oplus, \odot)$ is a commutative semiring:
   (a) $(A, \oplus)$ is a commutative monoid with identity element 0
   (b) $(A, \odot)$ is a commutative monoid with identity element 1
   (c) multiplication distributes over addition:
   $$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c) \text{ for } a, b, c \in A$$
   (d) multiplication with 0 annihilates: $a \odot 0 = 0$ for $a \in A$
2. Addition distributes over multiplication:
   $$a \oplus (b \odot c) = (a \oplus b) \odot (a \oplus c) \text{ for } a, b, c \in A$$
3. Complementary elements exist: $a \odot \bar{a} = 0$ and $a \oplus \bar{a} = 1$ for $a \in A$.

We are in this section interested in the two-element algebra $(\{0, 1\}, \oplus, \odot, \bar{})$, where we define the Boolean complement as $\bar{a} = 1 - a$. Note, that the only deviation of the arithmetic in two-element Boolean algebra to traditional operations is that $1 \oplus 1 = 1$. From the definition of the Boolean algebra follows that the structure $(\{0, 1\}, \oplus, \odot)$ is a semiring, notably the element 1 has no additive inverse. On this structure we can define vector operations such as addition, scalar multiplication and an inner product similarly to the Euclidean vector space. Then, the linear mappings from the n-dimensional to the m-dimensional Boolean space are given by the Boolean product of an $m \times n$ binary matrix with an n-dimensional binary vector (Gudder and Latrémolière, 2009). Given $Y \in \{0, 1\}^{m \times r}$ and $X \in \{0, 1\}^{n \times r}$, the Boolean matrix product is either defined elementwise as the Boolean inner product of row and column vectors or as the sum of outer product matrices

$$\left(Y \odot X^\top\right)_{ji} = Y_{j1} X_{i1} \oplus \ldots \oplus Y_{jr} X_{ir}, \qquad Y \odot X^\top = Y_{\cdot 1} X_{\cdot 1}^\top \oplus \ldots \oplus Y_{\cdot r} X_{\cdot r}^\top.$$

Other definitions of algebraic structures in the binary space are also thinkable as discussed in Miettinen (2015), yet these extensions are out of the scope of this work. The space defined via the Boolean operations has similar properties like the

nonnegative space, which also lacks additive inverses. Like the nonnegative space, the Boolean space is closed under its matrix product which is contrasted by the binary matrix product, returning possibly nonbinary matrices. We now define the Boolean matrix factorization objective as

$$\min_{X,Y} \left\| D - Y \odot X^\top \right\|^2 \qquad \text{s.t. } X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r}. \tag{BMF}$$

We express the Boolean product in elementary algebra via the Heaviside step function in order to avoid confusion of Boolean and numerical vector operations. In this manner we denote equivalent formulations of the Boolean matrix factorization problem.

THEOREM 2.9. *The following optimization problems are equivalent to the one of* (BMF)*:*

$$\min_{X,Y} \left\| D - \theta\big(YX^\top\big) \right\|^2 \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r} \tag{2.35}$$

$$\min_{X,Y} \left| D - \theta\big(YX^\top\big) \right| \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r} \tag{2.36}$$

$$\max_{X,Y} \operatorname{tr}\left((2D - \mathbf{1})\theta\big(XY^\top\big)\right) \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \{0,1\}^{m\times r} \tag{2.37}$$

The most popular method to approximate problem (BMF) is the greedy approach Asso (Miettinen et al., 2008). Outer products are subsequently computed, determining a feature cluster $X_{\cdot s}$ first and optimizing its cluster assignment $Y_{\cdot s}$ afterwards, such that the approximization error is minimized.

Boolean matrix factorization has a noteworthy relationship to frequent pattern mining (Aggarwal and Han, 2014) and discussing BMF using the terminology of pattern mining often comes more natural. To this end, we shortly introduce the denotation of pattern mining. In this respect, the binary data matrix represents a transactional database of m transactions and n items. Every transaction corresponds to a row of the data matrix, indicating the items which are contained in the transaction. A set of items is called a pattern and we say a transaction supports a pattern if the pattern is a subset of the transaction. The support of a pattern is then the number of supporting transactions in the database. Regarding the factorization of problem (BMF), every outer product indicates a pattern $X_{\cdot s}$ and its assigned transactions $Y_{\cdot s}$. A special case arises if the factor matrix Y in problem (BMF) is restricted to the supporting sets of X. This introduces the constraint $\langle D, \theta(YX^\top)\rangle = |\theta(YX^\top)|$ which enforces that all ones in the Boolean product matrix are covered by the data matrix.

COROLLARY 2.10. *The following optimization problems are equivalent if the search space is restricted to binary matrices $X \in \{0,1\}^{n\times r}$, $Y \in \{0,1\}^{m\times r}$:*

$$\min_{X,Y} \left\| D - \theta\big(YX^\top\big) \right\|^2 \qquad s.t.\ \left\langle D, \theta\big(YX^\top\big) \right\rangle = \left| \theta\big(YX^\top\big) \right| \tag{2.38}$$

$$\max_{X,Y} \left| \theta\big(YX^\top\big) \right| \qquad s.t.\ \left\langle D, \theta\big(YX^\top\big) \right\rangle = \left| \theta\big(YX^\top\big) \right| \tag{2.39}$$

Geerts et al. (2004) introduce the task of tiling to maximize the number of ones in the Boolean product according to Eq. (2.39). The term *tile* for the outer product $Y_{\cdot s}X_{\cdot s}^\top$ reflects its visualization as a single block matrix for suitably arranged columns and rows. The task to determine the constrained Boolean matrix factorization in Eq. (2.38) is also known as dominated Boolean matrix factorization (Miettinen, 2010). Belohlavek and Vychodil (2010); Belohlavek and Trnecka (2015) propose a greedy approach for dominant matrix factorizations based on an approximation algorithm of the set cover problem. Kontonasios and De Bie (2010) and Xiang et al. (2011) argue that lifting the restriction to the support sets enables more succinct descriptions and enhances robustness to noise; every flip of a single bit in the interior of a tile breaks it into two. The proposed algorithms handle the extension from the supporting set only in a post-processing step and provide no mechanisms to directly approximate a solution to Eq. (2.38).

CHAPTER 3

Optimizing Subject to Binary Constraints

So far, we have only roughly discussed which optimization schemes are used for the considered optimization problems. The attentive reader might have noticed that some approaches recur over various optimization tasks. In this section, we discuss the most important optimization paradigms: each approach is presented by means of a popular application. The chosen example often illustrates how and why the optimization scheme is suitable for the task at hand. The transfer to related problems is then trivial in most cases. Note that a mixture of the discussed approaches is often applied.

3.1 Greedy Approach

We call the greedy approach a successive calculation of the clusters. The word *greedy* refers to the optimization of the current cluster regardless of the possibility to improve the approximation by subsequent clusters. This approach is theoretically founded for the computation of truncated singular value decompositions. We recall that the SVD of a matrix $D = V\Sigma U^\top$ also provides the optimal approximation of rank r: let $\mathcal{S} = \{1, \dots, r\}$ denote the set of all indices up to r, then we have

$$(V_{\cdot\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}, U_{\cdot\mathcal{S}}) \in \arg\min_{X,Y} \left\| D - YX^\top \right\|^2 \qquad \text{s.t. } X \in \mathbb{R}^{n\times r}, Y \in \mathbb{R}^{m\times r}. \tag{3.1}$$

The derivation of this property and a more general discussion of SVD related optimization problems and their optimization can be found in (Udell et al., 2016). Due to the one-to-one relationship between the singular value decomposition of a matrix and its low-rank approximation, the calculation of the optimal rank-r factorization is reducible to calculating an optimal rank-one factorization, given

ALGORITHM 1 Computing a solution to the binary matrix factorization problem in Eq. (2.32) via the greedy approach.

1: **FUNCTION** PROXIMUS(D, r)
2: **FOR** $s \in \{1, \ldots, r\}$ **DO**
3: $(X_{\cdot s}, Y_{\cdot s}) \leftarrow \arg\min_{x,y} \left\| D - yx^\top \right\|^2$ s.t. $x \in \{0,1\}^n, y \in \{0,1\}^m$
4: $D \leftarrow \operatorname{diag}\left(\overline{Y_{\cdot s}}\right) D$
5: **END FOR**
6: **RETURN** Y
7: **END FUNCTION**

the optimal rank-$(r-1)$ factorization. Denoting with $\mathcal{T} = \{1, \ldots, r-1\}$ the outer product indices of a rank-$(r-1)$ factorization, then from Eq. (3.1) follows that

$$\left\| D - V_{\cdot\mathcal{S}} \Sigma_{\mathcal{S}\mathcal{S}} U_{\cdot\mathcal{S}}^\top \right\|^2 = \min_{x,y} \left\| \left(D - V_{\cdot\mathcal{T}} \Sigma_{\mathcal{T}\mathcal{T}} U_{\cdot\mathcal{T}}^\top \right) - yx^\top \right\|^2 \quad \text{s.t. } x \in \mathbb{R}^n, y \in \mathbb{R}^m$$
$$\Rightarrow (V_{\cdot r} \Sigma_{rr}, U_{\cdot r}) \in \arg\min_{x,y} \left\| (D - V_{\cdot\mathcal{T}} \Sigma_{\mathcal{T}\mathcal{T}} U_{\cdot\mathcal{T}}^\top) - yx^\top \right\|^2 \quad \text{s.t. } x \in \mathbb{R}^n, y \in \mathbb{R}^m.$$

This relationship motivates a greedy approach. Computing the factor matrices outer product by outer product leads to the optimal solution. This property does not hold if nonnegativity or binary constraints are introduced. Still, the greedy optimization scheme might lead to satisfying solutions if the optimal rank-one factorization is much more easily computed than the factorization of a higher rank. The drawback of the greedy approach is the lack of quality guarantees, where comparable numerical optimization methods assure the convergence to a local minimum of the objective at least.

We exemplify the application of the greedy approach by means of the algorithm PROXIMUS depicted in Algorithm 1 (Koyutürk and Grama, 2003). Every bicluster $(X_{\cdot s}, Y_{\cdot s})$ is determined as a solution to the rank-one problem BiMF in step 3 and the data matrix is reduced in step 4; the rows which are assigned to the current cluster are set to zero. Here, the optimization of the rank-one factorization is achieved by alternating optimization, based on the results in Theorem 2.7. BiMF factorizations of rank one trivially have orthogonal columns. Hence the optimal binary vector x minimizing the optimization problem in step 3 is given as $x \in \Theta(D^\top y/|y|)$ when y is fixed. Similarly, is the optimal y is determined while fixing x. In a similar fashion are other two-sided clustering algorithms designed, computing plaid (Cheng and Church, 2000; Lazzeroni and Owen, 2002; Turner et al., 2005) and Boolean matrix factorizations (Miettinen et al., 2008; Geerts et al., 2004). The greedy optimization scheme is yet less suited for one-sided clustering. Imagine that the cluster center x in step 3 is allowed to have nonnegative

ALGORITHM 2 Lloyd's Alternating Minimization for k-means

1: **FUNCTION** KMEANS(D, r)
2: **FOR** $k \in \{1, \ldots, K\}$ **DO**
3: $X \leftarrow D^\top Y \left(Y^\top Y\right)^{-1} \in \arg\min_X \left\| D - YX^\top \right\|^2$ s.t. $X \in \mathbb{R}_+^{n \times r}$
4: **FOR** $j \in \{1, \ldots, m\}$ **DO**
5: $s \leftarrow \arg\min_{t \in \{1,\ldots,r\}} \left\| D_{j\cdot} - X_{\cdot t}^\top \right\|^2$
6: $Y_{j\cdot} \leftarrow e_s^\top$ ▷ e_s is the s-th standard basis vector
7: **END FOR**
8: **END FOR**
9: **RETURN** Y
10: **END FUNCTION**

values. Then the optimal rank-one factorization where y is restricted to binary values always assigns the whole dataset to one cluster and x reflects the centroid of the dataset. Optimizing the interplay between multiple clusters is particularly crucial for one-sided clusterings.

3.2 Alternating Minimization

Probably the best known clustering procedure is *the* k-means algorithm, also called Lloyd's algorithm (Lloyd, 1982). This method is often visually portrayed as the repetitive assignment of points to the nearest cluster followed by an update of the cluster centers, but it also performs a theoretically well-founded alternating minimization of the within point scatter (KM). The theoretical aspects such as convergence are studied in the wider scope of alternating minimization.

We outline Lloyds' minimization in Algorithm 2. In every iteration, every factor matrix is updated with the optimal solution of the objective when the other factor matrix is fixed. The optimum with respect to X is given by the equivalence of Eqs. (2.9) and (2.7). Likewise, the optimal partition matrix minimizing the approximation error of problem (KM) is easily computed: we set in every row exactly that entry to one, which corresponds to the closest cluster center as stated in steps 5 and 6.

The ability to denote the solution of objective (KM) when one matrix is fixed in closed form makes alternating optimization very appealing. The generally hard combinatorial problem to find the best cluster partition is disassembled into the iterative solution of easier problems. This optimization scheme is also easily adapted for tri-factorizations (cf. Section 2.4). Cruicial is here that the binary cluster in-

dicator matrix reflects a partition. Finding an optimal binary matrix minimizing the optimization error when the other matrix is fixed is itself a hard combinatorial problem for which no closed-form solution is known. There are some attempts to transfer the elegant alternating minimization to non-partitioning clusters (Chawla and Gionis, 2013; Whang et al., 2018). However, these methods require the specification of additional parameters which are difficult to set in advance, such as the amount of overlap or the number outliers in the data.

The drawback of alternating minimization is the tendency to get stuck in local, less desirable minima. Therefore, a suitable initialization of the matrices is particularly important. We refer the reader to Celebi et al. (2013) for an overview of initialization techniques.

3.3 Orthogonal Nonnegative Relaxation

We have mentioned in Section 2.2.1 that the relation of nonnegative matrix factorization to clustering stems from the near-orthogonality of factor matrices solving objective (NMF). If we further constrain the feasible set to nonnegative and orthogonal matrices, then we obtain a new learning task called Orthogonal NMF (ONMF). Various authors emphasize the relationship of orthogonal NMF and k-means, some even erroneously claim equivalence (Ding et al., 2005, 2006b; Li and Ding, 2006). Let us have a closer look at this relationship and revisit the k-means objectives from Theorem 2.1. The orthogonal relaxation is to disregard the special structure of the matrix Y as a binary partition matrix, and to require only that Y is an orthogonal nonnegative matrix. Using this relaxation, solutions to the k-means objective from Eq. (2.9) are approximated by solutions to the orthogonal nonnegative matrix factorization given as

$$\min_{Y,X} \left\| D - ZX^\top \right\|^2 \qquad \text{s.t. } Z^\top Z = I,\ Z \in \mathbb{R}_+^{m\times n},\ X \in \mathbb{R}_+^{n\times r}. \tag{3.2}$$

Orthogonal nonnegative matrix factorization is an NP-hard problem, yet approximable in polynomial time (Asteris et al., 2015). Requiring that the columns of Z are orthogonal and nonnegative implicates that every row of Z has at most one nonzero entry. Thus, the relaxed matrix Z also indicates a nonoverlapping clustering by its support: by its nonzero entries. The indicated clustering does however not necessarily yield a partition of the data points since some rows of Z could be entirely zero. Hence, not all points are necessarily assigned to a cluster via orthogonal nonnegative matrix factorization.

ALGORITHM 3 Clustering via orthogonal nonnegative matrix factorization

1: **FUNCTION** ONMF(D, r, ϵ) $\triangleright\ \epsilon \gtrsim 0$
2: $(Z, X) \leftarrow \arg\min_{Z,X} \left\| D - ZX^\top \right\|^2$ s.t. $Z^\top Z = I,\ Z \in \mathbb{R}_+^{m\times r},\ X \in \mathbb{R}_+^{n\times r}$
3: $Y \leftarrow \theta_\epsilon(Z)$
4: **RETURN** Y
5: **END FUNCTION**

Pompili et al. (2014) show that clusterings obtained via the orthogonal relaxation are related, albeit not equivalent to spherical k-means. Their evaluation shows that clusterings based on ONMF are determined by a Voronoi tesselation, where each cell is a covex cone. That is, the clusters are assigned according to the directions of the datapoints only. Opposed to spherical k-means, the norm of data points have here an influence on the found cluster directions: data points which have a larger distance to the origin have a larger influence on the found clusters. We summarize the method associated with the orthogonal relaxation of the k-means problem in Algorithm 3. The calculation of the orthonormal factorization in step 2 is left open and requires a survey on its own. We refer the reader here to the penalty methods using multiplicative updates (Ding et al., 2006b; Li and Ding, 2006), the optimization on the Stiefel manifold (Yoo and Choi, 2010) and the alternating minimization and augmented Lagrangian approaches (Pompili et al., 2014). The thresholding to binary cluster assignments in step 3 employs the parameter ϵ to set all values smaller than zero to zero, countering numerical instabilities of the returned solution Z.

3.4 Spectral Relaxation

The constraint to nonnegative factor matrices in ONMF morphs the polynomially solvable problem of SVD to an NP-hard problem. Thus, being able to derive a suitable clustering from an orthogonal, not necessarily nonnegative relaxation would speed up the computation. An orthogonal relaxation of discussed one-sided cluster objectives results in an eigendecomposition, as stated by the Ky Fan theorem (Fan, 1949). Let us have a look at the trace maximization objectives summarized for one-sided clustering objectives in Table 2.1. All these objectives have an orthogonal relaxation of the form

$$\sum_{s=1}^{k} \lambda_s = \max_{Z} \operatorname{tr}\left(Z^\top W Z\right) \qquad \text{s.t. } Z^\top Z = I,\ Z \in \mathbb{R}^{m\times r} \qquad \text{(KyFan)}$$

ALGORITHM 4 The orthogonal relaxation of spectral clustering.

1: **FUNCTION** $\mathrm{SC}(L, r)$ ▷ L: graph Laplacian
2: $Z \leftarrow \arg\min_{Z} \operatorname{tr}\left(Z^\top L Z\right)$ s.t. $Z^\top Z = I, Z \in \mathbb{R}^{m\times r}$ ▷ eigenvectors
3: $(X, Y) \leftarrow \arg\min_{X,Y} \left\|Z - YX^\top\right\|^2$ s.t. $Y \in \mathbb{1}^{m\times r}, X \in \mathbb{R}^{n\times r}$ ▷ k-means
4: **RETURN** Y
5: **END FUNCTION**

where $W \in \mathbb{R}^{m\times m}$ is a symmetric matrix. The Ky Fan theorem says that the minimizer of the trace maximization above is given by the eigenvectors to the r largest eigenvalues of W. The maximum value is attained at the sum of the r largest eigenvalues.

The application of this relaxation is mainly known from the pipeline of methods called spectral clustering, which originates from the orthogonal relaxation of the minimum cut objective (Luxburg, 2007). Note however, that there are some examples of graphs where the minimum rational cut and the solution obtained by spectral clustering diverge (Guattery and Miller, 1998). We summarize the steps of Spectral Clustering (SC) in Algorithm 4. The algorithm is specified with respect to the input graph Laplacian L (cf. Table 2.2) and the number of clusters r. We assume here that L is a symmetric matrix (unlike the random walk Laplacian) which implies that the eigenvectors of L are the solutions to problem (KyFan). Usually L is defined as the symmetric normalized Laplacian. In this setting, the smallest eigenvalues of the Laplacian are correspond to the largest eigenvalues of the normalized adjacency matrix, say $\tilde{W} = I_W^{-1/2} W I_W^{-1/2}$. The employed weighted adjacency matrix reflects local similarities of points and is to be computed in a preprocessing step. There are multiple options to do so. Usually, W is defined via the k-nearest neighbour graph. Denoting with A the adjacency matrix to the asymmetric k-nearest neighbor graph, the matrix W is then defined as $W = \frac{1}{2}(A + A^\top)$. Another possibility is to define adjacency via the ϵ-radius neighborhood graph. In this variant, we have $W_{jl} = 1$ if point $l \in \mathcal{N}_\epsilon(j)$ is in the ϵ-neighborhood of point j and $W_{jl} = 0$ otherwise. This is a symmetric relationship. Per definition, we set the diagonal elements of W to zero; the indicated graph is supposed to have no self-loops.

Given the graph Laplacian and the number of expected clusters r, the eigenvectors to the r smallest eigenvalues are computed in step 2. The eigenvectors are then discretized to a binary cluster indicator matrix. Although there are more simple ways to do this, a k-means clustering is here performed as the prevailing standard. According to Corollary 2.3, the application of k-means is theoretically justified, but this relation seems to be unknown.

ALGORITHM 5 Binary Penalty Algorithm

FUNCTION $\mathrm{BP}(D, r, \phi)$ ▷ ϕ has its minimum at binary matrices
  $(X, Y) \leftarrow \underset{X,Y}{\arg\min} \left\| D - YX^\top \right\|^2 + \phi(X) + \phi(Y)$ s.t. $Y \in \mathbb{R}_+^{m \times r}, X \in \mathbb{R}_+^{n \times r}$
  $(X, Y) \leftarrow (\theta(X), \theta(Y))$
  RETURN (X, Y)
END FUNCTION

The specification of the eigendecomposition's rank is not thoroughly discussed. Dhillon (2001) initially propose to employ the $\log_2(r)$ first eigenvectors to determine r clusters according to the spectral relaxation of bipartite graph cut. For implementations of spectral clustering, choosing the number of employed eigenvectors equal to the number of derived clusters r has prevailed.

3.5 Nonnegative Relaxation with Nonbinary Penalization

Among other things, we observe from the orthogonal relaxation, that the relaxation of binary to nonnegative values with a following discretization step may result in clusterings having differing properties than solutions of the original objective. A flexible approach to address optimization problems with binary constraints is adding a penalty term to the relaxed objective, pointing the optimal solutions to binary values. According to this scheme, Lazzeroni and Owen (2002) propose a heuristic, where nonnegative matrix are shifted into the direction of binary numbers. In detail, a constant which is increasing with the number of iterations is added to entries larger than 0.5 and subtracted from smaller entries. The drawback of this method is its sensitivity to the shifting constant. If the constant is rapidly increasing then the solution might get stuck in a local optimum, satisfying the binary constraint but approximating the data less well. Otherwise, if the size is slowly increasing then the convergence to approximately binary values is not given in a specified time limit (Turner et al., 2005). We discuss theoretical advances of this scheme in Chapter 5.

We state Algorithm 5 to schematize the optimization of a binary matrix factorization by means of nonbinary penalizers. The function ϕ_M is here the penalizing function, obtaining its minima at binary matrices. Zhang et al. (2010, 2013) propose the *Mexican hat* function to penalize nonbinary entries, which is given as

$$\phi_M(X) = \|X \circ X - X\|^2 = \sum_{i,s} \left(X_{is}^2 - X_{is}\right)^2 .$$

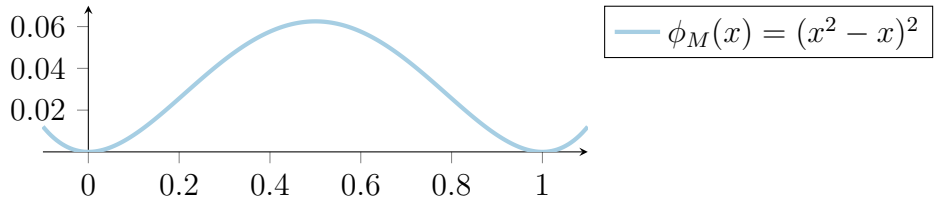

FIGURE 3.1: Plot of the Mexican hat function used as a penalizing function for nonbinary values.

The plot of this function is depicted in Figure 3.1. The Mexican hat function is smooth and thus the optimization in step 2 of Algorithm 5 can be accomplished by gradient descent. Zhang et al. (2010, 2013) propose multiplying updates for this purpose.

CHAPTER 4

A Spectral Approach to Density Based Clustering

Despite being one of the core tasks of data mining, and despite having been around since the 1930s (Driver and Kroeber, 1932; Klimek, 1935; Tryon, 1939), the question of clustering has not yet been answered in a manner that doesn't come with innate deficits. Equally, we will not be able to provide such an answer. Several advanced solutions to the clustering problem have become quite famous, and justly so, for delivering insight in data where the clusters do not offer themselves up easily. We have seen how spectral clustering employs distances defined by a weighted graph to identify the clusters which minimize the cut between them. *DBSCAN* (Ester et al., 1996) is a density-based clustering algorithm which has won the SIGKDD test of time award in 2014. Both spectral clustering and DBSCAN can find nonlinearly separable clusters, which trips up naive clustering approaches; these algorithms deliver good results. In this chapter, we propose a new clustering model which encompasses the strengths of both spectral clustering and DBSCAN; the combination can overcome some of the innate disadvantages of both individual methods.

For all their strengths, even the most advanced clustering methods nowadays still can be tripped up by some pathological cases: datasets where the human observer immediately sees what is going on, but which prove to remain tricky for all state-of-the-art clustering algorithms. One such example is the dataset illustrated in Figure 4.1: we will refer to it as the *two circles* dataset. It consists of two noisily[1] separated concentric circles, each encompassing the same number

[1]the scikit-learn clustering documentation (cf. `https://scikit-learn.org/stable/modules/clustering.html`) shows how SC and DBSCAN can succeed on this dataset, but that version contains barely any noise (scikit's noise parameter set to 0.05, instead of our still benign 0.1).

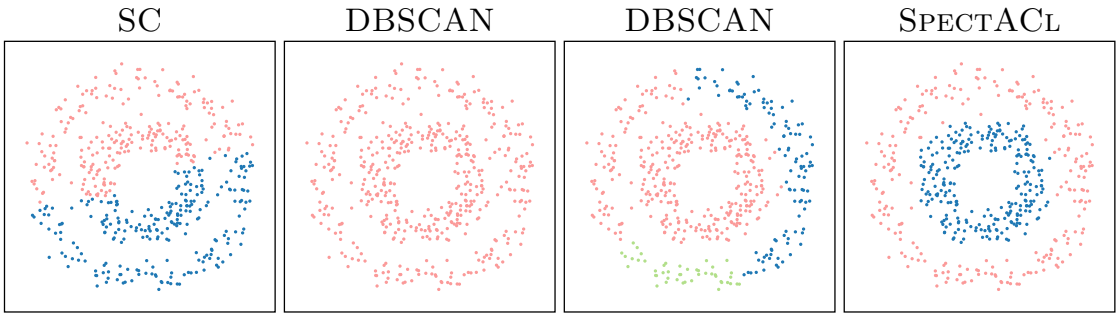

FIGURE 4.1: Performance of spectral clustering, DBSCAN, and SpectACl on two concentric circles. Best viewed in color.

of observations. As the leftmost plot in Figure 4.1 shows, spectral clustering does not at all uncover the innate structure of the data: two clusters are discovered, but both incorporate about half of each circle. It is well-known that spectral clustering is highly sensitive to noise (Bojchevski et al., 2017, Figure 1); if two densely connected clusters are additionally connected to each other via a narrow bridge of only a few observations, spectral clustering runs the risk of reporting these clusters plus the bridge as a single cluster, whereas two clusters plus a few noise observations (outliers) would be the desired outcome. The middle plots in Figure 4.1 shows how DBSCAN (with *minpts* set to 25 and 26, respectively) fails to realize that there are two clusters. This is hardly surprising, since DBSCAN is known to struggle with several clusters of varying density, and that is exactly what we are dealing with here: since both circles consist of the same number of observations, the inner circle is substantially more dense than the outer circle. The rightmost plot in Figure 4.1 displays the result of the new clustering method that we introduce in this chapter, SpectACl (*Spectral Averagely-dense Clustering*): it accurately delivers the clustering that represents the underlying phenomena in the dataset. Our main contributions are:

- we propose a new clustering method SpectACl, combining the benefits of spectral clustering and DBSCAN, while alleviating some of the innate disadvantages of each individual method,
- unlike spectral clustering, the discretization step applying k-means to the spectral embedding is theoretically sound in SpectACl: from SpectACl's objective function we derive an upper bound by means of the eigenvector decomposition and derive that the optimization of our upper bound is equal to k-means on the eigenvectors,
- we show that pursuing a spectral relaxation for a density-based approach yields interpretable eigenvectors, displaying the relevant parts of clusters.

4.1 Spectral Clustering and the Robustness Issue

Beyond theoretical concerns, an often reported issue with spectral clustering results is the noise sensitivity. This might be attributed to its relation with the minimum cut paradigm, where few edges suffice to let two clusters appear as one.

The state-of-the-art advances in spectral clustering follow two approaches aiming at increased robustness towards noise. The one is to incorporate ideas from density-based clustering, such as requiring that every node in a cluster has a minimal node degree (Bojchevski et al., 2017). Minimizing the cut subject to the minimum degree constraint resembles finding a connected component of core points, as performed by DBSCAN. The drawback of this approach is that it introduces a new parameter to spectral clustering, which is (comparably to the $minPts$ parameter in DBSCAN) influencing the quality of the result. The other is to learn the graph representation simultaneously with the clustering (Bojchevski et al., 2017; Kang et al., 2018; Nie et al., 2017). This is generally achieved by alternating updates of the clustering and the graph representation, requiring the computation of a truncated eigendecomposition in every iteration step. This results in higher computational costs. Of these newer methods, we compare our method against the robust spectral clustering algorithm by Bojchevski et al. (2017), since it incorporates both the notion of density and the learning of the graph representation.

4.2 Spectral Averagely-Dense Clustering

We propose a cluster definition based on the average density (i.e., node degree) in the subgraph induced by the cluster. This graph-based clustering objective is equal to the objective of kernel k-means (cf. Section 2.3.2) where the kernel matrix is the weighted adjacency matrix W. We assume for now that W is the adjacency matrix to the ϵ-neighborhood graph. We strive to solve the following problem, maximizing the *average cluster density*:

$$\max_{Y \in \mathbb{1}^{m \times r}} \operatorname{tr}\left(Y^\top W Y \left(Y^\top Y\right)^{-1}\right) = \sum_{s=1}^{r} R(Y_{\cdot s}, W). \tag{4.1}$$

The objective function returns the sum of average node degrees $R(Y_{\cdot s}, W)$ in the subgraph induced by cluster s, i.e.:

$$R(Y_{\cdot s}, W) = \frac{Y_{\cdot s}^\top W Y_{\cdot s}}{\|Y_{\cdot s}\|^2} = \frac{1}{|Y_{\cdot s}|} \sum_{j: Y_{js}=1} W_{j\cdot} Y_{\cdot s}. \tag{4.2}$$

The last term $W_{j\cdot}Y_{\cdot s}$ computes the degree of node j if only connections within cluster s count. Finding the clustering which maximizes the density, i.e., the node degree, in each cluster corresponds to the definition of clusters in DBSCAN. However, Eq. (4.1) also provides a connection to the minimum ratio cut objective if the matrix W is normalized. In this case, the Laplacian is given as $L = I - \tilde{W}$, and subtracting the identity matrix from Eq. (4.1) does not change the objective.

We discuss a new spectral relaxation of Eq. (4.1) as foreshadowed in our discussion about symmetric decompositions of kernel matrices and graph Laplacians (cf. Theorem 2.2 and 2.3). Therefore, we analyze the application of k-means to the (truncated) eigendecomposition.

4.2.1 Dense Clusters and Projected Eigenvectors

The function R from Eq. (4.2) is also known as the Rayleigh quotient. The values of the Rayleigh quotient depend on spectral properties of the applied matrix. As such, the density $R(y, W) \in [\lambda_m, \lambda_1]$ is bounded by the smallest and largest eigenvalues of the matrix W (Collatz, 1978). The extremal densities are attained at the corresponding eigenvectors. The eigenvectors $V_{\cdot 1}, \ldots, V_{\cdot d}$ to the d-largest eigenvalues span a space whose points y have a minimum density of

$$R(y, W) = \frac{y^\top W y}{\|y\|^2} = \frac{\left(\sum_k \alpha_k V_{\cdot k}^\top\right) W \left(\sum_k \alpha_k V_{\cdot k}\right)}{\left\|\sum_k \alpha_k V_{\cdot k}\right\|^2} = \frac{\sum_k \alpha_k^2 \lambda_k \|V_{\cdot k}\|^2}{\sum_k \alpha_k^2 \|V_{\cdot k}\|^2} \geq \lambda_d.$$

Here, we employ the fact that any point y in the span of the first eigenvectors has a representation as linear combination of these eigenvectors and we employ the orthogonality of the eigenvectors. As a result, the projection of W onto the subspace spanned by the first eigenvectors reduces the dimensionality of the space in which we have to search for optimal clusters, having a minimum density of λ_d.

This insight suggests a naive approach, where we approximate W by its truncated eigendecomposition. We restate in this case the optimization problem to find averagely dense clusters from Eq. (4.1) as

$$\max_{Y \in \mathbb{1}^{m \times r}} \sum_s \frac{Y_{\cdot s}^\top V \Lambda V^\top Y_{\cdot s}}{|Y_{\cdot s}|}. \tag{4.3}$$

A comparison with the k-means objective from Eq. (2.10) shows that the objective above is a trace maximization problem which is equivalent to k-means clustering on the data matrix $U = V\Lambda^{1/2}$.

Unfortunately, applying k-means to the spectral embedding U does not yield acceptable clusterings. The objective of k-means is nonconvex, having multiple local solutions. The dimensionality of the search space increases with every eigenvector which we include in the (truncated) eigendecomposition from W, due to the orthogonality of eigenvectors. As a result, even sophisticated initialization methods can not prevent that k-means returns clusterings whose objective function value approximates the global minimum, but which reflect seemingly haphazard groupings of the data points.

The eigenvectors are not only orthogonal, but also real-valued, having positive as well as negative values. The first eigenvectors have a high value of the density function R, but the mixture of signs in its entries makes an interpretation as cluster indicators difficult. Nonnegative eigenvectors would have an interpretation as fuzzy cluster indicators, where the magnitude of the entry V_{jk} indicates the degree to which point j is assigned to the fuzzy cluster k. Applying k-means to a set of fuzzy cluster indicators would furthermore reduce the space in which we search for possible cluster centers to the positive quadrant. One possibility is to approximate the matrix W by a symmetric nonnegative matrix factorization, but this replaces the polynomially solvable eigendecomposition with an NP-hard problem (Vavasis, 2009). We conduct another approach, showing that fuzzy cluster indicators are derivable from the eigenvalues by a simple projection.

OBSERVATION 4.1. *Let W be a symmetric real-valued matrix, and let v be an eigenvector to the eigenvalue λ. Let $v = v^+ - v^-$, with $v^+, v^- \in \mathbb{R}^m_+$ be the decomposition of v into its positive and negative parts. The nonnegative vector $u = v^+ + v^-$ has a density*

$$R(u) \geq |\lambda|.$$

Proof. An application of the triangle inequality shows that the entries of an eigenvector having a high absolute value, play an important role for achieving a high Rayleigh quotient. Let v be an eigenvector to the eigenvalue λ of W, then:

$$|\lambda||v_j| = |W_{j\cdot}v| = \left|\sum_l W_{jl}v_l\right| \leq \sum_l W_{jl}|v_l|. \tag{4.4}$$

Since $u_j = |v_j|$, from the inequality above follows that $Wu \geq |\lambda|u$. Multiplying the vector $u^\top$ from the left and dividing by $\|u\|^2$ yields the stated result. □

We refer to the eigenvectors, whose entries are replaced with their absolute values, i.e., $u_j = |v_j|$ for $1 \leq j \leq m$, as *projected eigenvectors*. The projected eigenvectors have locally similar values, resulting in the gradually changing shapes, illustrated on the two circles dataset in Figure 4.2. We visualize how projected eigenvectors provide fuzzy cluster indicators: a strongly indicated dense part of one

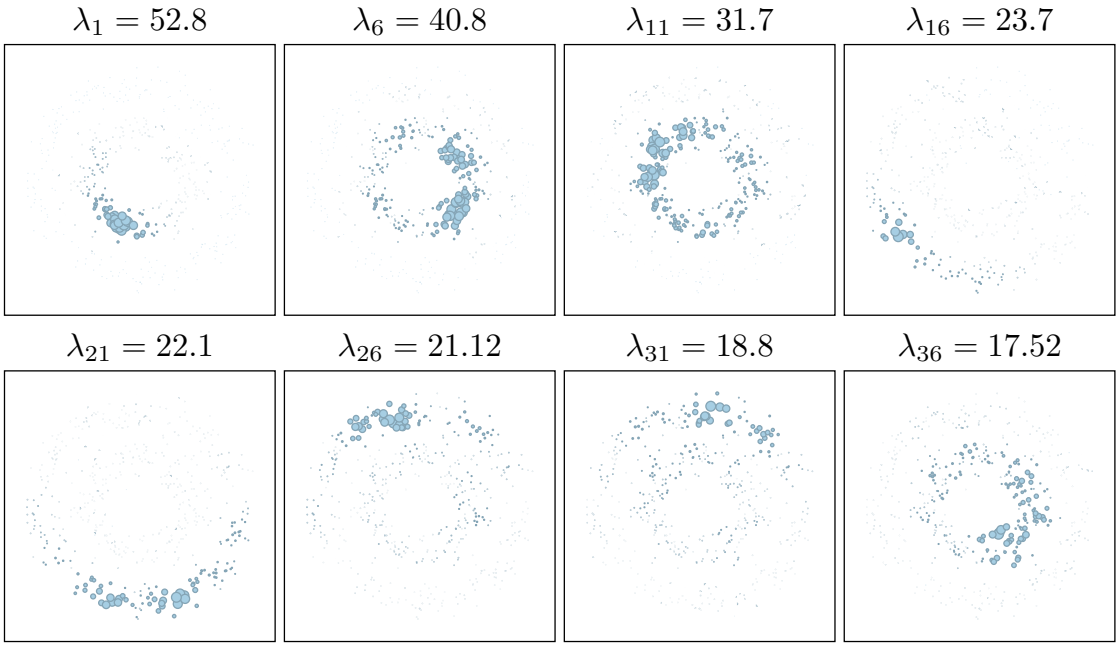

FIGURE 4.2: Visualization of every fifth eigenvector for the two circles dataset. The size of each point is proportional to the absolute value of the corresponding entry in the eigenvector.

ALGORITHM 6 Spectral Averagely Dense Clustering

FUNCTION SpectACl(W, r) ▷ W: adjacency matrix
  $(V, \Lambda) \leftarrow \arg\min_{V,\Lambda} \left\| W - V\Lambda V^\top \right\|^2$ s.t. $V \in \mathbb{R}^{m\times d}, \Lambda = \mathrm{diag}(\lambda), \lambda \in \mathbb{R}^d$
  $U_{jk} \leftarrow |V_{jk}|\sqrt{|\lambda_k|}$ ▷ projected embedding
  $(X, Y) \leftarrow \arg\min_{X,Y} \left\| U - YX^\top \right\|^2$ s.t. $Y \in \mathbb{1}^{m\times r}, X \in \mathbb{R}^{n\times r}$ ▷ k-means
  RETURN Y
END FUNCTION

of the two clusters fades out along the circle trajectory. Furthermore, we see that the projected eigenvectors are not orthogonal, since some regions are repeatedly indicated over multiple eigenvectors. These varying views on possible dense and fuzzy clusters are beneficial in order to robustly identify the clustering structure.

Applying k-means to the first d projected eigenvectors yields a set of r binary basis vectors, which indicate clusters in the subspace of points with an average density larger than λ_d. We summarize the resulting method SpectACl (Spectral Averagely-dense Clustering) in Algorithm 6. The algorithm performs a minimization of an upper bound on the average cluster density function with respect to

W if $d = m$. In the general case $d \ll m$, we optimize an approximation of the objective function, replacing W with the nonnegative product $UU^\top$. Certainly, a similar argument could be made for the pipeline of spectral clustering, considering the approximate objective (4.3) for $d = r$. However, the difference is that the result of spectral clustering deteriorates with an increasing accuracy of the eigendecomposition approximation ($d \to m$) due to orthogonality and negative entries in the eigenvectors as outlined above. We show in our experimental evaluation that the result of our method does in average not deteriorate with an increasing dimension of the embedding.

In the default setting of SpectACl, we determine W as adjacency matrix to the ϵ-neighborhood graph. However, our method is in principle applicable to any provided (weighted) adjacency matrix. We recall that the average density objective is equivalent to the ratio cut objective if we normalize the matrix W. We refer to this version as normalized SpectACl. In the normalized case, we compute the adjacency matrix according to the k-nearest neighbor graph, which is suggested for applications of spectral clustering.

Hence, our method only depends on the parameters r and ϵ, respectively k. How to determine the number of clusters, which is a general problem for clustering tasks, is beyond scope of this work. We do provide a heuristic to determine ϵ and we evaluate the sensitivity to the parameters ϵ and k.

4.3 Experiments

We conduct experiments on a series of synthetic datasets, exploring the ability to detect the ground truth clustering in the presence of noise. On real-world data, we compare the Normalized Mutual Information (NMI) of obtained models with respect to provided classes. A small qualitative evaluation is given by means of a handwriting dataset, illustrating the clusters obtained by SpectACl. Our Python implementation, and the data generating and evaluation script, are publicly available[2].

4.3.1 Experiments on Synthetic Data

We generate benchmark datasets, using the renowned scikit library. For each shape —moons, circles, and blobs— and noise specification we generate $m = 1500$ data points. The noise is Gaussian, as provided by the scikit noise parameter; cf. `http://scikit-learn.org`. Our experiments pit five competitors against each other: two new ones presented in this chapter, and three baselines. We

[2] `https://sfb876.tu-dortmund.de/spectacl`

compare our method SpectACl, using the ϵ-neighborhood graph represented by W (denoted SpectACl) and the symmetrically normalized adjacency matrix $\operatorname{diag}(W_{knn}\mathbf{1})^{-1/2} W_{knn} \operatorname{diag}(W_{knn}\mathbf{1})^{-1/2}$ where W_{knn} is the adjacency matrix to the k-nearest neighbor graph (denoted SpectACl (Normalized), or (N) as shorthand). The parameter ϵ is determined as the smallest radius such that 99% of the points have at least ten neighbors and the number of nearest neighbors is set to $k = 10$. We compare against the scikit implementations of DBSCAN (denoted DBSCAN) and symmetrically normalized spectral clustering (denoted SC). For DBSCAN, we use the same ϵ as for SpectACl and set the parameter $minPts = 10$, which delivered the best performance on average. We also compare against the provided Python implementation of robust spectral clustering (Bojchevski et al., 2017) (denoted RSC), where default values apply. Unless mentioned otherwise, our setting for the embedding dimensionality is $d = 50$.

EVALUATION We assess the quality of computed clusterings by means of the ground truth. Given a clustering $Y \in \{0,1\}^{m \times r}$ and the ground truth model $Y^* \in \{0,1\}^{m \times r}$, we evaluate the clustering by means of an adaptation of the micro-averaged F-measure, known from multi-class classification tasks. We compute a bijection of matching clusters σ with the Hungarian algorithm (Kuhn, 1955), maximizing $\mathsf{F} = \sum_s \mathsf{F}(s, \sigma(s))$, where

$$\mathsf{F}(s,t) = 2\frac{\operatorname{pre}(s,t)\operatorname{rec}(s,t)}{\operatorname{pre}(s,t) + \operatorname{rec}(s,t)},$$

$$\operatorname{pre}(s,t) = \frac{Y_{\cdot s}^\top Y_{\cdot t}^*}{|Y_{\cdot s}|} \quad \text{and} \quad \operatorname{rec}(s,t) = \frac{Y_{\cdot s}^\top Y_{\cdot t}^*}{|Y_{\cdot t}^*|}.$$

These functions denote precision and recall, respectively. Based on the perfect matching σ, we calculate precision and recall for the obtained factorization by

$$\operatorname{pre} = \frac{\sum_{s=1}^r |Y_{\cdot s}^* \circ Y_{\cdot \sigma(s)}|}{\sum_{s=1}^r |Y_{\cdot s}|} = \frac{|Y^* \circ Y_{\cdot \sigma(\cdot)}|}{|Y|}$$
$$\operatorname{rec} = \frac{\sum_{s=1}^r |Y_{\cdot s}^* \circ Y_{\cdot \sigma(s)}|}{\sum_{s=1}^r |Y_{\cdot s}^*|} = \frac{|Y^* \circ Y_{\cdot \sigma(\cdot)}|}{|Y^*|}.$$

The micro F-measure is then defined in terms of precision and recall as above. This is equivalent to a convex combination of the $\mathsf{F}(s, \sigma(s))$-measurements:

$$\mathsf{F} = 2\frac{\operatorname{pre} \cdot \operatorname{rec}}{\operatorname{pre} + \operatorname{rec}} = \sum_{s=1}^r \frac{|Y_{\cdot s}^*| + \left|Y_{\cdot \sigma(s)}\right|}{|Y^*| + |Y|} \mathsf{F}(s, \sigma(s)).$$

The F-measure takes values in $[0,1]$; the closer to one, the more similar are the computed clusterings to the ground truth.

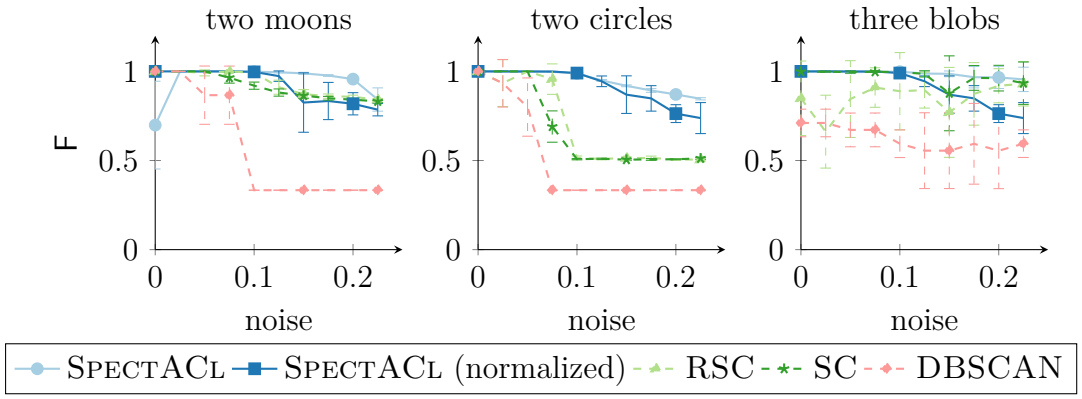

FIGURE 4.3: Variation of noise, comparison of F-measures (the higher the better) for the two moons (left), the two circles (middle) and three blobs (right) datasets.

NOISE SENSITIVITY In this series of experiments, we evaluate the robustness of considered algorithms against noise. For every noise and shape specification, we generate five datasets. A visualization of the clusterings for three generated datasets is given in Figure 4.4, setting the noise parameter to 0.1. Note that the depicted plot of the three blobs dataset shows a rare but interesting case, where two of the clusters are overlapping. In Figure 4.3 we plot for every considered algorithm the averaged F-measures against the noise.

From Figure 4.3 we observe that SpectACl typically attains the highest F-value, showing also a very low variance in the results. The sole exception is the case of the two moons dataset when no noise is added (leftmost figure). A closer inspection of the embedding in this case shows that SpectACl is tripped up by the absolute symmetry in the synthetic data. Since this is unlikely to occur in the real world, we do not further elaborate on this special case. The normalized version of SpectACl does not exhibit the artifact of symmetry, yet its results are less accurate when the noise is high (> 0.1). Particularly interesting is the comparison of normalized SpectACl with SC, since both methods employ the same weighted adjacency matrix. We observe that both methods have a similar F-measure for the two moons as well as the three blobs data when the noise is at most 0.15. For the three blobs data and a higher noise level, normalized SpectACl has a lower F-value than SC. However, remarkable is the difference at the two circles dataset, where normalized SpectACl attains notably higher F-values than SC. Figure 4.4 illustrates that normalized SpectACl detects the trajectory of the two circles while SC cuts both circles in half.

Comparing these results with Robust Spectral Clustering (RSC), we conclude that learning the graph structure together with the spectral clustering does not drastically affect the result. Given the two moons or the two circles dataset, the

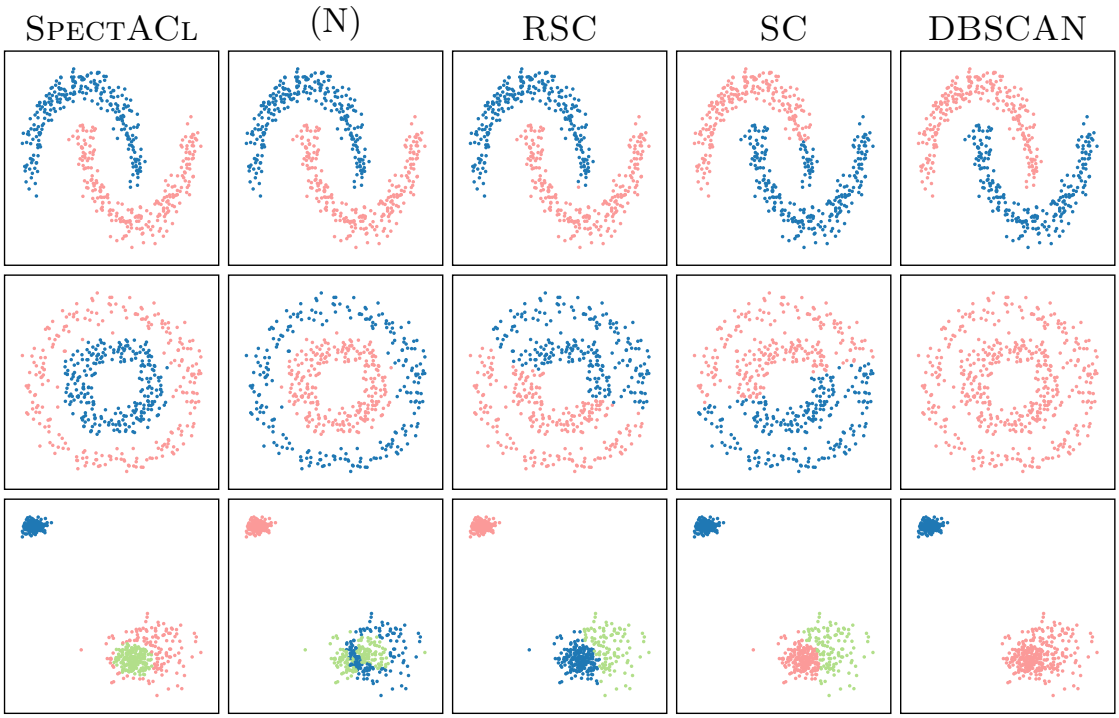

FIGURE 4.4: Cluster visualizations of regarded algorithms and synthetic datasets. Best viewed in color.

F-measure of RSC is close to that of SC. The example clusterings in the top row of Figure 4.4 show that RSC is able to correct the vanilla spectral clustering, where a chunk from the wrong moon is added to the blue cluster. Yet, in the case of the two circles with different densities, both methods fail to recognize the circles (cf. Figure 4.4, middle row). Surprisingly, on the easiest-to-cluster dataset of the three blobs, RSC attains notably lower F-measures than spectral clustering.

The plots for DBSCAN present the difficulty to determine a suitable setting of parameters ϵ and $minpts$. The method to determine these parameters is suitable if the noise is low. However, beyond a threshold, DBSCAN decides to pack all points into one cluster. We also adapted the heuristic for the parameter selection, determining ϵ as the minimum radius such that $70 - 90\%$ of all points have at least $minPts = 10$ neighbors. This does not fundamentally solve the problem: it merely relocates it to another noise threshold.

PARAMETER SENSITIVITY We present effects of the parameter settings for SpectACl, namely the embedding dimension d, the neighborhood radius ϵ and the number of nearest neighbors k. We summarize the depiction of these experiments in Figure 4.5. The plots on the top row show that the results of SpectACl

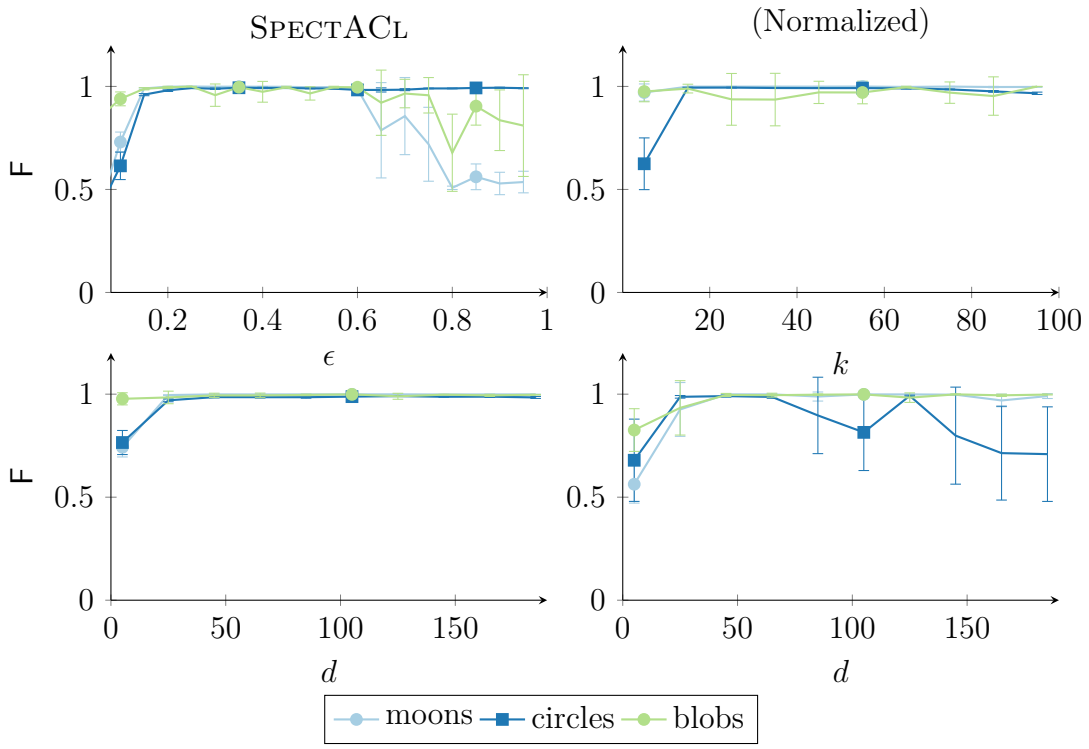

FIGURE 4.5: Variation of latent parameters used for SpectACl. The neighborhood radius ϵ (top left) and the number of nearest neighbors k (top right) determine the adjacency matrix for SpectACl and its normalized version. The parameter d (bottom left and right) specifies the number of projected eigenvectors. We plot the F-measure (the higher the better) against the variation of the parameter.

are robust to the parameters determining the adjacency matrix. The fluctuations of the F-measure when $\epsilon > 0.6$ are understandable, since the coordinates of points are limited to $[0, 3]$. That is, $\epsilon = 0.6$ roughly equates the diameter of the smaller blob on the top left in Figure 4.4 and is quite large. The plots on the bottom of Figure 4.5 indicate that SpectACl in the unnormalized and normalized version generally only require the dimension of the embedding to be large enough. In this case, setting $d > 25$ is favorable. The only exception is the two circles dataset when using normalized SpectACl, where the F-measure starts to fluctuate if $d > 75$. A possible explanation for this behavior is that eigenvalues in the normalized case lie between zero and one. When we inspect the first 50 projected eigenvectors of the normalized adjacency matrix, then we see that the last 19 eigenvectors do not reflect any sensible cluster structure, having eigenvalues in

Data	m	n	r	$\mu\left(\frac{\mathcal{N}_\epsilon(j)}{m}\right)$
Pulsar	17 898	9	2	0.11 ± 0.08
Sloan	10 000	16	3	0.09 ± 0.07
MNIST	60 000	784	10	0.05 ± 0.06
SNAP	1 005	–	42	0.05 ± 0.06

TABLE 4.1: Real-world dataset characteristics; number of samples m, features n, clusters (classes) r and relative average number of neighbors in the ϵ-neighborhood graph μ.

Algorithm	Pulsar	Sloan	MNIST	SNAP
SpectACl	**0.151**	0.224	0.339	0.232
SpectACl(N)	0.005	0.071	**0.756**	0.104
DBSCAN	0.001	**0.320**	0.005	–
SC	0.025	0.098	0.753	**0.240**
RSC	0.026	0.027	0.740	–

TABLE 4.2: Normalized Mutual Information (NMI) score (the higher the better) for real-world datasets.

$[0.4, 0.54]$. In unnormalized SpectACl, the unhelpful eigenvectors have a very low density and thus also a very low eigenvalue, being barely taken into account during the optimization.

4.3.2 Real-World Data Experiments

We conduct experiments on selected real-world datasets, whose characteristics are summarized in Table 4.1. The *Pulsar* dataset[3] contains samples of Pulsar candidates, where the positive class of real Pulsar examples poses a minority against noise effects. The *Sloan* dataset[4] comprises measurements of the Sloan Digital Sky Survey, where every observation belongs either to a star, a galaxy or a quasar. The *MNIST* dataset (LeCun et al., 1998) is a well-known collection of handwritten ciphers. The *SNAP* dataset refers to the Email EU core network data (Leskovec and Krevl, 2014), which consists of an adjacency matrix (hence this dataset has no features in the traditional sense). For this dataset, we only compare the two versions of SpectACl and spectral clustering, since robust spectral clustering and DBSCAN do not support a direct specification of the adjacency matrix. Table 4.2

[3] `https://www.kaggle.com/pavanraj159/predicting-pulsar-star-in-the-universe`
[4] `https://www.kaggle.com/lucidlenn/sloan-digital-sky-survey`

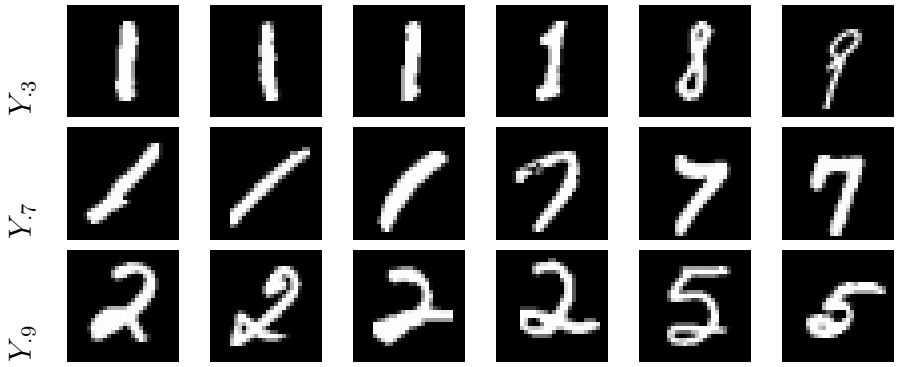

FIGURE 4.6: SpectACl clusters individual handwriting styles instead of cipher shapes.

summarizes the normalized mutual information between the found clustering and the clustering which is defined by the classification. We observe that the unnormalized version of SpectACl obtains the highest NMI on the pulsar dataset, the second-highest on the Sloan dataset, a significantly lower NMI than RSC, SC and the normalized version on the MNIST sample and a similar NMI on the SNAP dataset. The Sloan and MNIST datasets pose interesting cases. The notoriously underperforming DBSCAN obtains the highest NMI at the Sloan dataset while obtaining the lowest F-measure of 0.09, where SpectACl obtains the highest F-measure of 0.52 (equating the class assignments with the ground truth). The datasets Pulsar and Sloan are clustered in accordance with the classes when using the density-based approach, yet the MNIST clustering corresponds more with the clustering when using the minimum-cut associated approaches.

To understand why SpectACl fails to return clusters in accordance with the MNIST classes, we display some exemplary points for three SpectACl clusters in Figure 4.6. On the one hand, some identified clusters indeed contain only one cipher, e.g., there are clusters for zeros, ones and sixes exclusively. On the other hand, the clusters depicted in the figure contain a mixture of digits, where the digits are written in a cursive style or rather straight. Hence, instead of identifying the numbers, SpectACl identifies clusters of handwriting style, which is interesting information discovered from the dataset, albeit not the information the MNIST task asks for. To be fair, it is far from impossible that a similar rationale can be given for the clusters SpectACl's competitors find on the Pulsar or Sloan dataset; we lack the astrophysical knowledge required to similarly assess those results.

4.4 Discussion

We introduce SpectACl (*Spectral Averagely-dense Clustering*): a new clustering method that combines benefits from spectral clustering and the density-based DBSCAN algorithm, while avoiding some of their drawbacks.

By computing the spectrum of the weighted adjacency matrix, SpectACl automatically determines the appropriate density for each cluster. This eliminates the specification of the *minpts* parameter which is required in DBSCAN, and as we have seen in Figure 4.1, this specification is a serious hurdle for a DBSCAN user to overcome. On two concentric circles with the same number of observations (and hence with different densities), a DBSCAN run with $minpts = 25$ lumps all observations into one big cluster. When increasing *minpts* by one, DBSCAN jumps to four clusters, none of which are appropriate. SpectACl, conversely, can natively handle these nonconvex clusters with varying densities.

In both spectral clustering and SpectACl, the final step is to postprocess intermediate results with k-means. Whether this choice is appropriate for spectral clustering remains open to speculation. However, from the objective function of SpectACl, we derive an upper bound through the eigenvector decomposition, whose optimization we show to be equal to k-means optimization. Hence, for SpectACl, we demonstrate that this choice for k-means postprocessing is mathematically fundamentally sound.

In comparative experiments, competing with DBSCAN, spectral clustering, and robust spectral clustering, we find on synthetic data that the unnormalized version of SpectACl is the most robust to noise (cf. Figure 4.3), which identifies the appropriate cluster structure in three scenarios while the competitors all fail to do so at least once (cf. Figure 4.4). On real-life data, SpectACl outperforms the competition on the inherently noisy Pulsar dataset and the Sloan dataset, it performs similarly to spectral clustering on the SNAP dataset, and it is defeated by both spectral clustering and robust spectral clustering on the MNIST dataset (cf. Table 4.2). The latter observation is explained when looking at the resulting clusters: as Figure 4.6 illustrates, SpectACl focuses on clusters representing handwriting style rather than clusters representing ciphers, which is not unreasonable behavior for an unsupervised method such as clustering; this uncovered information is merely not reflected in the NMI scores displayed in the table.

Figure 4.5 provides evidence that SpectACl is robust with respect to its parameter settings. Hence, in addition to its solid foundation in theory and good empirical performance on data, SpectACl provides a clustering method that is easy to use in practice. Thus, SpectACl is an ideal candidate for outreach beyond data mining experts.

CHAPTER 5

Proximal Optimization for Boolean Matrix Factorization

In a large range of exploratory data mining tasks such as Market Basket Analysis, Text Mining, Collaborative Filtering or DNA Expression Analysis, the data is represented by a binary matrix. In this case, a binary representation of the cluster centers is desirable to reflect the binary nature of features. Methods which provide interpretable models in that sense are discussed from the viewpoint of binary matrix factorization, pattern mining, tiling and Boolean matrix factorization (Tatti and Vreeken, 2012; Zimek and Vreeken, 2015). Of those fields, Boolean matrix factorization is the most flexible one. Boolean factorizations are not required to model partitioning cluster assignments, such as binary matrix factorization and it does not restrict the cluster assignment to the support of a pattern, as is the case in tiling and pattern mining.

Therefore, providing a generic optimization method, solving the problem of BMF, has a possible impact on the optimization of all the regarded cluster objectives in Chapter 2. We have seen that the optimization of non-partitioning clusters is by and large an unsolved problem. Approaches tackling this issue require detailed specifications of parameters which are unknown in advance, such as the amount of overlap and outliers (Whang and Dhillon, 2017). In contrast, we aim at formalizing an optimization scheme which is able to derive the *true* model, regardless of the characteristics of that model; may it be overlapping or not, may it have a large amount of overlap or not.

Unfortunately, present BMF optimizations employ a greedy heuristic, which can not provide guarantees as known from numerical optimization methods, such as convergence to a local optimum. We make use of an example to show the innate disadvantage, emerging from a strong preference bias of the greedy approach in Figure 5.1. Here, a picture is transformed into a binary data matrix via its binary

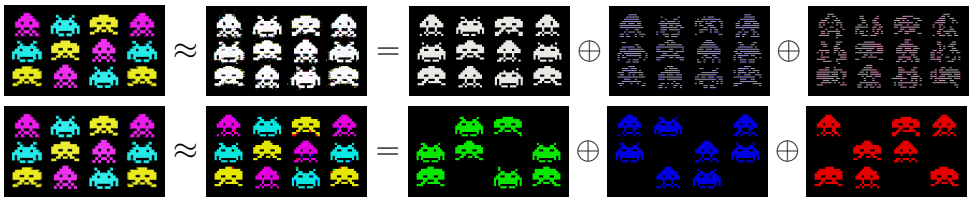

FIGURE 5.1: Results of a Boolean decomposition of rank three on a picture dataset, depicted on the left. On the top row, the picture is approximated by a greedy approach, on the bottom we display the result of the proposed proximal optimization scheme for BMF. Best viewed in color.

representation in RGB888 pixel format. On the top row, you see the decomposition obtained by the greedy approach PANDA+ (Lucchese et al., 2014) and on the bottom the approximation derived by the new method which we introduce in this chapter. Each of the three rightmost pictures visualizes an outer product, also called tile in the context of binary factorization. The difference between both results is striking, in particular because the color information gets completely lost in the greedy approximation. We observe that the greedy representation in black and white is attributed to the first outer product being overloaded with the information about shape. Since the color white has a binary representation as a constant one vector, any white pixel in one outer product picture also appears white in the resulting approximation. Accordingly, the greedy approach is very sensitive to making mistakes in the first derived tiles and is therewith also susceptible to noise.

In contrast, we propose a simultaneous optimization of the clusters based on a nonnegative relaxation with binary penalization. This approach is similar to the binary factorization by Zhang et al. (2007), employing the mexican hat function for the penalization of nonbinary values (cf. Section 3.5). However, our method follows latest results in nonconvex optimization theory and provides a general framework for the efficient optimization of matrix factorization under binary constraints. The main contribution of this chapter is the derivation of the proximal mapping, which implements the nonbinary penalization. This proximal mapping is usable as a building block to incorporate binary restrictions on one of the matrices in factorization objectives. Known proximal mappings involving other constraints such as nonnegativity, sparseness or the restriction to the probability simplex can then be integrated within the larger optimization framework.

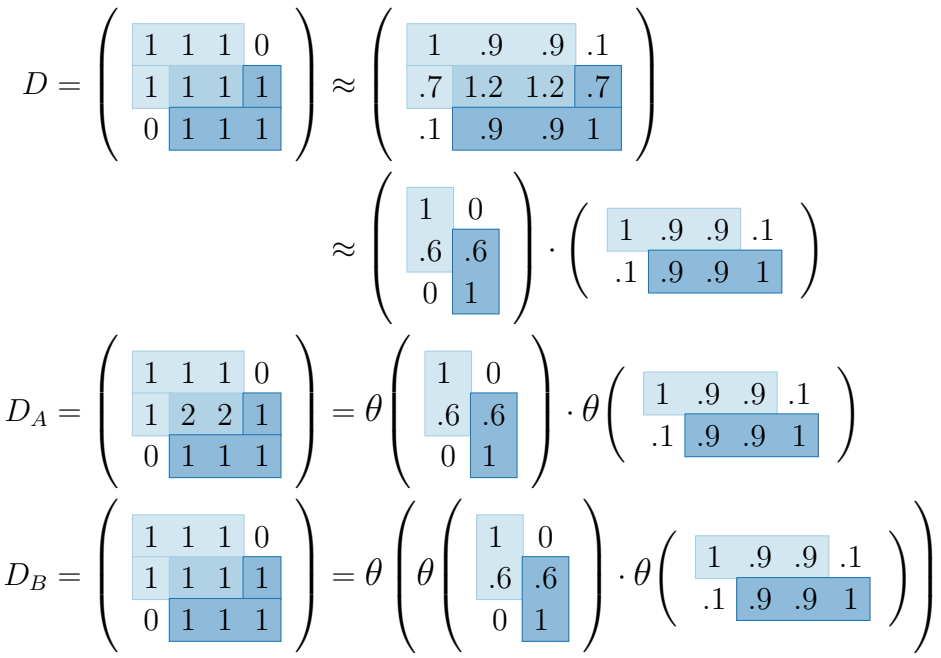

FIGURE 5.2: Approximation of a binary matrix D with two overlapping tiles (top) applying NMF (second from above) and the factorizations resulting from thresholding the factor matrices to binary matrices in elementary algebra (second from below) and Boolean algebra (below). Tiles are highlighted.

5.1 Approximating the Boolean Product in Elementary Algebra

A possible reason for the prevailing usage of heuristics in Boolean matrix factorization is the reasonable belief that relaxations to nonnegative or other continuous values are not apt to approximate a product in Boolean arithmetic. Contrary to this belief, we argue for the opposite; a nonnegative relaxation is particularly suited to derive overlapping clusterings and is therefore also suited to approximate Boolean matrix factorizations, whose main characteristic is to allow for overlap between the clusters. Now we need to be a bit careful with the word *approximate*. The BMF problem is NP-hard and NP-hard to aproximate within a constant factor (Miettinen et al., 2008). Hence, we will not be able to produce an efficient algorithm which comes arbitrarily close to the optimal Boolean solution (unless NP=P). Yet we are able to simulate some of the required characteristics of Boolean solutions in a relaxed space.

To make our case, we inspect how nonnegative and Boolean matrix factorizations deal with overlapping clusters. An example binary data matrix consisting of two overlapping tiles and its approximation by a nonnegative matrix factorization is shown in the top two equations of Figure 5.2. We see that the factors contain values smaller than one at entries which are involved in overlapping parts. With this, overlapping sections are equally well approximated as non-overlapping components. The matrices D_A and D_B in Figure 5.2 show the resulting binary and Boolean approximations, thresholding the nonnegative factor matrices at one half. We find that the reconstruction error is largest when the thresholded matrices are multiplied in elementary algebra, as it is the case in binary matrix factorization. In contrast, the fuzzy cluster indication by NMF is suited to indicate a definite clustering with respect to the Boolean algebra. This seems at first contradictory to the discussed near-orthogonality of NMF solutions in Section 2.2.1. However, although NMF has an in-built penalization of nonorthogonal factor matrices (cf. Eq. (2.1)), the approximation error is the dominating value of the objective function. Accordingly, the better the factorization approximates the data, implying the higher the rank, the more become solutions of NMF orthogonal. Conversely, we conclude that overlap between clusters is mirrored by a nonnegative approximation if the rank is low.

5.2 Proximal Alternating Linearized Minimization

Standard gradient descent-based optimization schemes for NMF problems are either multiplicative updates or projected gradient methods. The crucial aspect of both optimization schemes is the determination of the stepsize. The optimization with multiplicative updates is slow due to the conservative choice of the stepsize, ensuring that factor matrices are nonnegative. Projected gradient methods often employ a linesearch procedure to determine the optimal stepsize, but calculating the optimal stepsize in every step is also costly.

Bolte et al. (2014) extend optimization results known for convex optimization to the nonconvex case with the *Proximal Alternating Linearized Minimization* (PALM). This technique focuses on objectives breaking down into a smooth part F and a possibly nonsmooth component ϕ

$$\min_{X,Y} F(X,Y) + \phi_1(X) + \phi_2(Y) \qquad \text{s.t. } X \in \mathbb{R}^{n\times r}, Y \in \mathbb{R}^{m\times r}. \tag{5.1}$$

The function F reflects here the objective function or its smooth relaxation. We assume for now that $F(X,Y) = \left\|D - YX^\top\right\|^2$ returns the approximation error in the Frobenius norm. The nonsmooth part ϕ may return ∞, which can be used to model restrictions of the search space, e.g., the nonnegativity constraint of

NMF. PALM performes an alternating minimization on the linearized objective, substituting F with its first order Taylor approximation. This is achieved by alternating *proximal mappings* from the gradient descent update with respect to F, i.e., the following steps are repeated for $1 \leq k \leq K$:

$$X_{k+1} = \text{prox}_{\alpha_k \phi_1}(X_k - \alpha_k \nabla_X F(X_k, Y_k)); \tag{5.2}$$
$$Y_{k+1} = \text{prox}_{\beta_k \phi_2}(Y_k - \beta_k \nabla_Y F(X_{k+1}, Y_k)). \tag{5.3}$$

The proximal mapping of a function ϕ, is a function which returns a matrix satisfying the following minimization criterion:

$$\text{prox}_{\phi}(X) \in \arg\min_{X^*} \left\{ \frac{1}{2}\|X - X^*\|^2 + \phi(X^*) \right\}. \tag{5.4}$$

Loosely speaking, the proximal mapping gives its argument a little push into a direction which minimizes ϕ. For a detailed discussion, see, e.g., (Parikh et al., 2014). As we can see in Eqs. (5.2) and (5.3), the evaluation of this operator is a base operation. Similarly to the alternating minimization in Eqs. (2.2) and (2.3), finding the minimum of the proximal mapping in every iteration by numerical methods is infeasible in practice. Thus, the trick is to use only simple functions ϕ for which the proximal mapping can be calculated in a closed form.

5.2.1 Convergence Results

We state here the main convergence result for PALM in a specified version, requiring that the differentiable part of the objective is smooth and not only continuously differentiable, summarizing the discussion with respect to smooth functions in (Bolte et al., 2014). Yet first, we need to introduce some definitions.

DEFINITION 5.1 (PROPER, SEMI-CONTINUOUS FUNCTION). Let $f : \mathbb{R}^n \to (-\infty, \infty]$ be an extended real-valued function. We say f is *proper* if there exists at least one $x \in \mathbb{R}^n$ such that $f(x) < \infty$. The function f is called *lower semi-continuous* if for all $x_0 \in \mathbb{R}^n$ the

$$\liminf_{x \to x_0} f(x) \geq f(x_0).$$

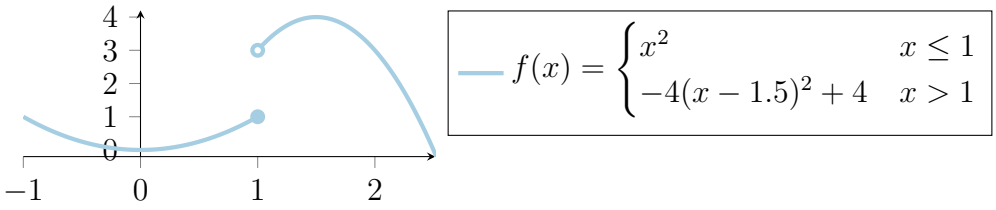

FIGURE 5.3: A lower semi-continuous function.

The convergence analysis of PALM requires that the nonsmooth part ϕ is a proper, lower semi-continuous function. In practice, this is not a demanding requirement. We display an example of a lower semi-continuous function in Figure 5.3. Let us exercise what is stated in the definition by this example. The function f has a point of discontinuity at $x = 1$. Any limit point of function values $f(x_k)$ for a sequence converging to the point of discontinuity ($x_k \to 1$) is either equal to one or to three. Thus, we have $\{1, 3\} \ni \liminf_{x \to 1} f(x) \geq f(1) = 1$. In other words, lower semi-continuous functions always attain the lowest limit point of function values. The restriction to lower semi-continuous functions ϕ ensures inter alia that the prox-operator is well-defined (cf. Eq. (5.4)).

DEFINITION 5.2 (KURDYKA-LOJASIEWICZ FUNCTIONS). Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper lower semi-continuous function and $x_0 \in \mathbb{R}^n$, such that[1] $\partial f(x_0) \neq \emptyset$. Define a subset of real-valued, continuous functions as

$$\mathcal{K} = \{\kappa : [0, t) \to \mathbb{R}_+ \mid t \in (0, \infty], \kappa(0) = 0, \kappa \in C^1(0, t), \kappa \in C^0[0, t)\}.$$

The function f is said to have the *Kurdyka-Łojasiewicz property* at point x_0 if there exists a concave and strictly increasing function $\kappa \in \mathcal{K}$ and an $\epsilon > 0$ such that for all $x \in \mathcal{N}_\epsilon(x_0)$ satisfying $f(x_0) < f(x) < f(x_0) + t$ the Kurdyka-Łojasiewicz inequality holds for all vectors $g \in \partial f(x)$, that is

$$\kappa'\left(f(x) - f(x_0)\right) \|g\| \geq 1.$$

A function f is called Kurdyka-Łojasiewicz function if the Kurdyka-Łojasiewicz property is satisfied at any point x_0 where $\emptyset \neq \partial f$.

The impact that a requirement of the Kurdyka-Łojasiewicz property has is not easy to understand. Here, it is only important to know which class of functions satisfy this property and how we can derive for specified functions if they satisfy the Kurdyka-Łojasiewicz property. For this reason, we state some of the general rules from which we can conclude the Kurdyka-Łojasiewicz property for a given function in the following section. The following theorem states the main convergence result for PALM.

[1] ∂ denotes here the Fréchet subdifferential

THEOREM 5.3. *(Bolte et al., 2014) Let $F : \mathbb{R}^{n\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ be a smooth function and let $\phi_1 : \mathbb{R}^{n\times r} \to [0,\infty]$ and $\phi_2 : \mathbb{R}^{m\times r} \to [0,\infty]$ be proper and lower semi-continuous functions. Denote with $(\mathcal{X},\mathcal{Y}) \subseteq \mathbb{R}^{n\times r} \times \mathbb{R}^{m\times r}$ the feasible set, defined as*

$$\mathcal{X} = \{X \in \mathbb{R}^{n\times r} \mid \phi_1(X) < \infty\} \quad \textit{and} \quad \mathcal{Y} = \{Y \in \mathbb{R}^{m\times r} \mid \phi_2(Y) < \infty\},$$

Assume that the following conditions hold:

1. *the partial gradients of F are Lipschitz-continuous, satisfying the following inequality for real-valued functions $M_{\nabla_X F}(Y)$ and $M_{\nabla_Y F}(X)$*

$$\begin{aligned}\|\nabla_X F(X,Y) - \nabla_U F(U,Y)\| &< M_{\nabla_X F}(Y)\|X-U\| \quad \forall\{X,U\} \subseteq \mathcal{X},\ Y \in \mathcal{Y}\\ \|\nabla_Y F(X,Y) - \nabla_V F(X,V)\| &< M_{\nabla_Y F}(X)\|Y-V\| \quad \forall\{Y,V\} \subseteq \mathcal{Y},\ X \in \mathcal{X}\end{aligned}$$

2. *the Lipschitz-moduli are bounded on the feasible set; there exists a constant $M > 0$ such that $M_{\nabla_X F}(Y) < M$ and $M_{\nabla_Y F}(X) < M$ for all $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$,*

3. *the function $\Psi(X,Y) = F(X,Y) + \phi_1(X) + \phi_2(Y)$ satisfies the Kurdyka-Łojasiewicz property,*

then the sequence (X_k, Y_k) generated by the update rules in Eqs. (5.2) and (5.3), where the stepsizes are given as

$$\alpha_k = M_{\nabla_X F}(Y_k)^{-1}, \quad \beta_k = M_{\nabla_Y F}(X_{k+1})^{-1}.$$

converges to a critical point of the function Ψ.

5.2.2 Kurdyka-Łojasiewicz Property

Deriving the Kurdyka-Łojasiewicz property on the basis of its definition is hard. Fortunately, there are some classes of functions, which allow for a more easy derivation of the Kurdyka-Łojasiewicz property. We introduce here the semi-algebraic and definable functions.

DEFINITION 5.4 (SEMI-ALGEBRAIC SETS AND FUNCTIONS). A set $\mathcal{X} \subseteq \mathbb{R}^n$ is called *semi-algebraic* if it is equal to a finite union $\mathcal{X} = \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_k$ of sets

$$\mathcal{X}_l = \{x \in \mathbb{R}^n \mid p_i(x) = 0, q_i(x) < 0, 1 \le i \le d\},$$

where p_i and q_i are polynomials. A function $f : \mathbb{R}^n \to (-\infty,\infty]$ is called semi-algebraic if its graph $\{(x, f(x)) \mid x \in \mathbb{R}^n\}$ is a semi-algebraic set.

Determining whether a function is semi-algebraic or not does in most cases not require a derivation of the properties according to the definition. Instead, many semi-algebraic functions are composed according to the following rules.

EXAMPLE (SEMI-ALGEBRAIC SETS AND FUNCTIONS). The following functions are semi-algebraic:

1. polynomial functions $p(x) = \sum_{\mathbf{n}\in\mathbb{N}^n} a_{\mathbf{n}} x^{n_1} \cdot \ldots \cdot x^{n_n}$ where $a_{\mathbf{n}} \neq 0$ for only finitely many $\mathbf{n} \in \mathbb{N}^n$,
2. the sum $\alpha f(x) + \beta g(x)$, product $f(x)g(x)$, division $f(x)/g(x)$ for $g(x) \neq 0$ and composition $f(g(x))$ of semi-algebraic functions f and g,
3. the function $f(x) = \|x\|_p$ for $x \in \mathbb{R}^n$ and $p \geq 0$,
4. indicator functions of semi-algebraic sets.

Although the class of semi-algebraic functions is large, there are still some very common functions which do not belong to that class, such as the exponential function and the logarithm. For this purpose, we introduce another class of functions, covering the semi-algebraic functions.

DEFINITION 5.5 (DEFINABLE SETS AND FUNCTIONS). The family $\mathcal{O} = \{\mathcal{O}_i\}_{i\in\mathbb{N}}$ of collections of subsets $\mathcal{O}_n \subseteq \mathcal{P}(\mathbb{R}^n)$ is called an o-minimal structure over $\mathbb{R}$ if it satisfies the following axioms:

1. $(\mathcal{O}_n, \cup, \cap, \cdot^{\complement})$ is a Boolean algebra (cf. Definition 2.8),
2. $A \times \mathbb{R}, \mathbb{R} \times A \in \mathcal{O}_{n+1}$ and $\{(x_1, \ldots, x_{n-1}) \mid (x_1, \ldots, x_n) \in A\} \in \mathcal{O}_n$ for $A \in \mathcal{O}_n$,
3. $\mathcal{O}_1 = \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_m$ where $\mathcal{I}_j \in \{(a,b), [a,b], (a,b], [a,b)\}$, $a, b \in [-\infty, \infty], m \in \mathbb{N}$ and $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 < x_2\} \in \mathcal{O}_2$,
4. $\{(x_1, \ldots, x_{n-1}, x_1) \in \mathbb{R}^n\} \in \mathcal{O}_n$.

Given an o-minimal structure $\mathcal{O}$, we call a set $A \subseteq \mathbb{R}^n$ definable if $A \in \mathcal{O}_n$. A function $f : \mathbb{R}^n \to (-\infty, \infty]$ is called definable if its graph is a definable set $\{(x, f(x)) \mid x \in \mathbb{R}^n\} \in \mathcal{O}_{n+1}$.

Similarly like semi-algebraic function, the set of definable functions is closed under most basic operations.

EXAMPLE (DEFINABLE SETS AND FUNCTIONS). The following functions are definable (van den Dries, 1998):

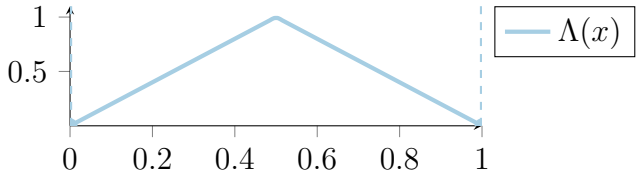

FIGURE 5.4: The function Λ penalizing nonbinary values.

1. semi-algebraic functions,
2. the sum $\alpha f(x) + \beta g(x)$, product $f(x)g(x)$, division $f(x)/g(x)$ for $g(x) \neq 0$ and composition $f(g(x))$ of definable functions f and g,
3. exp and log.

Most importantly, semi-algebraic functions are a subset of definable functions, which are again a subset of the Kurdyka-Łojasiewicz functions.

THEOREM 5.6. *(Attouch et al., 2010) Let $f : \mathbb{R}^n \to (-\infty, \infty]$ be a proper lower semi-continuous function. If f is definable in an o-minimal structure then it is a Kurdyka-Łojasiewicz function.*

5.3 Nonbinary Penalization within PALM

PALM states no convexity requirements for convergence in Theorem 5.3 and is thus suitable for the optimization of a broad class of objective functions. First, we discuss the minimization of the BMF problem in the framework of PALM, minimizing the approximation error of a Boolean factorization. The overall strategy is to derive a relaxed factorization of approximately binary matrices and to threshold these matrices to binary matrices in the end. Here, we discuss how the approximately binary factorization is derived.

5.3.1 A Binary Proximal Mapping

While the Mexican hat function (cf. Figure 3.1) can be seen as an $\ell 2$ regularization equivalent penalizer for binary values, here, we choose an $\ell 1$-equivalent form. Specifically, we choose $\phi_B(X) = \sum_{i,j} \Lambda(X_{ij})$, which employs the one-dimensional function

$$\Lambda(x) = \begin{cases} -|1 - 2x| + 1 & x \in [0, 1] \\ \infty & \text{otherwise} \end{cases}$$

to restrict matrix entries to the interval $[0, 1]$ and to penalize nonbinary values. As discussed by Zhang et al. (2010), restricting the entries of factor matrices between zero and one prevents an imbalance between the factor matrices in which one matrix is very sparse and the other very dense. The curve of Λ is depicted in Figure 5.4. We derive with the following proposition a closed form for the computation of the exact minimum as assigned by the proximal mapping with respect to ϕ_B.

THEOREM 5.7. *Let $\alpha > 0$ and $\phi_B(X) = \sum_{i,j} \Lambda(X_{ij})$ for $X \in \mathbb{R}^{m \times n}$. The proximal operator of $\alpha\phi_B$ maps the matrix X to the matrix $\text{prox}_{\alpha\phi_B}(X) = A \in [0, 1]^{m \times n}$ defined by $A_{ji} = \text{prox}_{\alpha\Lambda}(X_{ji})$, where for $x \in \mathbb{R}$ it holds that*

$$\text{prox}_{\alpha\Lambda}(x) = \begin{cases} \max\{0, x - 2\alpha\} & x \leq 0.5 \\ \min\{1, x + 2\alpha\} & x > 0.5 \end{cases}. \tag{5.5}$$

Proof. Let $\alpha > 0$, $X \in \mathbb{R}^{m \times n}$ for some $m, n \in \mathbb{N}$ and $A = \text{prox}_{\alpha\phi_B}(X)$. The function ϕ_B is fully separable across all matrix entries. In this case, the proximal operator can be applied entrywise to the composing scalar functions (Parikh et al., 2014), i.e., $A_{ji} = \text{prox}_{\alpha\Lambda}(X_{ji})$. It remains to derive the proximal mapping of Λ (Eq. (5.5)).

The proximal operator reduces to Euclidean projection if the argument lies outside of the function's domain (Parikh et al., 2014) and it follows that

$$\text{prox}_{\alpha\Lambda}(x) = 1 \text{ if } x > 1 \text{ and } \text{prox}_{\alpha\Lambda}(x) = 0 \text{ if } x < 0.$$

For $x \in [0, 1]$ we have $\Lambda(x) = -|1 - 2x| + 1$ and per definition of the proximal operator follows

$$\begin{aligned} \text{prox}_{\alpha\Lambda}(x) &= \arg\min_{x^* \in \mathbb{R}} \left\{ \frac{1}{2}(x - x^*)^2 - \alpha|1 - 2x^*| + 1\alpha \right\} \\ &= \arg\min_{x^* \in \mathbb{R}} \left\{ \underbrace{(x - x^*)^2 - 2\alpha|1 - 2x^*| + (2\alpha)^2}_{=g(x^*; x, \alpha)} \right\}, \end{aligned}$$

where the last step follows from a multiplication and addition of constants. The minimum of the function g is easily derived by completing the square

$$\begin{aligned} g(x^*; x, \alpha) &= \begin{cases} (x - x^*)^2 - 2\alpha(1 - 2x^*) + (2\alpha)^2 & x^* \leq 0.5 \\ (x - x^*)^2 + 2\alpha(1 - 2x^*) + (2\alpha)^2 & x^* > 0.5 \end{cases} \\ &= \begin{cases} (x^* - (x - 2\alpha))^2 + 2\alpha(1 - 2x) & x^* \leq 0.5 \\ (x^* - (x + 2\alpha))^2 - 2\alpha(1 - 2x) & x^* > 0.5 \end{cases}. \end{aligned}$$

The function g is a continuous piecewise quadratic function which attains its global minimum at the minimum of one of the two quadratic functions, i.e.,

$$\arg\min_{x^* \in \mathbb{R}} g(x^*; x, \alpha) \in \{x - 2\alpha \mid x \leq 0.5 + 2\alpha\} \cup \{x + 2\alpha \mid x > 0.5 - 2\alpha\}.$$

A function value comparison in the intersecting domain $x \in (0.5 - 2\alpha, 0.5 + 2\alpha]$ yields that

$$g(x - 2\alpha; x, \alpha) = -2\alpha(1 - 2x) \leq g(x + 2\alpha; x, \alpha) = 2\alpha(1 - 2x) \Leftrightarrow x \leq 0.5$$

□

The regularization with ϕ_B meets the requirements of the nonsmooth part in an optimization by PALM.

LEMMA 5.8. *The function $\phi_B(X) = \sum_{i,s} \Lambda(X_{is})$ is proper, lower semi-continuous and semi-algebraic function.*

Proof. The function ϕ_B is lower semi-continuous, because the function Λ is lower semi-continuous. At points of discontinuity of Λ we have

$$\liminf_{x \to 0} \Lambda(x) = \liminf_{x \to 1} \Lambda(x) = 0 = \Lambda(0) = \Lambda(1).$$

Similarly, the semi-algebraic property of ϕ_B, being a finite sum of function values of Λ, follows from the semi-algebraic property of $\Lambda(x)$. We write $\Lambda(x) = \delta_{[0,1]}(x)|1 - (1-2x)|$, where $\delta_{[0,1]}(x)$ is the indicator function, returning infinity if x is outside of the interval $[0, 1]$ and one otherwise. The function $|1-(1-2x)|$ is semi-algebraic as a composition of a polynomial function and the one-norm. The indicator function $\delta_{[0,1]}(x)$ is semi-algebraic if the interval $[0, 1]$ is a semi-algebraic set. This is indeed the case according to the definition of semi-algebraic sets, since the interval $[0, 1] = \{x \mid p(x) < 0\} \cup \{x \mid p(x = 0)\}$ for the polynomial function $p(x) = (x - 0.5)^2 - 0.25$. □

5.3.2 The Proposed BMF Framework: PAL-Tiling

We sketch our method, the general optimization scheme for Boolean matrix factorizations, called Proximal Alternating Linearized Tiling (PAL-Tiling) in Algorithm 7. Given an objective function $L(X, Y)$ having a suitable nonbinary relaxation $F(X, Y)$, which satisfies the convergence requirements of PALM concerning the smooth part of the objective, PAL-Tiling computes a Boolean matrix factorization via the nonbinary penalty approach. Our method is suitable to automatically determine the rank of the factorization. We will discuss this feature

ALGORITHM 7 The proposed general optimization scheme PAL-TILING for Boolean matrix factorizations. The implementation of the function TOSS, used in the function ROUND, specifies which tiles are kept. the specification of this method determines the rank selection strategy.

1: **FUNCTION** PAL-TILING$(D; \Delta_r = 10)$
2: $(X_K, Y_K) \leftarrow (\emptyset, \emptyset)$
3: **FOR** $r \in \{\Delta_r, 2\Delta_r, 3\Delta_r, \ldots\}$ **DO**
4: $(X_0, Y_0) \leftarrow$INCREASERANK(X_K, Y_K, Δ_r) ▷ Append Δ_r random columns
5: **FOR** $k = 0, 1, \ldots$ **DO** ▷ Select stopping criterion
6: $\alpha_k^{-1} \leftarrow M_{\nabla_X F}(Y_k)$
7: $X_{k+1} \leftarrow \text{prox}_{\alpha_k \phi_B}\left(X_k - \alpha_k \nabla_X F(X_k, Y_k)\right)$
8: $\beta_k^{-1} \leftarrow M_{\nabla_Y F}(X_{k+1})$
9: $Y_{k+1} \leftarrow \text{prox}_{\beta_k \phi_B}\left(Y_k - \beta_k \nabla_Y F(X_{k+1}, Y_k)\right)$
10: **END FOR**
11: $(X, Y) \leftarrow$ROUND(L, X_k, Y_k) ▷ Specify Function TOSS
12: **IF** RANKGAP(X, Y, r) **THEN RETURN** (X, Y) **END IF**
13: **END FOR**
14: **END FUNCTION**

15: **FUNCTION** ROUND(L, X_k, Y_k, D)
16: $(X^*, Y^*, L^*) \leftarrow (0, 0, \infty)$
17: **FOR** $t_x, t_y \in \{0, 0.05, 0.1, \ldots 1\}$ **DO**
18: $(X, Y) \leftarrow \left(\theta_{t_x}(X_k), \theta_{t_y}(Y_k)\right)$
19: **FOR** $s \in \{1, \ldots, r\}$ **DO**
20: **IF** TOSS$(X_{\cdot s}, Y_{\cdot s}, D)$ **THEN** $(X_{\cdot s}, Y_{\cdot s}) \leftarrow (\mathbf{0}, \mathbf{0})$ **END IF**
21: **END FOR**
22: **IF** $L(X, Y) < L^*$ **THEN** $(X^*, Y^*, L^*) \leftarrow (X, Y, L(X, Y))$ **END IF**
23: **END FOR**
24: **RETURN** (X^*, Y^*)
25: **END FUNCTION**

thoroughly in the following two sections. Here, we state the general scheme, having only the data matrix D and the rank increment Δ_r as input parameters. The rank determination proceeds as follows: increase the rank until the returned factorization uses fewer tiles than allowed under the provided rank. Decisive for the success or failure of this method is the specification which tiles are used. This depends on the objective function $L(X, Y)$ and the implementation of the function TOSS which is invoked in line 20.

Let us now go through the algorithm step by step. The factor matrices are initialized as empty matrices (having rank zero). At every rank increase, Δ_r columns of random values between zero and one are appended to the current matrices (X_K, Y_K) in line 4. In lines 5 and 11, the Boolean factorization of rank r is calculated. The function ϕ_B is the nonbinary penalization function, which has been defined in the previous Section. After the numerical minimization of the relaxed objective $F(X, Y)$, the matrices X_K and Y_K, having entries between zero and one, are rounded to binary matrices X and Y with respect to the Boolean factorization objective $L(X, Y)$. If the rounding procedure returns binary matrices having a rank smaller than r, then the current factorization is returned (line 12). Otherwise, we increase the rank and add Δ_r random columns to the relaxed solution of the former iteration (X_K, Y_K). Note that our method is not greedy as only the initialization of the factor matrices is based on the result of the former iteration. During the optimization of the relaxed objective, the results of former iterations, employing a smaller rank, might very well be changed.

Regarding the optimization, we propose as stopping criterion to check whether the average function decrease of the last 500 iterations is small enough. That is, at iteration k we check whether $\frac{1}{500}\sum_{t=k-500}^{k}(F(X_{t-1}, Y_{t-1}) - F(X_t, Y_t)) < \epsilon$. How to set the parameter ϵ is typically easily inferred from observing the function decrease during the first $10,000$ iterations. For most datasets, a maximum number of ten thousand iterations is appropriate. However, tracing the average function decrease helps to decide when to abort the optimization reasonably early. Throughout this work, we employ a stopping criterion of maximal 50,000 optimization iterations or a minimum average function decrease of $\epsilon = 10^{-4}$.

We also outline the rounding function in Algorithm 7. This function basically computes the thresholding parameters t_x and t_y with which the relaxed factor matrices are rounded to binary matrices, such that the objective function L is minimized. Some tiles of the binary factorization are tossed according to a specified criterion. Therefore, particular instances of the PAL-Tiling algorithm specify the objective function $L(X, Y)$, its smooth relaxation $F(X, Y)$ together with the corresponding gradients and the Lipschitz moduli $M_{\nabla_X F}(Y)$ and $M_{\nabla_Y F}(x)$ as well as the function Toss.

The default objective in Boolean matrix factorization is the minimization of the residual sum of squares as stated in problem (BMF). We state the function, gradients and Lipschitz constants, required for the minimization via PAL-Tiling if the rank of the factorization is provided in the following algorithm specification.

ALGORITHM SPECIFICATION 1 (VANILLA PAL-TILING). Apply Algorithm 7 PAL-TILING, with the following specification of the objective function L, its smooth part F and their partial gradients:

$$\begin{aligned} L(X,Y) &= \left|D - \theta\left(YX^\top\right)\right| \\ F(X,Y) &= \left\|D - YX^\top\right\|^2 \\ \nabla_X F(X,Y) &= \left(YX^\top - D\right)^\top Y \\ \nabla_Y F(X,Y) &= \left(YX^\top - D\right)X. \end{aligned}$$

The Lipschitz moduli of the partial gradients are given by

$$M_{\nabla_X F}(Y) > \left\|YY^\top\right\|, \qquad M_{\nabla_Y F}(X) > \left\|XX^\top\right\|.$$

The derivation of the Lipschitz moduli follows from the triangle inequality:

$$\begin{aligned} \left\|\nabla_X F(X,Y) - \nabla_U F(U,Y)\right\| &= \left\|(YX^\top - D)^\top Y - (YU^\top - D)^\top Y\right\| \\ &= \left\|XY^\top Y - UY^\top Y\right\| \leq \left\|X - U\right\|\left\|Y^\top Y\right\|. \end{aligned}$$

The function $F(X,Y) = \sum_{j,i}(D_{ji} - Y_{j\cdot}X_{\cdot i})^2$ is obviously smooth, because it is a polynomial function. For that reason, F is also a semi-algebraic function.

COROLLARY 5.9. *The function $F(X,Y) = \|D - YX^\top\|^2$ is smooth and a semi-algebraic function.*

The implementation of the vanilla PAL-TILING could be used to optimize a Boolean factorization of a fixed rank. However, we are particularly interested in an automatic determination of the rank of the factorization. Therefore, we discuss approaches to do so in the following sections, where we evaluate our method in comparison to state-of-the-art Boolean matrix factorization algorithms.

5.4 Discussion

We propose an optimization scheme for the problem of Boolean matrix factorization, based on latest numerical optimization results for nonconvex, objective functions. To this end, we derive a closed form of the proximal mapping with respect to a function which penalizes nonbinary values. A thresholding to binary values according to the actual objective function, involving the Boolean product, enables the derivation of factorizations which reflect overlapping clusterings, as desired by Boolean matrix factorization.

The proposed approach has applications to the general problem of matrix factorizations under binary constraints. This class of optimization problems comprises the learning of binary hash codes as well. In this scope, Shen et al. (2016) similarly propose to find optimal hashing codes via PALM. However, in this approach the binary factor matrix denoting the hash codes is strictly constrained to binary values during PALM's optimization. This is achieved by employing the regularizing function $\phi_{\mathbb{B}}(X)$, returning zero if the matrix X is binary and returning infinity otherwise. The proximal mapping of the function $\phi_{\mathbb{B}}$ maps therefore a matrix to its closest binary matrix. Although the resulting optimization with respect to unrelaxed, binary matrices seems desirable at first, we argue that such a procedure is likely to get stuck in local optima which reflect less suitable solutions.

CHAPTER 6

BMF Rank Selection by the Minimum Description Length

The possibility to derive a Boolean factorization by means of numerical optimization methods is a promising theoretically founded alternative to the predominantly used greedy heuristic (cf. Section 3.1). Yet, we also wish to provide a solution to the well researched topic of model selection in Boolean matrix factorization by virtue of PAL-Tiling. However, this objective is difficult to delineate: where to draw the line between structure and noise? One possibility is to apply the *Minimum Description Length* (MDL) principle as proposed by Miettinen and Vreeken (2014) to reduce the considerations into one: exploit just as many regularities as serves the compression of the data. Here, regularities indicate the structural part, the similarities between observations which establish a cluster. The description length counterbalances the complexity of the model (the amount of derived clusters) with the fit to the data, measured by the size of the encoded data using the model. Decisive for the feasibility of extracted components is the definition of the encoding.

Miettinen and Vreeken (2014) evaluate several encodings with respect to their ability to filter a planted structure from noise. Candidate clusters are created by the Boolean matrix factorization method Asso and are afterwards selected such that a specified description length is minimized. Another framework proposed by Lucchese et al. (2014) greedily selects the clusters which directly minimize the description length. Most recently, Karaev et al. (2015) propose another greedy scheme with focus on a setting where noise effects more likely flip ones to zeros than the other way round. All these methods are capable to identify the underlying clusters in respectively examined settings. By and large, the experiments indicate

however that the quality considerably varies depending on the distribution of noise and characteristics of the dataset (Miettinen and Vreeken, 2014; Karaev et al., 2015). Our main contributions are:

- we facilitate the application of the framework *PAL-Tiling* to optimize a description length having an approximation which is suitable for the optimization via PALM,
- we algorithmically integrate the built-in regularization of the model complexity by MDL in order to derive factorizations which do not use more tiles than necessary,
- we propose two algorithms as specifications of PAL-Tiling: one applying $\ell 1$-regularization on the matrix factorization (Panpal) and one employing an encoding by code tables as proposed by Siebes et al. (2006) (Primp).

6.1 MDL Principle

MDL is introduced by Rissanen (1978) as an applicable version of Kolmogorov complexity (Li and Vitányi, 2008; Grünwald, 2007). The learning task to find the best model according to the MDL principle is given by the following minimization problem

$$\min_M L(M) = L^D(M) + L^M(M) \qquad \text{s.t. } M \in \mathcal{M}.$$

The function L denotes the description length, which is composed of the compression size of the database in bits $L^D(M)$ (using model M for the encoding) and description size in bits of the model M itself $L^M(M)$. Specifications of this task differ in the definition of the encoding which defines in turn the set of considered models $\mathcal{M}$.

6.1.1 MDL for Pattern Mining

The minimum description length is suitable to tackle the problem of pattern explosion, arising in practical applications of pattern mining. The output of frequent pattern mining algorithms is sensitive to the minimum support threshold, defining the meaning of the word *frequent*. If this threshold is set too high, then only a few patterns are returned, reflecting nothing but common knowledge. A small decrease in the minimum support can result however in the output of a vast amount of patterns, whereby most of them are redundant as well. The minimum description length is applicable to select a set of patterns, giving a succinct description of all frequent patterns.

An encoding which is successfully applied in this respect uses code tables as proposed by Siebes et al. (2006). The two-columned code table assigns optimal prefix-free codes to a set of patterns: itemsets are listed on the left and assigned codes on the right. Such a dictionary from itemsets to code words can be applied to databases similarly as code words to natural language texts. However, the usage of codes for patterns in a transaction is not as naturally defined as for words in a text. Patterns are not nicely separated by blanks and the possibilities to disassemble a transaction into patterns are numerous. Therefore, we require for every transaction the indication of those patterns which are used for the transaction's encoding. This is modeled by the function *cover*, which partitions the items in a transaction into patterns of the code table.

Let $CT = \{(\mathcal{X}_s, C_s) | 1 \leq s \leq r\}$ represent a code table of r patterns $\mathcal{X}_s \subseteq \{1, \ldots, n\}$ and codes C_s. Theorem 5.4.1 in Cover and Thomas (2012) states that for any distribution P over a finite set Ω, an optimal set of prefix-free codes exists such that the number of required bits for the code of $x \in \Omega$ is approximately

$$codelength(x) \approx -\log(P(x)).$$

Desiring that frequently used codes are shorter in size, Siebes et al. (2006) introduce the function *usage* that maps a pattern to the number of transactions which use it for their cover,

$$usage(\mathcal{X}_s) = |\{j \in \{1, \ldots, m\} \mid \mathcal{X}_s \in cover(CT, D_{j\cdot})\}|.$$

The probability mass function over all itemsets $\mathcal{X}_s$ in the code table is defined as

$$P(\mathcal{X}_s) = \frac{usage(\mathcal{X}_s)}{\sum_{1 \leq t \leq r} usage(\mathcal{X}_t)}. \tag{6.1}$$

This implies that $codelength(\mathcal{X}_s) = -\log P(\mathcal{X}_s)$. The data matrix is encoded by a transactionwise concatenation of codes, denoted by the cover, i.e., transaction $D_{j\cdot}$ is encoded by a concatenation of codes C_s with $\mathcal{X}_s \in cover(CT, D_{j\cdot})$. Code C_s occurs $usage(\mathcal{X}_s)$ times in the encoded dataset. The size of the data description is then computed as

$$L^D_{\mathsf{CT}}(CT) = -\sum_{1 \leq s \leq r} usage(\mathcal{X}_s) \cdot \log(P(\mathcal{X}_s)).$$

The description of the model, the code table, requires the declaration of codes C_s and corresponding patterns $\mathcal{X}_s$. Code C_s has a bitlength of $-\log(P(\mathcal{X}_s))$ and a pattern $\mathcal{X}_s$ is described by concatenated standard codes of contained items. Standard codes arise from the code table consisting of singleton patterns only,

where the usage of singleton $\{i\}$ for $i \in \{1, \ldots, n\}$ is equal to its support $|D_{\cdot i}|$. In conclusion, the description size of the model is computed as

$$L^M_{\mathsf{CT}}(CT) = - \sum_{\substack{1 \leq s \leq r \\ usage(\mathcal{X}_s) > 0}} \left(\log\left(P(\mathcal{X}_s)\right) + \sum_{i \in \mathcal{X}_s} \log\left(\frac{|D_{\cdot i}|}{|D|}\right)\right).$$

Note that an efficient computation of the description length employs the possibility to calculate code lengths without realizations of actual codes. We further remark that the function L_{CT} originally uses the logarithm with base two. Here, we implicitly reformulate this description length by substituting with the natural logarithm. This is equivalent to multiplying the function by a constant which is negligible during minimization. In return, using the natural logarithm will shorten the derivations in Section 6.2.2.

Siebes et al. (2006) use a heuristic cover function in the algorithm Krimp which employs a specified, static order on patterns. For every transaction, the cover function returns a partition of the transaction by the following routine: iterating through all patterns of the code table in the given order, add a pattern to the transactions' cover whenever the transaction supports the pattern. Remove the items belonging to a selected pattern from the transaction and continue the pattern traversion. This way, the usage of a pattern is smaller than or equal to its support. The determination of patterns contained in the code table is performed by another greedy procedure in Krimp. An input set of frequent patterns – the code table candidate set – is traversed in another static order, adding a candidate pattern to the code table whenever that improves the compression size. Additionally, pruning methods are proposed to correct the selection of patterns in the code table.

Slim (Smets and Vreeken, 2012) differs in its candidate generation, which is dynamically implemented according to an estimated compression gain and dependent on the current code table. This strategy typically improves the compression size, but mainly reduces the amount of returned patterns. Both approaches consider a vast amount of candidates until the best set of patterns is determined. Time consumption is dominated by computing the usage for each evaluated candidate. SHrimp (Hess et al., 2014) exploits the indexing nature of trees in order to efficiently identify those parts of the database which are affected by an extension of the code table. Siebes and Kersten (2011) restrict with the algorithm Groei the code table to a constant number of patterns. They resort to a heuristic beam search algorithm, but only for tiny datasets, the beam width parameter can be set to a level allowing a reasonably wide enough exploration of the search space, or else the run time explodes.

All these algorithms follow the heuristic cover definition of Krimp which restricts the usage of a pattern to supporting transactions and prohibits a cover

by overlapping patterns. As discussed in the context of tiling (cf. Section 2.4.5), these restrictions result in less succinct descriptions of the dataset, being susceptible to noise. Correspondingly, there are methods adapting the model selection by minimum description length for Boolean matrix factorization.

6.1.2 MDL for BMF

The scheme to employ MDL for rank selection is easily adopted from the one used for pattern mining. In every iteration, the description length of the current factorization is compared with the description length from the former iteration, having a lower rank. If the current factorization achieves a lower description length, then another factorization with an increased rank is computed and the procedure repeats itself. Otherwise, the factorization of the former iteration is returned. The performance of this method depends on the algorithm computing the factorization of a given rank and on the encoding which determines the description length.

Lucchese et al. (2010) propose an encoding reflecting sparse data representations; describing matrices only by their positions of ones. Consequently, the model is described with $L^M_{\ell 1}(X,Y) = |X| + |Y|$ bits and the data with $L^D_{\ell 1}(X,Y) = |D - \theta(YX^\top)|$ bits, up to a multiplicative constant. We denote the resulting description length with $L_{\ell 1}(X,Y) = |D - \theta(YX^\top)| + |X| + |Y|$. The algorithm Panda uses a factorization method which adds a tile (outer product) to the current factorization in a two stage heuristic.

Miettinen and Vreeken (2014) argue that the encoding used in Panda is too coarse. They investigate multiple encodings, applying Asso as Boolean factorization method. Their best-performing encoding is called *Typed XOR DtM* encoding. This is based upon the description of n-dimensional binary vectors by number and distribution of ones. We refer to the Typed XOR DtM description length as L_{TXD} and to the corresponding algorithm as Mdl4bmf. The experimental evaluation suggests that Mdl4bmf's rank estimation is accurate in a setting with moderate noise, i.e., less than 15%, and moderate number of planted tiles, i.e., less than 15. It seems to have a tendency to underfit, as opposed to Panda, which returns on some synthetically generated datasets a rank ten times higher than the actual rank.

On the other hand, the heuristic optimization of Panda is applicaple to any objective function. Lucchese et al. (2014) enhance the algorithm Panda to a faster version Panda+ and empirically evaluate the ability to detect the *true* clustering via various description lengths and optimization algorithms. In the evaluation with respect to synthetically generated datasets, having less than 10% equally distributed noise, Panda+ using the Typed XOR description length L_{TX}

is outperforming any other choice. The performance is explained with the direct minimization of the description length in PANDA+, opposed to ASSO, minimizing only the residual sum of squares.

Another algorithm which tries to incorporate the direct optimization of the MDL-cost measure is NASSAU (Karaev et al., 2015). Remarking that the formerly proposed algorithms do not reconsider which clusters have been mined at previous iterations, NASSAU refines the whole factorization every few steps, employing ASSO for the BMF optimization. The experiments focus on a setting where noise effects which flip a one to a zero are prevalent. In this case, differences to MDL4BMF are often hard to capture while NASSAU typically outperforms PANDA+.

6.2 Minimizing the Description Length in PAL-Tiling

We wish to derive Boolean factorizations minimizing the description length, hoping that the regularization of the model complexity by the MDL principle is suitable to yield factorizations of an appropriate rank. However, most of the proposed description lengths are not continuous, let alone smooth. Therefore, we need to derive a smooth relaxation of the description lengths we wish to apply in PAL-TILING. We assume that there exists a factor $\mu \geq 0$ and a regularizing function G such that the smooth relaxation of the description length has the form

$$F(X,Y) = \frac{\mu}{2}\left\|D - YX^\top\right\|^2 + \frac{1}{2}G(X,Y). \tag{6.2}$$

Here, the multiplication with one half refers to the traditional formulation of the residual sum of squares in problem (NMF), which shortens the formulation of gradients. In order to meet the convergence criteria of PALM, we require the regularizing function G to be a smooth definable function, having partial gradients which are Lipschitz-continuous with moduli $M_{\nabla_Y G}(X)$ and $M_{\nabla_X G}(Y)$, such that

$$\|\nabla_X G(X,Y) - \nabla_U G(U,Y)\| < M_{\nabla_X G}(Y)\|X - U\|,$$

and similarly for $\nabla_Y G$. From the triangle inequality follows that the Lipschitz moduli of the partial gradients of F are then given as

$$M_{\nabla_X F}(Y) = \mu\left\|YY^\top\right\| + \frac{1}{2}M_{\nabla_X G}(Y)$$
$$M_{\nabla_Y F}(X) = \mu\left\|XX^\top\right\| + \frac{1}{2}M_{\nabla_Y G}(X).$$

We specify the function TOSS in Algorithm 8, which we employ in the framework

ALGORITHM 8 The tossing function for the application of PAL-TILING implementing the determination of the rank via the minimum description length.

```
1: FUNCTION Toss(x, y, D)
2:     RETURN |x| ≤ 1 OR |y| ≤ 1
3: END FUNCTION
```

of PAL-TILING to determine the rank via the minimum description length principle. Since the regularization of the model complexity by MDL is supposed to yield factorizations which penalizes tiles reflecting noise effects, we employ a very simplistic function TOSS. The function solely decides that singleton tiles, consisting of one row or column, are to be disregarded in the factorization. Therewith, we only need a specification of the description lengths and their relaxed objective for the optimization in PAL-TILING.

6.2.1 Panpal

The cost measure $L_{\ell 1}$ (as applied by PANDA) is easily integrated into PAL-TILING. Since the proximal operator ensures that the factor matrices in all steps are nonnegative, the $\ell 1$-norm of the factor matrices equates a simple summation over all matrix entries. Thus, the $\ell 1$-norm is a smooth function on the nonnegative domain of the factor matrices and can be used as regularizing function. We call the resulting algorithm PANPAL as it employs the cost measure of PANDA in the minimization technique PAL-TILING:

ALGORITHM SPECIFICATION 2 (PANPAL). Apply PAL-TILING with the function TOSS from Algorithm 8. We employ the following objective L and its smooth part F:

$$L(X,Y) = L_{\ell 1}(X,Y) = \left|D - \theta\left(YX^\top\right)\right| + |X| + |Y|$$
$$F(X,Y) = \frac{1}{2}\left\|D - YX^\top\right\|^2 + \frac{1}{2}(|X| + |Y|)$$

The partial gradients and the corresponding Lipschitz moduli are given as follows:

$$\nabla_X F(X,Y,D) = \left(YX^\top - D\right)^\top Y + 0.5 \cdot \mathbf{1}$$
$$\nabla_Y F(X,Y,D) = \left(YX^\top - D\right)X + 0.5 \cdot \mathbf{1}$$
$$M_{\nabla_X F}(Y) > \left\|YY^\top\right\|, \qquad M_{\nabla_Y F}(X) > \left\|XX^\top\right\|$$

The smooth relaxation F in PANPAL is a polynomial function and thus semi-algebraic. Therefore, the optimization of the relaxed objective of PANPAL via PALM converges to a local minimum.

6.2.2 Primp

So far, the cost measure of KRIMP has been disregarded in the context of Boolean matrix factorization. Since the traditionally employed cover function is incompatible with overlapping patterns or patterns which cover more items than persistent in the transaction, the task to find the best encoding by code tables is associated with the sub-domain of pattern mining (Miettinen and Vreeken, 2014; Lucchese et al., 2014; Karaev et al., 2015). The definition of the long-established cover function is heuristically determined under the assumption that there is one globally valid cover function which is applicable on all datasets if a suitable code table is found. Although this approach might be favorable in sub-domains like classification or detection of changes in a data stream (Vreeken et al., 2011; Leeuwen and Siebes, 2008), there is generally no reason why the cover function should not be data-dependent. Thus, we break away from the conventional view on the cover function as a predefined instance and regard it as an extrapolation of the mapping from patterns to transactions which is defined by the matrix Y. Thereby, we intend to learn a suitable pair of code table and cover function, which is tailored to the dataset. This is motivated by the following observation.

LEMMA 6.1. *Let D be a data matrix. For any code table CT and its cover function there exists a Boolean matrix factorization $D = \theta\big(YX^\top\big) + N$ such that non-singleton patterns in CT are mirrored in X and the cover function is reflected by Y. The description lengths correspond to each other, such that*

$$L_{\mathsf{CT}}(CT) = L_{\mathsf{CT}}(X, Y) = L^D_{\mathsf{CT}}(X, Y) + L^M_{\mathsf{CT}}(X, Y),$$

where the functions returning the model and the data description size are given as

$$\begin{aligned} L^D_{\mathsf{CT}}(X, Y) &= -\sum_{s=1}^{r} |Y_{\cdot s}| \cdot \log(p_s) - \sum_{i=1}^{n} |N_{\cdot i}| \cdot \log(p_{r+i}) && = L^D_{\mathsf{CT}}(CT) \\ L^M_{\mathsf{CT}}(X, Y) &= \sum_{s:|Y_{\cdot s}|>0} \big(X_{\cdot s}^\top \mathbf{c} - \log(p_s)\big) + \sum_{i:|N_{\cdot i}|>0} (\mathbf{c}_i - \log(p_{r+i})) && = L^M_{\mathsf{CT}}(CT). \end{aligned}$$

The probabilities p_s and p_{r+i} indicate the relational usage of non-singleton patterns $X_{\cdot s}$ and singletons $\{i\}$,

$$p_s = \frac{|Y_{\cdot s}|}{|Y| + |N|}, \; p_{r+i} = \frac{|N_{\cdot i}|}{|Y| + |N|}.$$

We denote with $\mathbf{c} \in \mathbb{R}^n_+$ the vector of standard code lengths for each item, i.e.,

$$\mathbf{c}_i = -\log\left(\frac{|D_{\cdot i}|}{|D|}\right).$$

The proof of Lemma 6.1 can be found in Appendix B. The transfer from a code table encoding to a Boolean matrix factorization provides another view on the objective of KRIMP-related algorithms. While the focus of matrix factorizations lies on the extraction of a given ground truth, the originally formulated task aims at the derivation of subjectively interesting patterns – equating interestingness with the ability to compress. The non-reducibility requirement in Boolean matrix factorization, returning linearly independent columns of factor matrices and the bound on the rank ($r \leq \min\{m, n\}$) is in alignment with the interpretation of interestingness with compressing abilities.

Considering, in reverse, the transfer from a matrix factorization to an encoding by code tables, we naturally receive access to the treatment of negative noise. The term *negative noise* refers to the decomposition of the data into a factorization and a noise matrix $D = \theta\big(YX^\top\big) + N$. If patterns are not restricted to their support, then the noise matrix $N \in \{-1, 0, 1\}^{m \times n}$ may take negative values. We can calculate the description size $L_{\mathsf{CT}}(X, Y)$ for arbitrary factor matrices, even if the usage of patterns is not restricted to their support. Yet, the question arises if this also has a suitable interpretation with regard to the encoding. In fact, the interpretation is simple: the items in a transaction having a negative noise entry can be transmitted just as the items with positive noise entries; their singleton codes are appended to the belonging transaction. If the item is not contained in any other pattern used in this transaction, then it corresponds to a positive and otherwise to a negative noise entry. The resulting cover function maps a transaction $D_{j\cdot}$ to the patterns $X_{\cdot s}$ where $Y_{js} = 1$ and the singletons i having $N_{ji} \neq 0$.

The compression size $L_{\mathsf{CT}}(X, Y)$ is not continuous. There are points of discontinuity at factorizations where one of the columns of Y or N are zero, that is when a pattern in X or a singleton code is not used at all. We employ a simple hack to fix this issue and assume that each pattern in X is used at least once. For singletons, we do not wish to make such an assumption; usages of singletons indicate errors in the approximation, which we want to keep as small as possible. Therefore, we bound the description length of singleton codes by the approximation error. Then, we obtain a smooth function which meets the requirements of PAL-TILING. This is specified by the following theorem whose proof can be found in Appendix B.

THEOREM 6.2. *Given binary matrices X and Y and $\mu = 1 + \log(n)$, it holds that*

$$L^D_{\mathsf{CT}}(X, Y) \leq \mu \big\| D - YX^\top \big\|^2 - \sum_{s=1}^{r} (|Y_{\cdot s}| + 1) \log\left(\frac{|Y_{\cdot s}| + 1}{|Y| + r}\right) + |Y| \tag{6.3}$$

The description length of the data is bounded by the smooth function on the right of Eq. (6.3). The description length of the model is discontinuous at points

where one of the patterns is not used at all. The description size of one side of the code table, representing the patterns by singleton codes $\mathbf{c}$, has an upper bound of

$$\sum_{s:|Y_{\cdot s}|>0} X_{\cdot s}^\top c \leq \sum_{s=1}^{r} X_{\cdot s}^\top c = \left|X^\top \mathbf{c}\right|,$$

which is easily integrated into the smooth approximation. The product $X^\top \mathbf{c}$ returns a nonnegative vector, and thus the the $\ell 1$-norm of this vector boils down to a simple sum over all entries. The remaining terms which compose the model complexity are limited upwards by a constant, due to the fixation of the rank during the minimization of the relaxed objective. Thus, we minimize the relaxed function as denoted in the box below. The required Lipschitz constants are computed in Appendix B. We refer to this algorithm as PRIMP, as it performs PAL-TILING with the objective of KRIMP.

ALGORITHM SPECIFICATION 3 (PRIMP). Apply PAL-TILING with the function TOSS from Algorithm 8. We employ the following constants, objective L and its smooth part F:

$$\begin{aligned}
c_i &= -\log\left(\frac{|D_{\cdot i}|}{|D|}\right), \quad \mu = 1 + \log(n) \\
L(X,Y) &= L_{\mathsf{CT}}(X,Y) \\
F(X,Y) &= \frac{\mu}{2}\left\|D - YX^\top\right\|^2 + \frac{1}{2}G(X,Y) \\
G(X,Y) &= -\sum_{s=1}^{r}(|Y_{\cdot s}|+1)\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) + \left|X^\top\mathbf{c}\right| + |Y|.
\end{aligned}$$

The partial gradients are given as follows:

$$\begin{aligned}
\nabla_X F(X,Y) &= \mu\big(YX^\top - D\big)^\top Y + \frac{\mathbf{c}\mathbf{1}^\top}{2} \\
\nabla_Y F(X,Y) &= \mu\big(YX^\top - D\big)X - \frac{1}{2}\left(\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) - 1\right)_{js}.
\end{aligned}$$

Further, we specify the employed Lipschitz moduli:

$$M_{\nabla_X F}(Y) > \mu\left\|YY^\top\right\|, \qquad M_{\nabla_Y F}(X) > \mu\left\|XX^\top\right\| + m.$$

The smooth relaxation of PRIMP is a definable function, because it is a composition of polynomials, division by functions which are not zero and the definable logarithmic function. Therefore, the iterates in the relaxed optimization converge to a local minimum of the relaxed objective.

6.3 Experiments

We conduct experiments on a series of synthetic data matrices, exploring the ability to detect the planted factorization, i.e., to recover generated matrices X^* and Y^* in presence of various noise structures. In real-world data experiments we compare the description lengths of obtained models. Also, we perform a qualitative evaluation of the factor methods, visualizing the algorithms' understanding of tiles and noise on the basis of images. We compare the PAL-Tiling instances Panpal and Primp with the available implementations of PaNDa+[1], Mdl4bmf[2] and Nassau[2]. For Panda+ we choose the TypedXOR measure and use 20 randomization rounds and correlating items as suggested in the literature (Lucchese et al., 2014). Apart from that, the default settings apply.

We exclude Slim from our experiments as it can not be fairly compared to Boolean matrix factorization algorithms. To illustrate, Slim returns far more patterns (500 to 3000 monotonically increasing with noise) than planted (25) in our synthetic datasets. Hence, a depiction of this algorithm's rank would distort the rank charts.

For synthetic and real-world experiments, we set the rank increment $\Delta_r = 10$; sensitivity to this parameter is explored in Section 6.3.1. For our image evaluation, we set (if possible) a maximum number of 10 returned tiles and depict the four most informative tiles. Here, we set $\Delta_r = 1$, to consistently allow for multiple factorization rounds.

A separate run time comparison of the aforementioned algorithms is not conducted. This is because we can not guarantee that the underlying data structures and platform specific optimizations are equally well tuned, especially for the approaches for which we make use of the reference implementation (Panda+, Mdl4bmf and Nassau). Note, however, that due to the formulation of PAL-Tiling in terms of linear algebra, a highly parallel implementation on Graphics Processing Units (GPU) is straightforward. Therefore, experiments regarding Panpal and Primp are executed on a GPU with 2688 arithmetic cores and 6GiB GDDR5 memory. The run time of the GPU based algorithms is about 50 times lower, compared to the ordinary implementations, e.g., a task that is finished by Primp in a few seconds, requires 30 minutes by Mdl4bmf. We provide the source code of our algorithms together with the data generating script[3].

[1] http://hpc.isti.cnr.it/~claudio/web/archives/20131113/index.html
[2] http://people.mpi-inf.mpg.de/~skaraev/
[3] http://sfb876.tu-dortmund.de/primp

ALGORITHM 9 Generation of synthetic datasets for Boolean matrix factorizations.

FUNCTION GENERATENOISYBMF($n, m, r^*, \xi_{max}, p^*_+, p^*_-$)
  $X^* \leftarrow$ UNIFORMRAND($\left\{X \in \{0,1\}^{n\times r} \middle| 0.01 \leq \frac{|X_{\cdot s}|}{n} \leq \xi_{max}\right\}$)
  $Y^* \leftarrow$ UNIFORMRAND($\left\{Y \in \{0,1\}^{m\times r} \middle| 0.01 \leq \frac{|Y_{\cdot s}|}{m} \leq \xi_{max}\right\}$)
  $D \leftarrow Y^* \odot {X^*}^\top$
  FOR $1 \leq j \leq m, 1 \leq i \leq n$ **DO**
    IF $D_{ji} = 1$ **THEN** $D_{ji} \leftarrow 0$ with probability p^*_-
      ELSE $D_{ji} \leftarrow 1$ with probability p^*_+
  END FOR
  RETURN (X^*, Y^*, D)
END FUNCTION

6.3.1 Experiments on Synthetic Data

We generate data matrices according to the scheme established by Miettinen and Vreeken (2014); Karaev et al. (2015) and Lucchese et al. (2014). Yet, we constrain the set of generated factor matrices to contain at least one percent of uniquely assigned ones to ensure linear independence of column vectors. This ensures that for $r^* \leq n, m$ and generated matrices $X^* \in \{0,1\}^{n\times r^*}$ and $Y^* \in \{0,1\}^{m\times r^*}$, the matrix $D = Y^* {X^*}^\top$ indeed has rank r^*. We describe the data generation process as a function from dimensions n and m, rank r^*, density parameter ξ_{max} and noise probabilities p^*_+ and p^*_- as stated in Algorithm 9. We generate datasets for distinct settings with dimensions $(n, m) \in \{(500, 1600), (1600, 500), (800, 1000), (1000, 800)\}$, $r \in [5, 25]$, $\xi_{max} \in [0.1, 0.3]$ and $p^*_\pm \in [0, 0.25]$. Table 6.1 summarizes the basic statistics of the generated datasets.

EVALUATION We quantify how well a computed clustering (X, Y) matches the planted clustering (X^*, Y^*) by the F-measure, as introduced in Section 4.3.1. We assume w.l.o.g. that $X, X^* \in \{0,1\}^{n\times r}$ and $Y, Y^* \in \{0,1\}^{n\times r}$, otherwise we attach zero columns to the matrices such that the dimensions match. We require the following substitutions of precision and recall, in order to take the selection of features in a tile into account:

$$\mathrm{pre}(s,t) = \frac{\left|(Y^*_{\cdot s} \circ Y_{\cdot t})(X^*_{\cdot s} \circ X_{\cdot t})^\top\right|}{\left|Y_{\cdot t} X_{\cdot t}^\top\right|} \qquad \mathrm{rec}(s,t) = \frac{\left|(Y^*_{\cdot s} \circ Y_{\cdot t})(X^*_{\cdot s} \circ X_{\cdot t})^\top\right|}{\left|Y^*_{\cdot s} {X^*_{\cdot s}}^\top\right|}.$$

Variation	$p_+^*[\%]$	$p_-^*[\%]$	r^*	ξ_{max}	Density [%]	Overlap [%]
Uniform Noise	0	0	25	0.1	6.6 ± 0.8	2.3 ± 0.3
	25	25	25	0.1	28.3 ± 0.4	2.3 ± 0.3
Pos/Neg Noise	25	3	25	0.1	30.6 ± 0.4	3.0 ± 0.4
	3	25	25	0.1	8.5 ± 0.4	2.9 ± 0.5
Rank	10	10	5	0.1	11.3 ± 0.4	0.3 ± 0.2
	10	10	45	0.1	18.6 ± 0.9	8.7 ± 1.0
Tile Size	10	10	25	0.1	15.3 ± 0.6	2.3 ± 0.3
	10	10	25	0.3	40.6 ± 4.3	26.9 ± 6.1

TABLE 6.1: Characteristics of generated datasets. The values are aggregated over eight generated datasets, four for each combination of dimensions $(n, m) \in \{(500, 1600), (800, 1000)\}$. Overlap denotes the percentage of overlapping entries in relation to the region covered by all tiles together and density is the region covered by all tiles together in relation to nm.

Given the tile-matching function σ, the calculation of precision and recall for the whole factorization is accordingly specified by

$$\text{pre} = \frac{\sum_{s=1}^{r} \left| (Y_{\cdot s}^* \circ Y_{\cdot \sigma(s)})(X_{\cdot s}^* \circ X_{\cdot \sigma(s)})^\top \right|}{\sum_{s=1}^{r} \left| Y_{\cdot s} X_{\cdot s}^\top \right|} = \frac{\left| (Y^* \circ Y_{\cdot \sigma(\cdot)})(X^* \circ X_{\cdot \sigma(\cdot)})^\top \right|}{\left| Y X^\top \right|}$$
$$\text{rec} = \frac{\sum_{s=1}^{r} \left| (Y_{\cdot s}^* \circ Y_{\cdot \sigma(s)})(X_{\cdot s}^* \circ X_{\cdot \sigma(s)})^\top \right|}{\sum_{s=1}^{r} \left| Y_{\cdot s}^* X_{\cdot s}^{*\top} \right|} = \frac{\left| (Y^* \circ Y_{\cdot \sigma(\cdot)})(X^* \circ X_{\cdot \sigma(\cdot)})^\top \right|}{\left| Y^* X^{*\top} \right|}.$$

The plots which display the F-measure indicate its average value with error bars having the length of twice the standard deviation. We furthermore express the values of involved cost measures in relation to the empty model

$$\%F(X, Y) = \frac{F(X, Y)}{F(\mathbf{0}, \mathbf{0})} \cdot 100.$$

MAKE SOME NOISE In the following series of experiments, varying the noise, we plot the F-measure and the rank of the returned Boolean factorization against the percentage of noise which is added. The planted factorization has a rank of $r^* = 25$ and density parameter $\xi_{max} = 0.1$. The noise level varies from 0% to 25% as displayed on the x-axis.

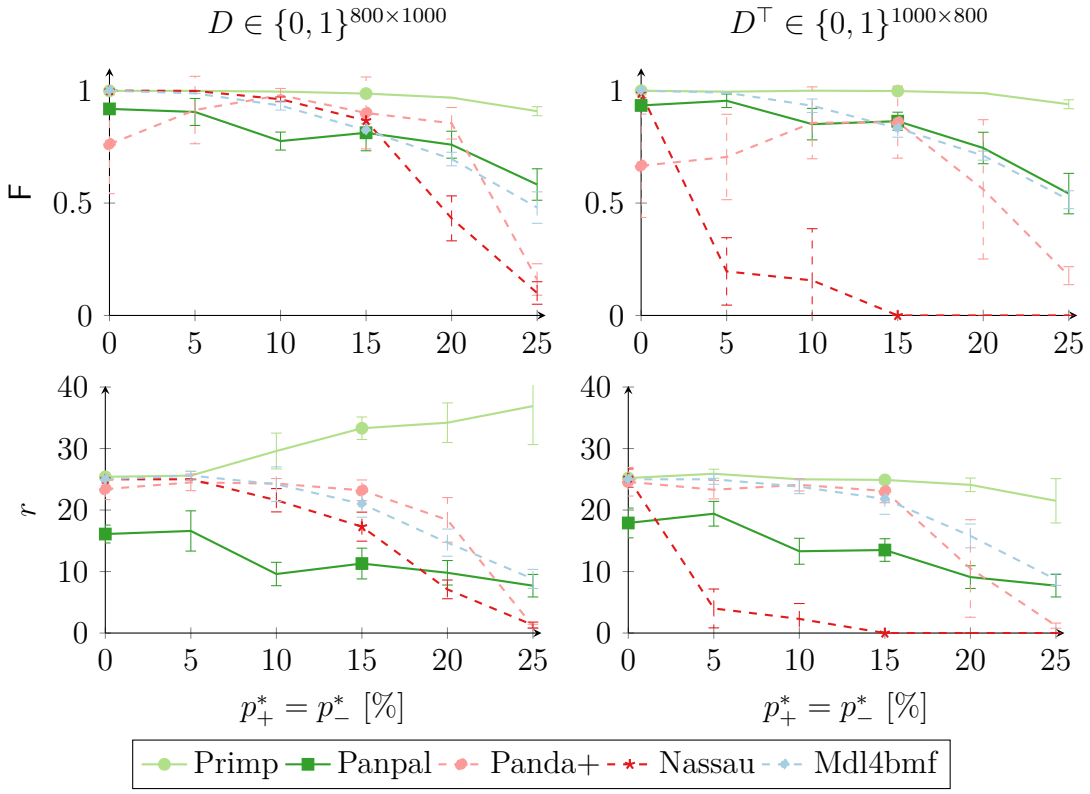

FIGURE 6.1: Variation of uniform noise for 800 × 1000- and 1000 × 800-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., $p^*_+ = p^*_-$ is indicated on the x-axis (best viewed in color).

First, we compare the effects of the matrix dimensions and aggregate results over 10 generated matrices with dimensions 800×1000 and 500×1600 together with their transpose, as depicted in Figures 6.1 and 6.2. Comparing the results for a data matrix and its transpose is particularly interesting for the algorithm PRIMP. Since the description length of code table encoding applies different regularizations on X and Y, we want to asses how transposition affects PRIMP's results in practice. The remaining algorithms minimize an objective which is invariant to a transposition of the input matrix. Desirably, this is also reflected in practice.

We observe from Figures 6.1 and 6.2 that the algorithms likely return fewer tiles with increasing noise. This culminates in the replication of almost none of the clusters at highest noise level for the algorithms PANDA+ and NASSAU. NASSAU particularly strongly underestimates the rank if the data matrix is transposed. That is, if the input matrix is more wide than tall then NASSAU returns only a few tiles, if any, even if the noise is low. PANDA+ yields correct rank estimations up

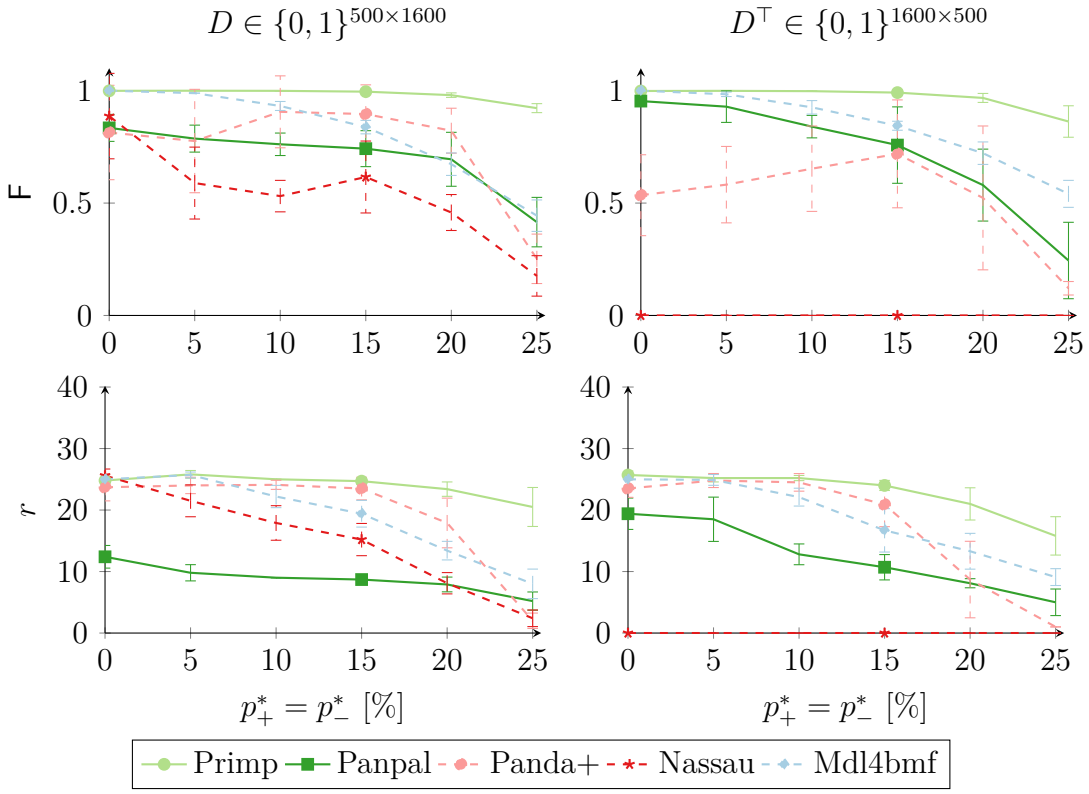

FIGURE 6.2: Variation of uniform noise for 500 × 1600- and 1600 × 500-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., $p_+^* = p_-^*$ is indicated on the x-axis (best viewed in color).

to a noise of 15%, but its fluctuating F-measure indicates that planted tiles are not correctly recovered after all. In particular, its F-values differ from the untransposed to the transposed case even if the rank estimations are approximately correct. MDL4BMF shows a robust behavior towards a transposition of the input matrix. Its suitable rank estimations up to a noise of 15% are mirrored in a high F-measure. PANPAL consistently underestimates the rank, yet achieves comparatively high F-measures. The results exhibit minor deviations from the untransposed to the transposed case, which become more recognizable when n and m deviate further from each other (Figure 6.2) and the noise level is low. Under these circumstances, PANPAL yields higher rank estimations if the matrix is transposed. We note, that the code of PAL-TILING and therewith also the code of PANPAL inhibits only one distinction between X and Y, which is the order in which gradient descent steps are invoked. Apparently, this influences the sparseness of the resulting factor matrices. We propose a thorough examination of this effect for future work. PRIMP

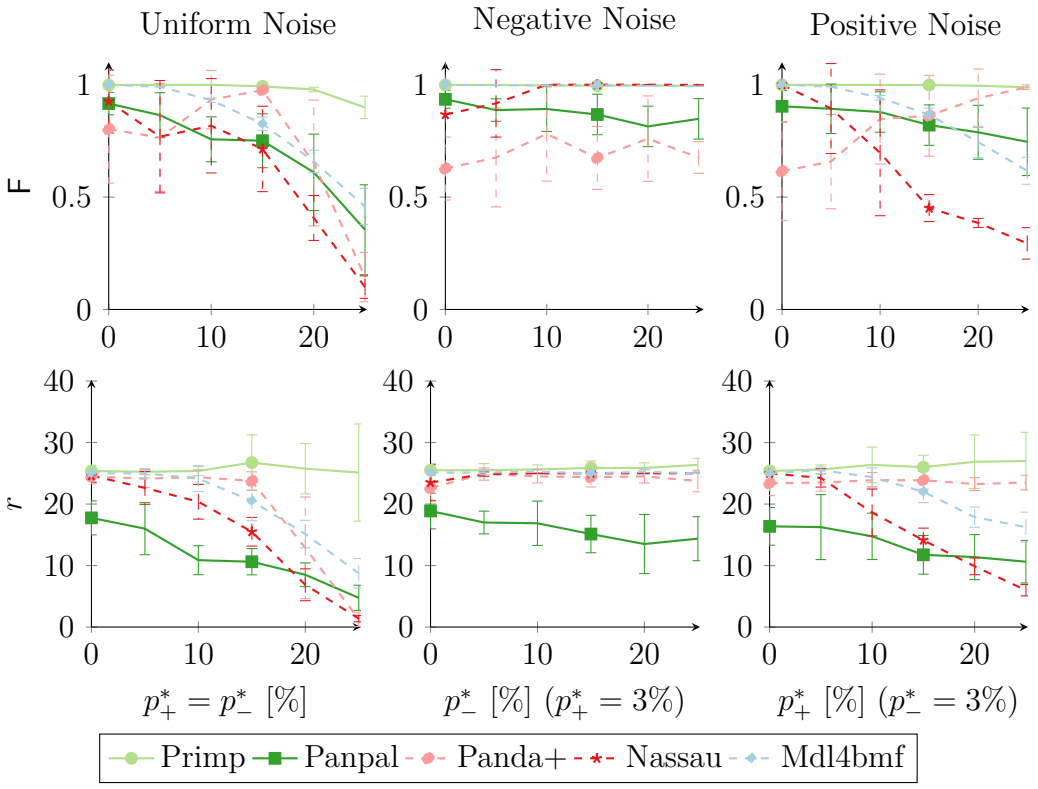

FIGURE 6.3: Variation of uniform, positive and negative noise. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., p^*_+ and p^*_- are indicated on the x-axis (best viewed in color).

is characterized by overall high values in the F-measure. It has a tendency to estimate the rank higher in the untransposed case, that is if the input matrix is more tall than wide. This is particularly notable if the matrices are almost square (Figure 6.1). That aside, the overall high F-measure shows that additionally modeled clusters cover only a small area in comparison to planted ones.

In Figure 6.3 we contrast varying distributions of positive and negative noise (p^*_+ and p^*_-). From here on, we aggregate results over eight matrices, two matrices are generated for each of the considered matrix dimensionalities. However, we make an exception for Nassau and transpose the input matrix if $n > m$, as Nassau tends to return far too few tiles in this case. On the left of Figure 6.3, we show the aggregated results when varying uniform noise, as discussed for individual dimensions before. All algorithms except for Primp tend to return fewer tiles with increasing noise. Notably, Primp's rank estimations are correct in the mean, yet the variance increases with the noise. Despite correct rank estimations, Panda+

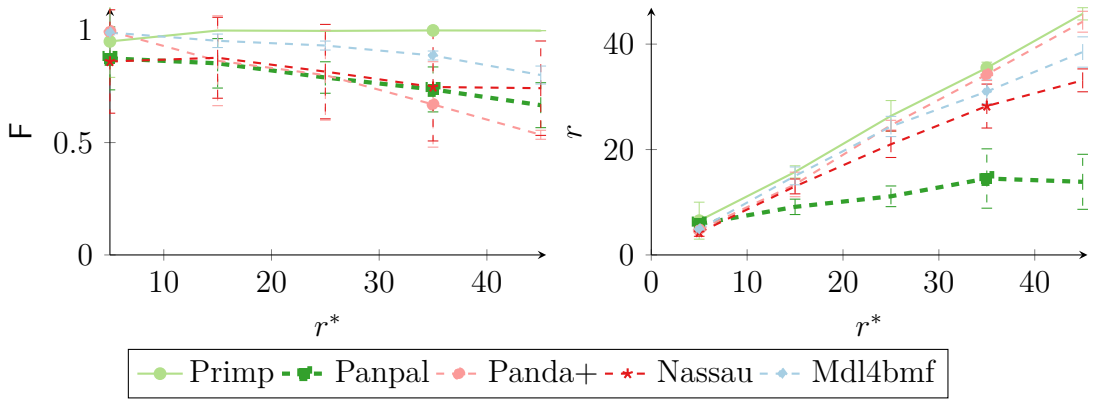

FIGURE 6.4: Variation of the rank $r^* \in \{5, \ldots, 45\}$ of the generated factorization. Comparison of F-measures (the higher the better) and estimated rank (the closer to the identity function the better) of calculated Boolean matrix factorizations for uniform noise of $p^*_+ = p^*_- = 10\%$ (best viewed in color).

displays volatile F-measure values. The middle plot depicts variations of negative noise while positive noise is fixed to 3%. In this setting, the algorithms PRIMP, MDL4BMF and NASSAU are capable of identifying the generated factorization for all noise levels. The suitability of NASSAU in the prevalence of negative noise corresponds to the experimental evaluation by Karaev et al. (2015). MDL4BMF and PRIMP yield equally appropriate results in this experiment. The approximations of PANDA+ and PANPAL are less accurate. Although PANDA+ correctly estimates the rank around 25 and PANPAL's estimations lie between 10 and 20, PANPAL achieves higher F-measures than PANDA+.

The plots on the right of Figure 6.3 show the impact of variations on the positive noise, fixing the negative noise to 3%. Here, NASSAU, MDL4BMF and PANPAL tend to underestimate the rank the more the noise increases, similarly to but not as drastic as in experiments with uniformly distributed noise. PANDA+ shows a poor recovery of the factorization at 0% positive noise, but its F-value peculiarly increases with increasing positive noise. PRIMP robustly identifies the factorization for all levels of noise, yet inhibits a high variance in rank estimations.

SENSITIVITY TO GENERATING PARAMETERS We present effects on variations from the rank in Figure 6.4 whereby the default parameters of 10% uniform noise and maximum tile width and length $\xi_{max} = 0.1$ apply. We observe a hierarchy of algorithms in the tendency to underestimate the rank throughout all values of r^*. By far the lowest rank estimations are returned by PANPAL, followed by NASSAU, MDL4BMF, PANDA+ and PRIMP. PANDA+ and PRIMP consistently return accurate rank estimations. In this respect, it is remarkable that for ranks

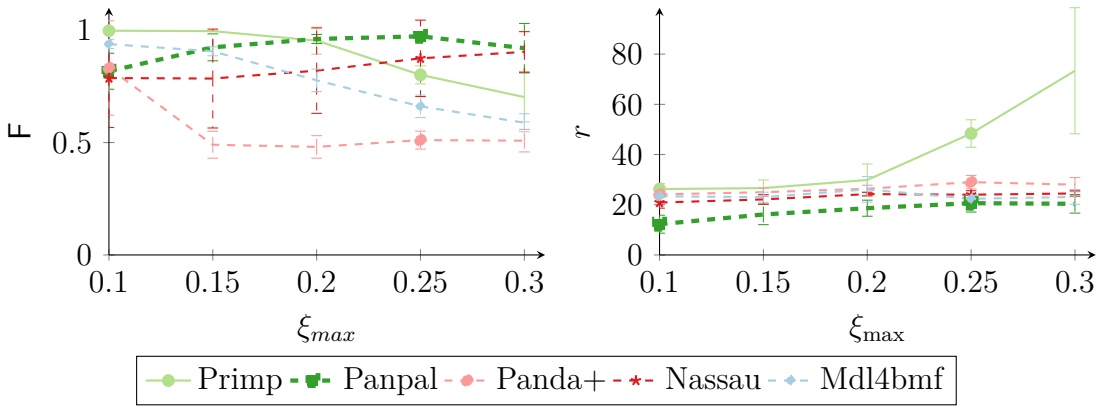

FIGURE 6.5: Variation of density and overlap influencing parameter $\xi_{max} \in [0.1, \ldots, 0.3]$. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for uniform noise of $p_+^* = p_-^* = 10\%$ (best viewed in color).

higher than 30, PANPAL obtains higher F-values than PANDA+ despite of modeling only a fraction of the planted tiles. PRIMP provides a steadily accurate recovery of the factorization.

In Figure 6.5 we vary the density and overlap influencing parameter ξ_{max}, which determines the maximum ratio of ones in a column vector of X and Y. We observe two classes of algorithms. The first class, consisting of PRIMP, PANDA+ and MDL4BMF decreases in the F-measure with increasing ξ_{max}. In this class, PRIMP always retrieves highest F-values. In return, the F-values from the second class of PANPAL and NASSAU increase with ξ_{max}. Here, PANPAL bounds the F-values of NASSAU from above. Correspondingly, PRIMP and PANPAL have a *break-even-point* at $\xi_{max} = 0.2$. From this value on, PRIMP starts to considerably overestimate the rank while PANPAL's tendency to underestimate the rank decreases. For $\xi_{max} \geq 0.2$, PANPAL estimates the rank close to 20 in average. That is, five planted tiles are not modeled in average. Still, the F-measure indicates that for the denser and more overlapping datasets, PANPAL most accurately discovers the clusters.

SENSITIVITY TO THE RANK INCREMENT In the default setting of our synthetic experiments, the PAL-TILING algorithms PRIMP and PANPAL have to increase the rank two times by $\Delta_r = 10$ to estimate the rank $r^* = 25$ correctly. In the experiments varying the rank, we have seen that PRIMP is able to find the correct rank if twice as many decisions correctly have to be made. Here, we want to assess how robust the performance of PAL-TILING algorithms to the parameter Δ_r is. What happens if, e.g., $\Delta_r = 2$ and correspondingly 23 rank increments have to be administered correctly?

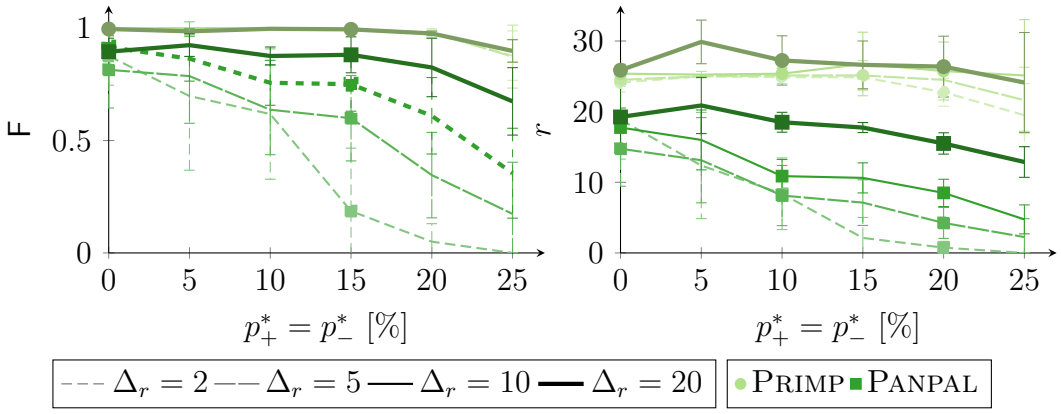

FIGURE 6.6: Variation of rank increment $\Delta_r \in \{2, 5, 10, 20\}$. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for uniform noise of $p^*_+ = p^*_-$ indicated by the x-axis (best viewed in color).

Figure 6.6 shows F-measurements and estimated ranks of the algorithms PRIMP and PANPAL, invoked with diverse rank increments $\Delta_r \in \{2, 5, 10, 20\}$ on datasets with varying uniform noise. It is noticeable that the rank estimations of PANPAL rapidly diverge with increasing noise while the plots of PRIMP stay comparatively close. PANPAL's tendency to underestimate the rank grows for smaller rank increments. In return, the rank estimations of PANPAL can be improved by choosing a large rank increment, i.e. $\Delta_r \approx r^*$. However, since we do not know the rank in real-world applications, different increment values have to be tried and compared, contradicting our goal to automatically determine this parameter. Still, PANPAL yields potentially useful lower bounds on the actual rank.

The average rank estimations of PRIMP have a maximum aberration of five from the actual rank throughout all noise variations. The graphical display of the estimated rank for $\Delta_r = 20$ has a peak at 5% uniform noise but is close to r^* otherwise. For rank increments smaller than 10, the estimations do not distinctly decrease until the noise exceeds 20%. Here, a rank increment of $\Delta_r = 5$ yields the most accurate rank estimations, having also lowest standard deviations from the mean. Particularly, PRIMP's tendency to overestimate the rank in specific settings can be corrected by choosing smaller rank increments. Nonetheless, all these rank deviations barely effect the F-measure, which demonstrates the robust and well fitted recovery of the underlying model regardless of the choice of rank increment.

COMPARISON OF DESCRIPTION LENGTHS We have seen how well the competing algorithms perform with regard to the F-measure. Then again, assessing the performance on real data requires other measurements. Possible candidates

	Algorithm	F-measure	$\%RSS$	$\%L_{\mathsf{CT}}$	$\%L_{\ell 1}$	$\%L_{\mathsf{TXD}}$
$p^*_\pm = 25\%$	Planted	1.0 ± 0.0	88.37 ± 1.24	89.88 ± 1.25	89.51 ± 1.25	96.5 ± 0.61
	PRIMP	**0.9±0.05**	**89.58±1.71**	**91.0±1.54**	**90.65±1.59**	**97.0±0.62**
	PANPAL	0.35 ± 0.2	97.17 ± 1.62	97.56 ± 1.46	97.47 ± 1.49	99.16 ± 0.56
	MDL4BMF	0.46 ± 0.08	96.6 ± 0.86	97.25 ± 0.76	97.11 ± 0.77	99.2 ± 0.25
	PANDA	0.14 ± 0.11	99.14 ± 0.77	99.28 ± 0.62	99.24 ± 0.66	99.75 ± 0.19
	NASSAU	0.1 ± 0.05	100.5 ± 0.29	100.7 ± 0.3	100.69 ± 0.31	99.75 ± 0.15
$r^* = 45$	Planted	1.0 ± 0.0	50.27 ± 1.25	54.67 ± 1.29	53.29 ± 1.29	69.77 ± 0.96
	PRIMP	**1.0±0.0**	**50.32±1.23**	**54.74±1.27**	**53.35±1.27**	**69.85±0.93**
	PANPAL	0.67 ± 0.1	73.0 ± 6.04	75.04 ± 5.56	74.29 ± 5.71	84.66 ± 3.78
	MDL4BMF	0.8 ± 0.04	62.67 ± 1.41	66.72 ± 1.53	65.48 ± 1.46	79.21 ± 1.03
	PANDA	0.53 ± 0.02	89.02 ± 1.76	92.34 ± 2.16	92.76 ± 2.09	86.0 ± 0.7
	NASSAU	0.74 ± 0.21	64.43 ± 10.08	68.27 ± 10.24	67.28 ± 10.43	77.47 ± 4.87
$\xi_{max} = 0.3$	Planted	1.0 ± 0.0	24.94 ± 2.74	27.84 ± 2.78	27.11 ± 2.7	51.97 ± 1.04
	PRIMP	0.7 ± 0.1	**27.04±2.46**	**31.23±2.33**	**30.33±2.25**	57.52 ± 2.03
	PANPAL	**0.92±0.11**	29.45 ± 3.17	31.98 ± 3.06	31.31 ± 3.1	57.09 ± 3.86
	MDL4BMF	0.59 ± 0.04	45.08 ± 2.14	48.53 ± 2.01	47.88 ± 1.96	73.81 ± 1.49
	PANDA	0.51 ± 0.05	54.12 ± 8.9	57.07 ± 8.83	56.74 ± 8.86	75.88 ± 2.32
	NASSAU	0.9 ± 0.09	29.11 ± 5.19	32.07 ± 5.2	31.42 ± 5.2	**56.79±4.2**

TABLE 6.2: Average values of objective functions of computed and planted models, denoted in relation to the objective function of the empty model. For each setting (variation of one data generation parameter while the others are set to default values $r^* = 25$, $\xi_{max} = 0.1$ and $p^*_\pm = 10\%$) the average value is computed over all considered dimension variations.

are the optimized description lengths. Subsequently, we relate selected costs of computed and planted models to the F-measure and discuss whether we can deduce a suitable extraction of the underlying model from a low cost measure; is lower always better?

Table 6.2 displays average costs of the returned factorization, measured by the functions RSS, the residual sum of squares, L_{CT}, the compression size obtained by code tables, $L_{\ell 1}$, the $\ell 1$-regularized residual sum of squares and L_{TXD}, the Typed XOR DtM (TXD) measure. All these measurements are listed in relation to the empty model. We examine three parameter settings, one for the highest value in each variation of the data generation parameters r^*, ξ_{max} and $p^*_\pm$. Thereby, default settings of $r^* = 25$, $\xi_{max} = 0.1$, $p^*_\pm = 10\%$ apply. The costs of the planted model are shaded out while the highest F-measure and lowest mean costs of computed models are highlighted.

We trace that a high F-measure often corresponds to lower costs, regardless of the cost function. This effect is immediately perceivable at rows where PRIMP attains highest F-values and all of its costs are highlighted as well. Yet, the exper-

Dataset D	m	n	Density [%]
Abstracts	859	4977	1.02
Mushroom	8124	120	19.33
MovieLens5M	29980	9044	1.81
MovieLens500K	3329	3015	4.99
Chess	3196	75	49.33

TABLE 6.3: Characteristics of considered datasets: Number of rows m, number of columns n and density $|D|/(nm)$ in percent.

iments for $\xi_{max} = 0.3$ display a more diverse ranking among the measurements. In this setting, PRIMP decidedly overestimates the rank but still obtains lowest costs in all but the L_{TXD} measure. PANPAL attains the highest F-value, closely followed by NASSAU. Both algorithms reach second or third lowest cost measurements by the functions RSS, L_{CT} and $L_{\ell 1}$. The TXD description length reflects here the order of F-values more suitably but still not accurately; NASSAU obtains lowest costs, closely followed by PANPAL and PRIMP. All in all, none of the description lengths reflects the ranking of the F-measure in this particular case. If the F-measure would be unknown, then a possible clue would be given by the estimated rank, showing that slight improvements in the costs of PRIMP are achieved by a disproportionate increase of the rank.

While the TXD costs appear suitable to reflect an appropriate extraction of tiles for $\xi_{max} = 0.3$, in the setting of $p^*_{\pm} = 25\%$ we observe another facet. Here, we see that NASSAU reaches the same average TXD measurement as PANDA+ although NASSAU increases the residual sum of squares in comparison to the empty model. This is indicated by relative costs larger than 100% in all measurements but TXD. Still, the ranking with respect to TXD matches the ranking of the F-measure, but the example shows that a compression with respect to the TXD description length is achievable without adaptation to the data.

6.3.2 Real-World Data Experiments

We conduct experiments on five datasets, whose characteristics are summarized in Table 6.3. *Chess* and *Mushroom* are discretized benchmark UCI datasets having a comparatively high density and around 50 times more rows than columns. The *Abstracts* dataset indicates the presence of stemmed words, excluding stop-words, in all ICDM paper abstracts until 2007 (De Bie, 2011). It is a sparse dataset with around 5 times as many columns (words) as rows (documents). Finally, the *Movie-*

Lens5M and *MovieLens500K* are binarized versions of the MovieLens10M[4] and MovieLens1M[5] datasets, where rows correspond to users and columns to movies. We set $D_{ji} = 1$ iff user j recommends movie i with more than three out of five stars. After selecting only those users which recommend more than 50 movies and those movies which receive more than five recommendations, we obtain two datasets with a balanced number of of rows and columns. The MovieLens5M dataset, containing 5 million ones, and the MovieLens500K dataset, having 500 thousand ones, have a (as one would expect, due to the dataset domain) high amount of negative noise due to missing values. Originally, we intended to consider only the MovieLens5M dataset, but NASSAU and MDL4BMF could not terminate in reasonable time – we aborted the calculations after one month. Therefore, we also prepared the smaller MovieLens500K dataset. (For comparison: While PRIMP, PANPAL and PANDA+ require around ten minutes to compute the result for MovieLens500K, MDL4BMF and NASSAU need more than five days.) Furthermore, we note that we transpose the Abstracts dataset for NASSAU, as it returns a rank of zero otherwise.

We state the estimated rank and the attained costs, relative to the costs of the empty model for every considered dataset and algorithm in Table 6.4. The lowest costs are highlighted for each measure and dataset. We observe, similarly to the evaluation of description lengths in Section 6.3.1, a tendency toward compliance among the rankings of all measures, except for the TXD description length. As such, PRIMP mostly obtains minimal costs in all datasets but Mushroom, where MDL4BMF reaches the lowest overall cost measures. The models of PANDA+ exhibit for sparse datasets low TXD measurements although the fit to the data is low ($\%RSS > 100\%$). The discrepancy between the TXD compression size and the other measurements is most remarkably for the MovieLens datasets. Here, the ranking with respect to TXD is almost inverse to the ranking with respect to other measurements.

This leads to the question which measurement indicates the suitable factorization in such situations? Luckily, we have for the MovieLens data the possibility to assess how many recommendations would fail by the submitted bad reviews, which are not reflected in the input data. We state the relative amount of recommendations which correspond to bad reviews, $|D_- \circ \theta(YX^\top)| / |D_-|$ where D_- is the matrix having $D_{ji} = 1$ iff user j rates movie i with less than 2.5 of five stars, in Table 6.5. We observe that the lower the TXD description length is, the higher is the rate of recommendation failures, regardless of the estimated rank. Therefore, we expect PRIMP to discover the most liable grouping of users and movies, having the lowest

[4] `http://grouplens.org/datasets/movielens/10m/`
[5] `http://grouplens.org/datasets/movielens/1m/`

Data		Algorithm	Rank	$\%RSS$	$\%L_{\mathsf{CT}}$	$\%L_{\ell 1}$	$\%L_{\mathsf{TXD}}$
Abstracts		PRIMP	46	**93.0**	**98.6**	**96.33**	96.12
		PANPAL	1	99.8	99.96	99.89	99.74
		MDL4BMF	24	95.84	100.68	100.49	97.05
		NASSAU	3	99.81	101.76	103.05	96.84
		PANDA+	133	113.34	125.27	140.49	**88.19**
Chess		PRIMP	18	**24.61**	**31.3**	**29.32**	**62.8**
		PANPAL	6	40.76	46.71	45.67	78.92
		MDL4BMF	3	35.91	39.34	39.51	68.88
		NASSAU	10	31.92	39.03	38.94	65.78
		PANDA+	27	25.76	36.58	35.74	65.01
MovieLens	500K	PRIMP	78	**88.59**	**93.29**	**91.4**	89.37
		PANPAL	15	94.05	95.92	94.87	92.26
		MDL4BMF	56	89.65	94.97	93.43	88.72
		NASSAU	29	111.15	118.47	120.58	85.89
		PANDA+	120	160.93	165.61	168.58	**79.49**
	5M	PRIMP	209	**89.31**	**93.14**	**91.2**	88.16
		PANPAL	38	93.68	95.72	94.39	88.73
		PANDA+	1919	181.42	202.87	201.45	**72.23**
Mushroom		PRIMP	14	35.75	40.89	40.25	56.09
		PANPAL	7	44.03	51.23	48.52	63.75
		MDL4BMF	87	**23.39**	**36.6**	**32.47**	**50.37**
		NASSAU	65	40.60	58.77	54.93	50.62
		PANDA+	40	100.30	117.34	112.80	66.98

TABLE 6.4: Comparison of cost measures for real-world datasets.

approximation error and a very low ration of traceable wrong recommendations. Similarly, it is questionable if low TXD costs indicate suitable models in specific cases where the approximation error diverges such as for the *Abstracts* dataset.

6.3.3 Qualitative Inspection of Mined Tiles

The F-measure gives a hint at the kind of factorization we can expect from the algorithms. PANPAL returns a coarse view, modeling only a few tiles which match actually persistent ones. The quality of the results of PANDA+ substantially varies and MDL4BMF and particularly PRIMP are most often able to identify the persis-

MovieLens	PRIMP	PANPAL	MDL4BMF	NASSAU	PANDA+
500K	2.38	2.23	3.68	10.33	18.78
5M	2.08	2.78	-	-	23.14

TABLE 6.5: Percentage of traceable wrong recommendations of computed models for the MovieLens datasets, i.e., the relative amount of user-movie recommendations which correspond to bad reviews (< 2.5 stars out of five).

tent interrelations. Yet how do the algorithms relate in their actual cognition of structure and noise, how does the optimization influence the computed decomposition?

Image data allows us to visually inspect the resulting factorizations without the need to specify a numeric measure. We can intuitively assess the attempts to capture relevant sub-structures. However, some preprocessing is required in order to feed $w \times h$ images to the mining algorithms. We employ a standard representation of images: the RGB888 pixel format. Each of the $w \cdot h$ pixels is represented by 24 bits, using 8 bits per color (red, green and blue). In order to convert an image into a set of transactions, we divide it into blocks (patches) of $4 \cdot 4$ pixels, resulting in a total of $\frac{w}{4} \cdot \frac{h}{4}$ transactions per image. We adopt this representation from computer vision, where image patches are a standard preprocessing step for raw pixel data (Jarrett et al., 2009). Within each block, let $(r, g, b)_{l,k}$ denote the pixel at row l and column k, where $r, g, b \in \{0, 1\}^8$ are the 8-bit binary representation of its red, green and blue color values. We model the concatenation of all 16 pixels within one block as one transaction

$$[(r, g, b)_{1,1}, (r, g, b)_{1,2}, (r, g, b)_{1,3}, (r, g, b)_{1,4}, (r, g, b)_{2,1}, \ldots, (r, g, b)_{4,4}]$$

which has a length of $24 \cdot 16 = 384$ bits.

This way, we process two images: an illustration of *Alice* in Wonderland (Figure 6.7) and a selection of "aliens" from the classic game *Space Invaders* (Figure 6.8). We select Alice because the image contains multiple connected areas, each representing a reasonable substructure, i.e., hair, face, dress, arm and background. In return, the Space Invaders image contains multiple patterns in terms of color and shape, but the components are clearly spatially separable.

The original Alice image, as well as reconstructions $\theta(XY)$ and the top-4 tiles generated by NASSAU, MDL4BMF, PANDA+, PANPAL and PRIMP, are depicted in Figure 6.7. Clearly, only PANDA+ and PRIMP select patterns, i.e., blocks of pixels which provide a reasonable reconstruction of the original image. PANPAL's tendency to underestimate the rank (choosing only three tiles) becomes apparent here again. Regarding the figured structures, PANDA+, PANPAL and PRIMP discover a hair-related substructure, where the one found by PRIMP has the most

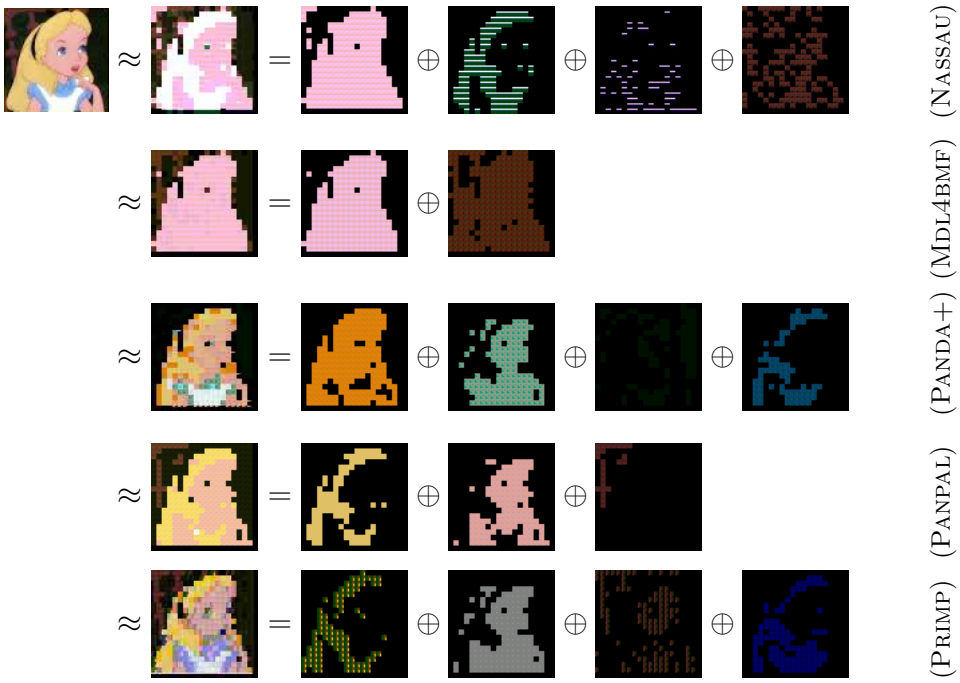

FIGURE 6.7: Reconstructions of the Alice image and visualizations of the top-4 outer products. Best viewed in color.

distinctive contours, and Panda+, Panpal and Primp identify a face-related structure. The reconstructions and factors found by Nassau and Mdl4bmf are not easy to interpret without knowledge of the original image.

Reconstruction results and top-4 tiles of the Space Invaders image are shown in Figure 6.8. All methods reconstruct at least the shape of the aliens. In terms of color, however, the results diverge. Panda+ and Nassau interpret all colors as negative noise effects on the color white; white has a binary representation of 24 ones. Panpal recovers the yellow color correctly and it extracts the full blue channel from the image—an identical pattern is also detected by Primp. Primp and Mdl4bmf reconstruct all three colors of the original image, yet the reconstruction of Mdl4bmf exhibits injections of white blocks. Only Primp is capable to reconstruct the color information correctly.

Having a look at derived tiles, the greedy processes of Panda+ and Nassau become particularly visible; Panda+ and Nassau overload the first factor with all the shape information. The remaining factors reduce the quantitative reconstruction error, but have no deeper interpretation. Mdl4bmf tries to model one type of aliens by each tile. Although this would result in a reasonable description

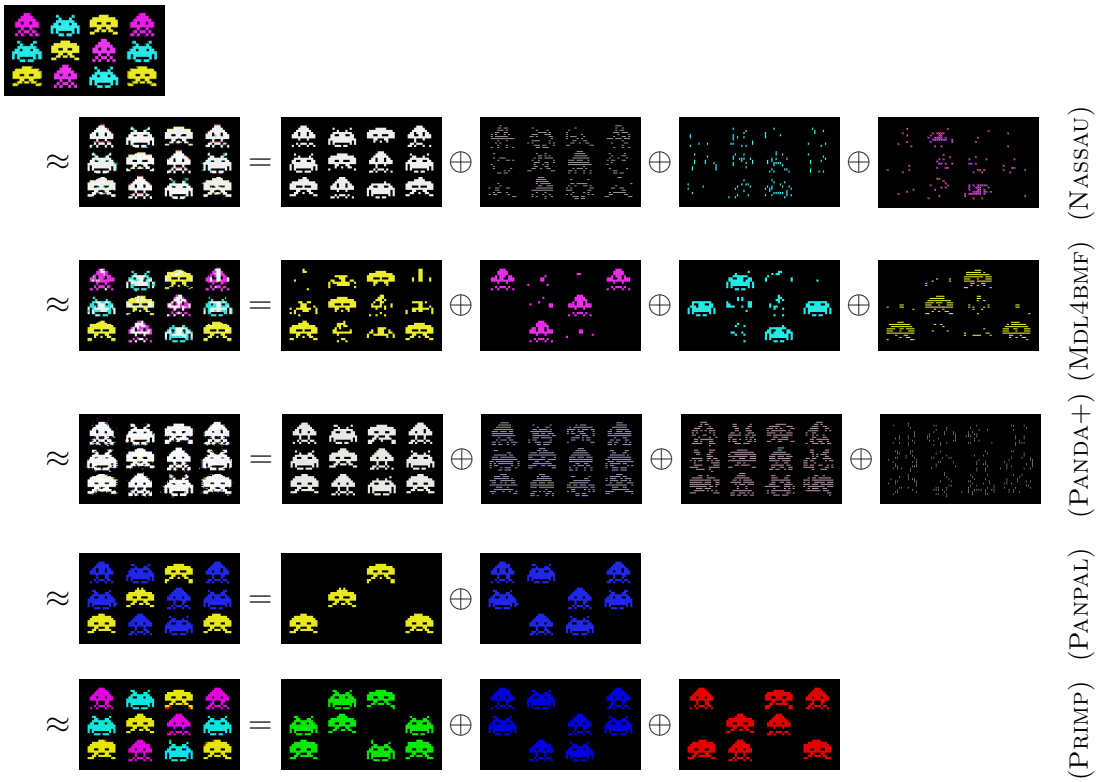

FIGURE 6.8: Reconstructions of the Space Invaders image and visualizations of the top-4 outer products. Best viewed in color.

of the image, the actual extraction of tiles suffers from the greedy implementation. We can see that, e.g., the first tile captures information about the yellow aliens as well as strayed parts of other aliens. This unfortunate allocation of tiles results in the injection of white blocks in the reconstruction image. Panpal clearly separates yellow and blue aliens but interprets differences from the color blue to purple and to turquoise as noise. Finally, Primp separates by its tiles the three basic color channels which are actually used to mix the colors that appear in the original image. Hence, Primp achieves the factorization rank that corresponds to the natural amount of color concepts in the image, unlike all other competitors.

The results of this qualitive experiment particularly illustrates the benefits of a non-greedy minimization procedure. Even though Panpal is often not able to minimize the costs due to an underestimation of the rank, its categorization into tiles always yields interpretable parts.

6.4 Discussion

We introduce the minimization of description lengths for the derivation of Boolean matrix factorizations with an automatically determined rank by PAL-Tiling. Requiring that the description length has a smooth relaxed function, which combines the matrix factorization error with a regularizing function, PAL-Tiling minimizes the relaxed objective under convergence guarantees.

Aiming at the robust identification of Boolean matrix factorizations in presence of various noise distributions, we consider two description lengths in this framework which defines two factorization algorithms. The first algorithm uses a simple $\ell 1$-norm regularization on the factor matrices and is called Panpal. The second minimizes the MDL-description length of the encoding by code tables as known from Krimp (Siebes et al., 2006). Foregoing the heuristics in computing the usage of codes, we extend the application of this encoding from pattern mining to Boolean matrix factorization and derive an upper bound which induces the relaxed objective. We refer to this instance of PAL-Tiling as Primp.

Our experiments on synthetically generated datasets show that the quality of competing algorithms Panda+, Mdl4bmf and Nassau is sensitive towards multiple data generation parameters. The first of the two newly introduced algorithms, Panpal, regularly underestimates the true factorization rank. We have seen that this property can be beneficial in settings with large, overlapping tiles which induce dense datasets (cf. Figure 6.5). In all other settings, the second algorithm Primp is able to detect the underlying structure, regardless of the considered distribution of noise or variations the factorization rank (cf. Figures 6.1-6.4).

A comparison of cost measures, as provided by the description lengths, on real-world datasets show that Primp also most often achieves lowest costs (cf. Table 6.4). With experiments based on images, we visualize the derived tiles under presence of ambiguous factorization structures and special noise distributions (cf. Figures 6.7 and 6.8). The quality of the reconstruction by established algorithms varies considerably between both images. On the contrary, Panpal and Primp provide solid representations of the original images. The extracted factors reveal a parts-based decomposition of the data (as known from nonnegative matrix factorizations), which allows for interpretation of the results. In the Space Invaders image (cf. Figure 6.8), Panpal partitions the space invaders into those with a blue component in their color (tile 2) and those without blue components (tile 1). On the other hand, Primp divides the space invaders by the primary colors they contain (repeating each space invader exactly twice, hence finding structure in the data too, albeit a different structure from the one found by Panpal). From the Alice image (cf. Figure 6.7) particularly Primp manages to extract coherent factors representing the hair (tile 1) and the face (tile 3).

The implementation of the other popular cost measure, the TXD description length, is not readily realizable in PAL-Tiling. A real-valued relaxation of this description length involves the gamma function, which is not definable and thus, we cannot say whether this description length is a Kurdyka-Łojasiewicz function. Therefore, convergence to a local extremal point of the relaxed TXD function is not guaranteed in PALM. However, other description lengths are possibly worth exploring. For instance, a symmetric regularization of factor matrices appears to be more suitable in the scope of BMF, opposed to the asymmetric regularization implemented by Primp. The formulation of a description length which equally penalizes the model complexity in an encoding by code tables could simplify the implementation effort of Primp while maintaining the ability to suitably select the rank.

CHAPTER 7

BMF Rank Selection by the False Discovery Rate

Often enough in explorative data mining, the user is left alone with the result; a bunch of groupings which supposedly expresses the underlying relations in the dataset. The absence of quality guarantees is an eyesore for the painstaking data miner. Whenever data is collected from an imperfect (noisy) channel—arising from tainted or inaccurate measurements, or transmission errors—the method of choice might be fooled by the noise, resulting in phantom patterns which actually don't exist in the data. Thus, the investigation of *trustworthiness* of data mining techniques is important in practice. While some approaches for the supervised setting exist, e.g., significant pattern mining (Llinares-López et al., 2015), statistical emerging pattern mining (Komiyama et al. (2017) and references therein), insights for the unsupervised case are still missing.

In the scope of Boolean matrix factorization, if some of the modeled tiles emerge from noise, false discoveries happen and the algorithm overfits. We prove two bounds on the probability that a found tile is constituted of random Bernoulli-distributed noise. Both allow us to exploit specific properties of a tile, resulting in different strengths for different types of input data. The bounds require an additional input from the user: an estimate to the positive noise level p_+^*. While this might seem to be a prohibitive burden, our experimental results show that a rough estimate suffices—the user should merely know if her data is pretty noisy or not so much.

We have seen in the last chapter how state-of-the-art BMF techniques filter the structure from the noise and estimate the rank by employing rather complicated regularization terms originating from the description lengths when encoding the data. The belief in the correctness of these algorithms is based on empirical evaluations, inter alia, experiments with synthetically added Bernoulli-distributed

noise. Under this noise assumption, we employ our bounds to devise a new, theoretically well-founded, rank estimation strategy. Since our technique may be used as a plug-in replacement for existing heuristics, our findings improve a whole class of BMF algorithms. Our main contributions are:

- the first provision of bounds on the probability that a tile with specified properties is generated from random noise;
- the validation of required properties for factorizations which minimize the approximation error, showing that both bounds are non-trivial;
- the exemplification of algorithmic use of the bounds for automatic rank selection and the empirical evaluation on synthetic and real-world datasets—a step towards trustworthy data mining.

7.1 False Discoveries in Boolean Matrix Factorization

The generation of noise depends on two parameters; the probability to flip a zero to a one p_+^* and the probability to flip a one to a zero p_-^*. We refer to the first kind of noise as positive noise and to the latter as negative noise. The Boolean decomposition, where we keep positive and negative noise apart looks as follows:

$$D = \left(Y^* \odot X^{*\top} \oplus N_+\right) \circ \left(\overline{Y^* \odot X^{*\top}} \oplus \overline{N_-}\right). \tag{7.1}$$

The factor matrices $X^* \in \{0,1\}^{n \times r}$ and $Y^* \in \{0,1\}^{m \times r}$ denote the *true* model and $N_+, N_- \in \{0,1\}^{m \times n}$ are the binary positive and negative noise matrices.

The first step towards trustworthy pattern mining is a measure of trustworthiness. The False Discovery Rate (FDR) (Benjamini and Hochberg, 1995) is a simple yet powerful way to measure the relative amount of hypothesis rejections which have gone awry.

DEFINITION 7.1 (FDR). Let $\mathcal{H}$ be a finite set of null hypotheses. We define the random variable $\mathbf{Q}$ to denote the ratio between false discoveries and discoveries in total, that is the number of erroneously rejected null hypotheses divided by the number of rejected null hypotheses. We say *the FDR is controlled at level* q if

$$\mathbb{E}(\mathbf{Q}) \leq q.$$

In our setting, a null hypothesis states that the intersections of columns and rows indicated by a tile $yx^\top$ do not reflect underlying relations. In other words, we say that the hypothesis H_s^0 is true when there is no overlap between the underlying model $Y^*X^{*\top}$ and the s-th tile $Y_{\cdot s}X_{\cdot s}^\top$. This definition of a null hypothesis might seem too restrictive for some applications. Therefore, we discuss possible relaxations of this requirement in Section 7.1.2. Bearing this in mind, we see that a BMF of rank r corresponds to a joint rejection of r null hypotheses $\{H_1^0, H_2^0, \ldots, H_r^0\}$. Thus, if the correct rank is r^*, any rank $r > r^*$ factorization is likely to state some erroneous rejections of null hypotheses, a.k.a. false discoveries.

Now, given factor matrices $X \in \{0,1\}^{n\times r}$ and $Y \in \{0,1\}^{m\times r}$, we define a random variable $\mathbf{Z}_s$ with domain $\{0,1\}$, which takes the value 1 if and only if the null hypothesis H_s^0 is not to be rejected, i.e., the outer product $Y_{\cdot s}X_{\cdot s}^\top$ covers only (or mostly – depending on the definition) noise. The FDR of a BMF is therefore computed via

$$\mathbb{E}(\mathbf{Q}) = \frac{1}{r}\sum_{s=1}^{r} P(\mathbf{Z}_s = 1) \;\leq q \text{ if } P(\mathbf{Z}_s = 1) \leq q. \tag{7.2}$$

7.1.1 Tiles in Bernoulli Matrices

We aim at assessing the probability $P(\mathbf{Z}_s = 1)$. Therefore, we need to employ an independence assumption on the noise.

DEFINITION 7.2 (BERNOULLI MATRIX). Let $\mathbf{B}$ be an $m \times n$ random matrix whose entries are i.i.d copies of a Bernoulli (p) random variable, that is

$$P(\mathbf{B}_{ji} = 1) = p,\; P(\mathbf{B}_{ji} = 0) = 1 - p\,,$$

then we say $\mathbf{B}$ is a *Bernoulli*(p) *matrix.*

In what follows, we assume that the positive noise matrix N_+ is a realization of a Bernoulli (p_+^*) matrix. If a tile $Y_{\cdot s}X_{\cdot s}^\top$ does not approximate the model $Y^*X^{*\top}$ then it has to have some overlap with the positive noise matrix, otherwise the tile wouldn't contribute to minimizing the data approximation error. The *overlap* is computed as the sum of common 1 entries

$$\left|Y_{\cdot s}X_{\cdot s}^\top \circ N_+\right| = \left\langle Y_{\cdot s}X_{\cdot s}^\top, N_+\right\rangle = \operatorname{tr}\left(X_{\cdot s}Y_{\cdot s}^\top N_+\right) = Y_{\cdot s}^\top N_+ X_{\cdot s}\,.$$

This quantity is used to determine the density of a tile in the positive noise matrix.

DEFINITION 7.3 (δ-DENSE). Let A be an $m \times n$ binary matrix and $\delta \in [0,1]$. We say a tile or binary outer product $yx^\top$ is δ-*dense in* A if

$$y^\top Ax \geq \delta|x||y|\,.$$

Since a Boolean matrix product, approximating the data matrix well, covers a high proportion of ones in D, the tiles returned by a Boolean factorization are expected to be dense in D. We will discuss in the following section how dense a tile has to be in the data matrix in order to approximate it well. The following theorem explores the probability with which a δ-dense tile of given minimal size exists in a Bernoulli matrix. This gives us an upper bound on the probability $P(\mathbf{Z}_s = 1)$ from Eq. (7.2), which in turn allows us to bound the FDR.

THEOREM 7.4. *Suppose* $\mathbf{B}$ *is an* $m \times n$ *Bernoulli*(p) *matrix,* $\delta \in [0, 1]$, $1 \leq a \leq n$, *and* $1 \leq b \leq m$. *The probability that* $\mathbf{B}$ *has a* δ*-dense tile of size* $|x| \geq a$ *and* $|y| \geq b$ *is no larger than*

$$\binom{n}{a}\binom{m}{b} \exp\left(-2ab(\delta - p)^2\right) \ . \tag{7.3}$$

Proof. If a δ-dense tile $yx^\top$ exists in $\mathbf{B}$, having the size $(|x|, |y|) \geq (a, b)$, then we can construct a δ-dense sub-tile of exact size (a, b). This follows by induction from the observation that removing the sparsest column/row in $\mathbf{B} \circ yx^\top$ from the tile does not decrease the density. Thus, the probability that a δ-dense tile of size at least (a, b) exists is no larger than the probability that a tile of size (a, b) exists.

Now, let $yx^\top$ be such a tile with $|x| = a$ and $|y| = b$. The probability that $yx^\top$ is δ-dense in $\mathbf{B}$ is equal to

$$\begin{aligned} P\left(\frac{y^\top \mathbf{B} x}{|x||y|} \geq \delta\right) &= P\left(\left(\frac{1}{ab}\sum_{i,j} x_i y_j \mathbf{B}_{ji}\right) - p \geq \delta - p\right) \\ &\leq \exp\left(-2ab(\delta - p)^2\right), \end{aligned}$$

where the inequality follows from Hoeffding's inequality. An application of the union bound over all possible combinations to place a ones in x and b ones in y yields the statement of the theorem. □

The proof of Theorem 7.4 indicates that the tightness of the bound in Eq. (7.3) might suffer from the extensive use of the union bound. This originates from the numerous possibilities to select a set of columns and rows of given cardinality. If we expect that rows and columns which are selected by a tile have proportionately many ones in common, we bypass the requirement to take all possible column and row selections into account. To this end, given an $m \times n$ matrix A, we assess the value of the function

$$\eta(A) = \max_{1 \leq i \neq k \leq n} \langle A_{\cdot i}, A_{\cdot k} \rangle \ .$$

THEOREM 7.5. *Let* $\mathbf{B}$ *be an* $m \times n$ *Bernoulli*(p) *matrix and let* $\tau > p^2$*. The function value of* η *satisfies* $\eta\left((1/\sqrt{m})\mathbf{B}\right) \geq \tau$ *with probability no larger than*

$$\frac{n(n-1)}{2} \exp\left(-\frac{3}{2} m \frac{(\tau - p^2)^2}{2p^2 + \tau}\right). \tag{7.4}$$

Proof. Let $\mathbf{B}$ be as described above and $1 \leq i \neq k \leq n$. The variance of the random variable $\mathbf{B}_{ji}\mathbf{B}_{jk}$ is

$$\mathbb{E}\left[\left(\mathbf{B}_{ji}\mathbf{B}_{jk} - p^2\right)^2\right] = p^2(1-p^2)\,.$$

Since the variables $\mathbf{B}_{ji}\mathbf{B}_{jk}$ are independent for $j \in \{1, \ldots, m\}$, the Bernstein inequality yields

$$\begin{aligned} P(\langle \mathbf{B}_{\cdot i}, \mathbf{B}_{\cdot k} \rangle \geq m\tau) &= P\left(\sum_j \left(\mathbf{B}_{ji}\mathbf{B}_{jk} - p^2\right) \geq m\tau - mp^2\right) \\ &\leq \exp\left(-1/2 \frac{(m\tau - mp^2)^2}{mp^2(1-p^2) + 1/3(m\tau - mp^2)}\right) \leq \exp\left(-1/2 \frac{m(\tau - p^2)^2}{2/3p^2 + 1/3\tau}\right) \end{aligned}$$

where we made use of the relations $\mathbb{E}[\mathbf{B}_{ji}\mathbf{B}_{jk}] = p^2$ and $1 - p^2 \leq 1$. The union bound over all possible pairs of distinct rows ($i \neq k$) yields the final result. □

If the columns of a matrix A are normalized, then the function $\eta(A)$ returns the *coherence* of A. The coherence measures how close the column vectors are to an orthogonal system, an extensively studied property in the field of compressed sensing (Foucart and Rauhut, 2013). If all columns of a matrix are orthogonal to each other, then the coherence is zero. The bound in Eq. (7.4) also implies a bound on the coherence of the matrix A. Thus, we refer to Bound (7.4) as the *coherence bound* and to Bound (7.3) as the *density bound*. For any given tile, we can now derive two upper bounds on the quantity $P(\mathbf{Z}_s = 1)$ from Eq. (7.2), and thus control the FDR.

7.1.2 Rejecting the Rejection of Null Hypotheses

How does the density and the coherence bound now help assessing the probability $P(\mathbf{Z}_s = 1)$ from Eq. (7.2)? Let us reconsider the universal formulation of a null hypothesis, which poses that a tile does not reflect actual relations given by the *true* model $Y^* X^{*\top}$. First, we relax this definition by counting those tiles which cover only a fraction of the *true* model among the false discoveries as well. Given factor matrices X and Y and and a fraction parameter $\alpha \in [0, 1]$, we define the

null hypothesis $H_s^0(\alpha)$ to be true if the overlap between the s-th tile and the model is smaller than α

$$\frac{\left|Y_{\cdot s}X_{\cdot s}^\top \circ Y^* \odot X^{*\top}\right|}{\left|Y_{\cdot s}X_{\cdot s}^\top\right|} \leq \alpha. \tag{7.5}$$

Assuming that the positive noise is generated by a Bernoulli (p_+^*) matrix $\mathbf{B}$, we define the random matrix corresponding to the decomposition of the data matrix given in Eq. (7.1) by

$$\mathbf{D} = \left(Y^* \odot X^{*\top} \oplus \mathbf{B}\right) \circ \left(\overline{Y^* \odot X^{*\top}} \oplus \overline{N_-}\right). \tag{7.6}$$

The generalization of the data to a random matrix enables us to asses the probability of a false discovery.

COROLLARY 7.6. *Let* $\mathbf{B}$ *be an* $m \times n$ *Bernoulli* (p_+^*) *matrix and define the random matrix* $\mathbf{D}$ *as in Eq.* (7.6)*. Given matrices* $X \in \{0,1\}^{n\times r}$ *and* $Y \in \{0,1\}^{m\times r}$ *with an observed density of*

$$Y_{\cdot s}^\top D X_{\cdot s} = \delta_s |Y_{\cdot s}||X_{\cdot s}|,$$

then for $\rho = \max\{\delta_s - \alpha - p_+^*, 0\}$*, the probability that the null hypothesis* $H_s^0(\alpha)$ *is true and thus not to be rejected is bounded by*

$$\begin{aligned} P(\mathbf{Z}_s = 1) &= P\left(\frac{\left|Y_{\cdot s}X_{\cdot s}^\top \circ Y^* \odot X^{*\top}\right|}{\left|Y_{\cdot s}X_{\cdot s}^\top\right|} \leq \alpha \wedge \frac{Y_{\cdot s}^\top \mathbf{D} X_{\cdot s}}{|Y_{\cdot s}||X_{\cdot s}|} \geq \delta_s\right) \\ &\leq \binom{n}{|X_{\cdot s}|}\binom{m}{|Y_{\cdot s}|} \exp\left(-2|X_{\cdot s}||Y_{\cdot s}|\rho^2\right) \end{aligned}$$

Proof. We apply the triangle inequality to the density of tile s, substituting $\mathbf{D}$ with its decomposition:

$$\left|Y_{\cdot s}^\top \mathbf{D} X_{\cdot s}\right| \leq \left|Y_{\cdot s}X_{\cdot s}^\top \circ Y^* \odot X^{*\top}\right| + \left|Y_{\cdot s}X_{\cdot s}^\top \circ \mathbf{B}\right|.$$

Dividing by $|Y_{\cdot s}||X_{\cdot s}|$ and applying Eq. (7.5) yields that $Y_{\cdot s}X_{\cdot s}^\top$ is $(\delta_s - \alpha)$-dense in $\mathbf{B}$. The probability for this event is bounded by Theorem 7.4. □

Similar considerations lead to a false discovery bound based on coherence. In this setting, we define the null hypothesis $H_s^0(\beta)$ for $\beta \in \mathbb{N}$ to hold if

$$\eta\left(Y_{\cdot s}X_{\cdot s} \circ Y^* \odot X^{*\top}\right) \leq \beta. \tag{7.7}$$

This restriction affects the tile-wise overlap between the underlying and the computed model more than the definition based on density does. As such, Eq. (7.7) implies that each column of the outer product $Y_{\cdot s}{X_{\cdot s}}^\top$ covers at most β rows of each tile $Y_{\cdot t}^* {X_{\cdot t}^*}^\top$ of the underlying model. The probability of a false discovery according to this definition of a null hypothesis is bounded by the following corollary.

COROLLARY 7.7. *Let* $\mathbf{B}$ *be an* $m \times n$ *Bernoulli*(p_+^*) *matrix and define the random matrix* $\mathbf{D}$ *as in Eq.* (7.6). *Given* $X \in \{0,1\}^{n\times r}$ *and* $Y \in \{0,1\}^{m\times r}$ *having*

$$\eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ D\right) = \tau_s m,$$

then for $\rho = \max\{\tau_s - {}^{\beta}/_{m}, p^2\}$, *the probability that the null hypothesis* $H_s^0(\beta)$ *holds as defined in Eq.* (7.7) *is bounded by*

$$\begin{aligned} P(\mathbf{Z}_s = 1) &= P\left(\eta\left(Y_{\cdot s}X_{\cdot s} \circ Y^* \odot X^{*\top}\right) \leq \beta \wedge \eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ \mathbf{D}\right) = \tau_s m\right) \\ &\leq \frac{n(n-1)}{2} \exp\left(-\frac{3}{2} m \frac{(\rho - p^2)^2}{2p^2 + \rho}\right) \end{aligned}$$

Proof. From the composition of D as denoted in Eq. (7.1) and the definition of η, computing a maximum, follows that

$$\eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ \mathbf{D}\right) \leq \eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ Y_{\cdot s}^* X_{\cdot s}^{*\top}\right) + \eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ \mathbf{B}\right).$$

Applying Eq. (7.7) and $\eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ \mathbf{D}\right) \geq \tau_s m$ yields $\eta(\mathbf{B}) \geq \tau_s m - \beta$. The probability that this inequality holds is bounded by Theorem 7.5. □

We assume from now on that $\alpha = \beta = 0$, by what both definitions of the null hypothesis concur. The following results are though easily adapted to a parametrized definition of the null hypothesis.

7.2 Theoretical Comparison of Proposed Bounds

The bounds from the previous section supposedly enable a theoretically well-founded approach to select the rank for a Boolean factorization. Given any factorization, the proposed bounds help to toss all tiles which may just as well have arisen from noise. However, the tightness of the bounds is the linchpin of the applicability of this scheme. Since we do not require a penalization term of the model complexity to determine the correct rank in the FDR controlled scenario, we can choose the most simple objective function: the residual sum of squares as defined in problem (BMF). The minimization of the residual sum of squares is not only simple to implement, but this function is also simple enough to let us derive characteristics of its optima with regard to coherence and minimum density of tiles. This enables a theoretic characterization of those tiles which would be tossed by the bounds. Moreover, this contributes to a fundamental understanding of the nature of tiles in a minimizing factorization. Assuming the data is composed as stated in Eq. (7.1), we explore the circumstances which have to be met such that a tile in the noise matrix contributes to minimizing the approximation error.

LEMMA 7.8. *Let X and Y be $n\times r$ and $m\times r$ binary matrices and let $s \in \{1,\dots,r\}$. We collect in $\mathcal{T} = \{1,\dots,r\} \setminus \{s\}$ all indices except s and denote with the matrix $M = Y_{\cdot\mathcal{T}} \odot X_{\cdot\mathcal{T}}^\top$ the Boolean matrix factorization excluding the s-th tile. If (X,Y) is a solution of* (BMF)*, then the density of tile $(x,y) = (X_{\cdot s}, Y_{\cdot s})$ is lower-bounded on the area which is not covered by any other tile, i.e.,*

$$\frac{y^\top \left(D \circ \overline{M}\right) x}{y^\top \overline{M} x} \geq \frac{1}{2}. \tag{7.8}$$

Proof. Let X, Y, M and (x,y) be as described above. The Boolean product of X and Y is written in dependence of M as

$$Y \odot X^\top = Y_{\cdot 1}X_{\cdot 1}^\top \oplus \ldots \oplus Y_{\cdot r}X_{\cdot r}^\top = M + yx^\top \circ \overline{M}.$$

The approximation error of a Boolean product is the sum of uncovered ones and covered zeros in the data:

$$\begin{aligned}
\left|D - Y \odot X^\top\right| &= \left\langle D - \mathbf{1} + \mathbf{1} - Y \odot X^\top, D - Y \odot X^\top \right\rangle \\
&= \left\langle D, \overline{Y \odot X^\top} \right\rangle + \left\langle \overline{D}, Y \odot X^\top \right\rangle \\
&= \left\langle D, \overline{M + yx^\top \circ \overline{M}} \right\rangle + \left\langle \overline{D}, M + yx^\top \circ \overline{M} \right\rangle \\
&= \left\langle D, \overline{M} \right\rangle - y^\top (D \circ \overline{M}) x + \left\langle \overline{D}, M \right\rangle + y^\top \left(\overline{D} \circ \overline{M}\right) x \\
&= |D - M| - y^\top \left(D \circ \overline{M}\right) x + y^\top \left(\overline{D} \circ \overline{M}\right) x.
\end{aligned}$$

Since (X,Y) minimizes the RSS, we have $|D - M| \geq \left|D - Y \odot X^\top\right|$. Hence

$$y^\top (D \circ \overline{M}) x - y^\top (\overline{D} \circ \overline{M}) x = 2y^\top (D \circ \overline{M}) x - y^\top \overline{M} x \geq 0.$$

Transforming this inequality yields the final result. □

Note that the proof of Lemma 7.8 implies, that the density in Eq. (7.8) has to be larger than one half, if the objective function incorporates a regularization term on the factor matrices. This could be for instance the $\ell 1$-norm of the matrices. From Lemma 7.8 we now conclude the following property of tiles which reflect a false discovery.

COROLLARY 7.9. *Let the matrices X and Y solve* (BMF)*, and let $s \in \{1,\dots,r\}$. If the tile $Y_{\cdot s}X_{\cdot s}^\top$ is a false discovery and has no overlap with the remaining tiles, i.e., $\langle Y_{\cdot s}, Y_{\cdot t}\rangle\langle X_{\cdot s}, X_{\cdot t}\rangle = 0$ for $s \neq t$, then $Y_{\cdot s}X_{\cdot s}^\top$ is $1/2$-dense in the positive noise matrix N_+.*

A similar procedure leads to a bound on the coherence.

LEMMA 7.10. *Let the matrices X and Y solve* (BMF) *and let $s \in \{1, \dots, r\}$. If the outer product $Y_{\cdot s}X_{\cdot s}{}^\top$ is δ-dense in D, then*

$$\eta(D) > \delta|Y_{\cdot s}|\frac{\delta|X_{\cdot s}| - 1}{|X_{\cdot s}| - 1} \tag{7.9}$$

Proof. Let X, Y and s be described as above. Denote by $\mathcal{I}_s = \{i \in \{1, \dots, n\} \mid X_{is} = 1\}$ the set of all items indicated by $X_{\cdot s}$. Since the $\ell 1$-norm is bounded for a vector x with a nonzero entries by $|x| \leq \sqrt{a}\|x\|$ and since $Y_{\cdot s}X_{\cdot s}^\top$ is δ-dense, it holds that

$$\|\operatorname{diag}(Y_{\cdot s})DX_{\cdot s}\|^2 \geq \frac{\left|Y_{\cdot s}^\top DX_{\cdot s}\right|^2}{|Y_{\cdot s}|} \geq \delta^2|Y_{\cdot s}||X_{\cdot s}|^2.$$

The norm above is equal to

$$\begin{aligned}\|\operatorname{diag}(Y_{\cdot s})DX_{\cdot s}\|^2 &= X_{\cdot s}^\top D^\top \operatorname{diag}(Y_{\cdot s})DX_{\cdot s} \\ &= \sum_{i,k \in \mathcal{I}_s} D_{\cdot i}^\top \operatorname{diag}(Y_{\cdot s})D_{\cdot k} \\ &= Y_{\cdot s}^\top DX_{\cdot s} + \sum_{i \neq k \in \mathcal{I}_s} D_{\cdot i}^\top \operatorname{diag}(Y_{\cdot s})D_{\cdot k}.\end{aligned}$$

Combining both (in)equalities above yields

$$\begin{aligned}\sum_{i \neq k \in \mathcal{I}_s} D_{\cdot i}^\top \operatorname{diag}(Y_{\cdot s})D_{\cdot k} &\geq Y_{\cdot s}^\top DX_{\cdot s}\left(\frac{Y_{\cdot s}^\top DX_{\cdot s}}{|Y_{\cdot s}|} - 1\right) \\ &\geq \delta|Y_{\cdot s}||X_{\cdot s}|\left(\delta|X_{\cdot s}| - 1\right).\end{aligned}$$

According to the pigeonhole principle, indices $i \neq k$ exist, $i, k \in \mathcal{I}_s$ such that

$$\langle D_{\cdot i}, D_{\cdot k}\rangle > \delta|Y_{\cdot s}|\frac{\delta|X_{\cdot s}| - 1}{|X_{\cdot s}| - 1}.$$

□

If we assume that a tile $Y_{\cdot s}X_{\cdot s}^\top$ is a false discovery from a solution (X, Y) to the optimization problem (BMF), then Eq. (7.9) applies to N_+.

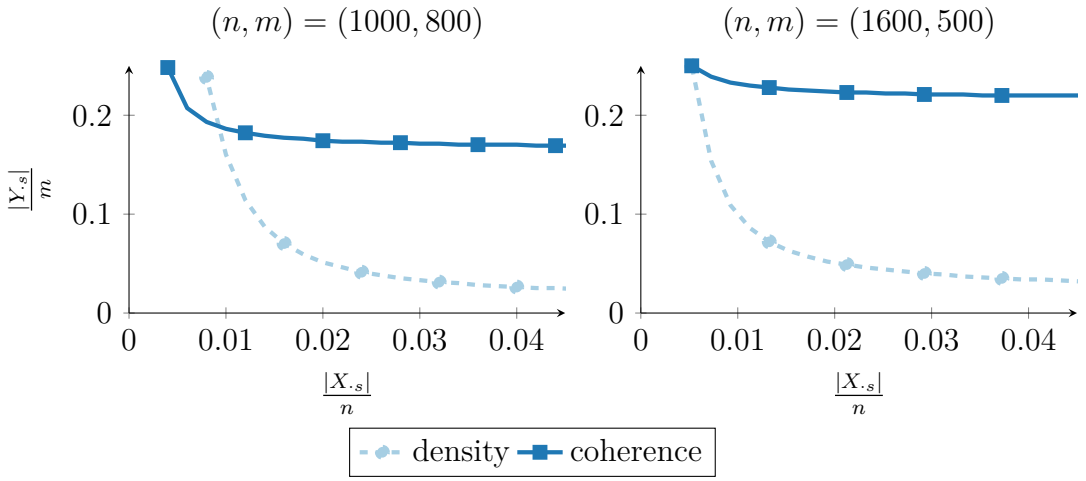

FIGURE 7.1: Minimum relative size $|Y_{\cdot s}|/m$, depending on $|X_{\cdot s}|/n$, for which $P(\mathbf{Z}_s = 1) \leq 0.01$, based on density (blue) and coherence (green).

These results enable a theoretic comparison of the bounds based on coherence and density. Figure 7.1 contrasts the two bounds for two settings of dimensions. The plot on the left refers to almost square dimensions $(n, m) = (1000, 800)$ and the one on the right to more imbalanced dimensions $(n, m) = (1600, 500)$. Let (X, Y) be a solution of (BMF) and assume that the positive noise matrix is a Bernoulli (p_+^*) matrix with probability $p_+^* = 0.1$. We plot the minimum relative size $|Y_{\cdot s}|/m$ against the relative size $|X_{\cdot s}|/n$ such that the probability of a false discovery is $P(\mathbf{Z}_s = 1) \leq 0.01$. The blue curve displays the minimum tile size, assessing the false discovery probability by Corollary 7.6, while green refers to Corollary 7.7. Thereby, we assume that the tile is $1/2$-dense in N_+ and the value $\eta(N_+)$ is bounded by Eq. (7.9). Figure 7.1 indicates that under the given circumstances the coherence provides a more loose bound than the density. The difference between the required sizes is larger if the dimensions are disproportionate. This suggests that more tiles are rejected as potential false discoveries by the coherence bound, in particular for wide or tall data matrices. Most importantly, the density bound is tight enough to let most of the tiles pass, which would be generated according to the synthetic data generation in Algorithm 9. The default setting of the data generating algorithm specifies the relative sizes to lie in the interval $[0.01, \xi_{max} = 0.1]$. The plot shows, e.g., that the density bound would accept tiles having a size of $0.03n \times 0.5m$, even if the negative noise flips half of the ones to zeros. This demonstrates that the density bound is tight enough to align with the common sense regarding the minimum size of identifiable tiles.

ALGORITHM 10 The tossing functions for the application of PAL-TILING implementing the determination of the rank via FDR control.

FUNCTION TOSSDENS(x,y,D)
  $\rho \leftarrow \max\left\{\frac{yDx}{|y||x|} - \alpha - p_+, 0\right\}$
  RETURN $\binom{n}{|x|}\binom{m}{|y|}\exp\left(-2|x||y|\rho^2\right) > q$
END FUNCTION

FUNCTION TOSSCOH(x,y,D)
  $\rho \leftarrow \max\left\{\frac{\eta\left(Y_{\cdot s}X_{\cdot s}^\top \circ D\right) - \beta}{m}, p_+^2\right\}$
  RETURN $\frac{n(n-1)}{2}\exp\left(-\frac{3}{2}m\frac{(\rho - p^2)^2}{2p^2 + \rho}\right) > q$
END FUNCTION

7.3 Algorithmic Integration of FDR Control

The false discovery bounds might be applied as a postprocessing step to any Boolean factorization result. Here, we also establish the use of these bounds to directly estimate the rank. In the rounding procedure at the end of the relaxed optimization in PAL-TILING (Algorithm 7), we can integrate a check of the false discovery bounds via a specification of the function TOSS.

Since we intend to solve problem (BMF), a suitable smooth relaxed objective is the residual sum of squares. Therefore, we employ the vanilla PAL-TILING (Algorithm Specification 1) and the function TOSS as stated in Algorithm 10. We call the resulting algorithm TRUSTPAL, which requires additional to the parameters of PAL-TILING the estimated noise probability p_+, the null hypothesis defining parameters α, β (default value 0) and the FDR control level q (default value 0.01). The functions in Algorithm 10 define that a tile is supposed to be tossed if the risk that it is a false discovery is not bounded above by q. That is, if Corollary 7.6 or 7.7 does not yield $P(\mathbf{Z}_s = 1) \leq q$, then the tile s is removed from the factorization. Multiple combinations of the density and coherence bound are possible. Here, we distinguish between TRUSTPAL employing the density bound, that is function TOSSDENS and TRUSTPAL employing the coherence bound, where we toss tile s when TOSSCOH$(X_{\cdot s}, Y_{\cdot s}, D)$ and TOSSCOH$(Y_{\cdot s}, X_{\cdot s}, D^\top)$ both return true. The application to the transposed factorization serves a symmetric test of the coherence bound.

7.4 Experiments

Our experimental evaluation serves the assessment of provided bounds in practical applications. Although the theoretical properties of minimizing factorizations yield satisfactory bounds on the size of a tile (cf. Figure 7.1), in practice no feasible existing algorithm can guarantee to return optimal solutions of problem (BMF). In addition, we evaluate the sensitivity to the estimated noise probability.

The implementation of TrustPal follows the highly parallel GPU implementation from the framework PAL-Tiling. The source code of TrustPal, together with Julia scripts to generate data and to compare proposed bounds, is provided[1].

We compare the two variants of TrustPal, employing the bounds based on density or coherence to determine the rank, and the performance of the algorithm Primp. For both bounds, we assume that the null hypothesis with regard to a tile holds if the tile covers only noise. That is, the parameters of the rounding procedure in Algorithm 10 are set to $\alpha = \beta = 0$. We have seen in the experimental evaluation of Section 6.3.1 that PanPal displays a strong tendency to underestimate the rank, but is able to yield more accurate results than Primp in some particular settings. Whenever that is the case, we display the results for PanPal in the following plots.

7.4.1 Experiments on Synthetic Data

We generate 1600×500 and 1000×800 data matrices according to Algorithm 9. For every parameter variation, we generate 8 matrices, 4 for each dimension setting $(n, m) \in \{(1000, 800), (1600, 500)\}$. If not stated otherwise, the default settings $p_+^* = p_-^* = 0.1$, $r^* = 25$ and $d = 0.1$ apply. We compare again the computed models against the planted structure by the adaptation of the micro-averaged F-measure, as discussed in Section 6.3.1.

Figure 7.2 displays the performance of the density and coherence approach of TrustPal with Primp. The noise probability is between $p_\pm^* \in \{0, 0.05, \ldots, 0.25\}$. While the plots on the left show aggregated results over 20 almost square 1000×800 matrices, the plots on the right refer to 20 more imbalanced datasets with dimension 1600×500. The input parameter of TrustPal, the estimated noise probability, is consistently set to $p_+ = 10\%$. The plots show that both probability bounds yield similar results. In particular, if the noise percentage exceeds the estimated noise probability, no more than the actually planted tiles are discovered. An overestimation of the rank, as happening with Primp on more square matrices, is prevented.

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

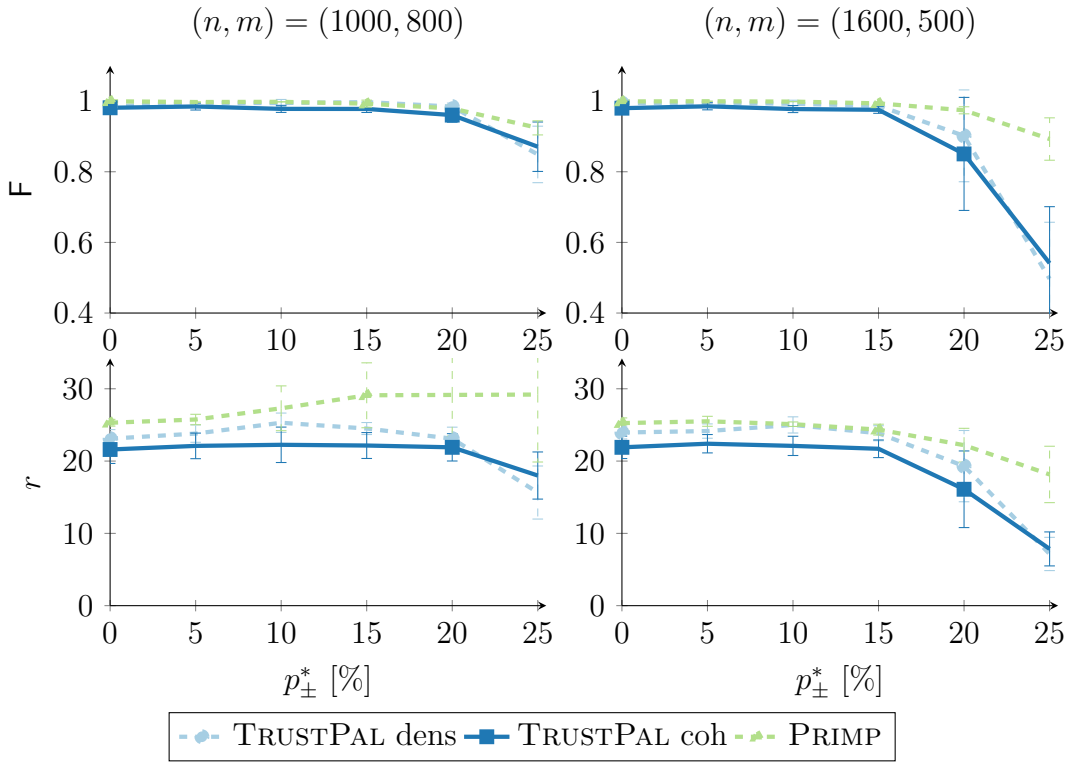

FIGURE 7.2: Variation of uniform noise for 1600 × 500- and 1000 × 800-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise indicated on the x-axis.

We state the average measures over all variations of the positive noise parameter $p^*_+ \in \{5, 10, \ldots, 25\}$ and the rank $r^* \in \{5, 10, \ldots, 45\}$ in Table 7.1. We see that all algorithms consistently gain high F-values and an average deviation of the rank which is close to zero. Yet, Primps average rank deviates to a positive amount and TrustPal rather underestimates the rank. Note, that an overestimation of the rank does not necessarily imply a false discovery; planted tiles might be split.

In Figure 7.3 we plot the F-measure and the computed rank against the parameter ξ_{max}, which bounds the maximum size of a tile. Here, we add a comparison to the algorithm Panpal, whose tendency to underestimate the rank comes in handy for more dense matrices, when ξ_{max} is larger than 0.2. We see that TrustPal is able to find the right balance and obtains the highest F-measure over all variations of ξ_{max}. In total, the experimental evaluation suggests that the estimation of the noise probability is not critical in practice and that both bounds are suitable to yield accurate rank estimations under false discovery control.

Vary	Algorithm	F-measure	$r - r^*$
p_+^*	TrustPal Dens	0.99 ± 0.015	-0.39 ± 1.65
	TrustPal Coh	0.98 ± 0.023	-1.77 ± 1.57
	Primp	0.99 ± 0.004	1.21 ± 2.89
r^*	TrustPal Dens	0.98 ± 0.050	-0.475 ± 2.54
	TrustPal Coh	0.96 ± 0.069	-2.875 ± 3.17
	Primp	0.98 ± 0.074	0.975 ± 2.14

TABLE 7.1: Average F-measure and difference between computed and planted rank $r - r^*$ for varied positive noise and true rank. For each setting the average value is computed over all dimension variations.

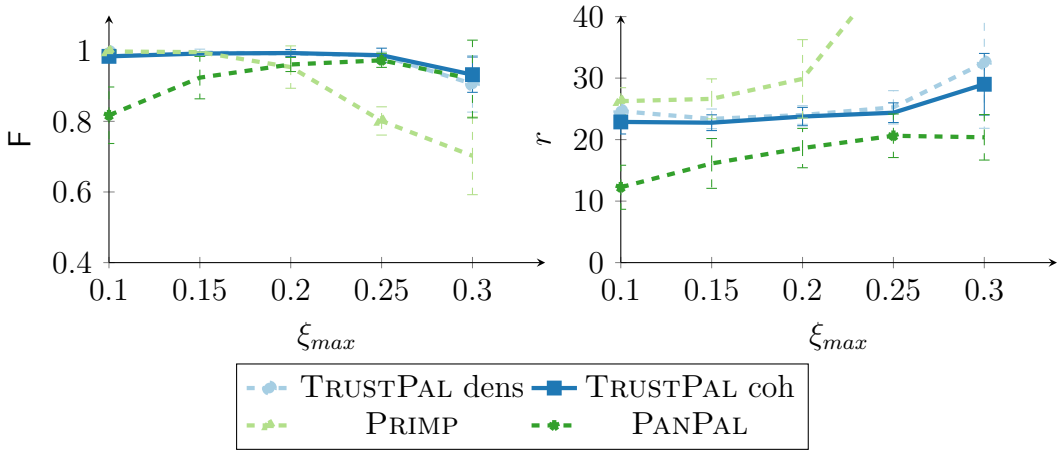

FIGURE 7.3: Variation of density and overlap influencing parameter $\xi_{max} \in [0.1, \ldots, 0.3]$. Comparison of F-measures (the higher the better) and the estimated rank (the closer to 25 the better) for uniform noise of $p_\pm^* = 10\%$.

7.4.2 MovieLens Experiments

Comparing the performance of algorithms in terms of the false discovery rate proves difficult for real-world datasets, since, obviously, the actual noise distribution is unknown. Recommendation data at least provides some information about its positive noise in form of negative reviews. We explore the performance of algorithms on the binarized MovieLens500K dataset, as discussed in Section 6.3.2. We compare the output of TrustPal for various noise probability estimations p_+ to the results from Primp and PanPal.

Table 7.2 summarizes the results. We estimate the positive noise to be small $p_+ \in \{0.01, 0.05, 0.1\}$ as we not often expect that users give a positive rating of a

Algorithm	Bound	p_+	r	$\%RSS$	Wrong rec.
TrustPal	coh	0.1	23	80.68	2.43
		0.05	25	80.52	2.38
		0.01	25	80.26	2.44
	dens	0.1	26	81.59	2.11
		0.05	35	79.54	2.43
		0.01	25	78.22	2.72
Primp	-		78	88.59	2.38
Panpal	-		15	94.05	2.23

TABLE 7.2: Comparison of TrustPal for estimated noise probabilities p_+ with Primp and Panpal on the MovieLens dataset. Denoted are the rank r, approximation error $\%RSS$ and the percentage of traceable wrong recommendations, i.e., user-movie recommendations corresponding to bad reviews (< 2.5 of five stars).

movie they do not actually like. We observe again that a variation of the estimated noise probability does not make much of a difference. The residual sum of squares decreases with decreasing noise estimations, but not more than 4% in total. The calculated rank and the percentage of wrong recommendations do not display such a monotone behaviour. Using the coherence bound, the rank only slightly increases from 23 to 25 when less noise is assumed. The rank determined by the density bound does not display a monotone behaviour. It jumps from 26 to 35 to 25 with decreasing expected noise. The estimated ranks of TrustPal are close to 25, which differs notably from the rank of 78 from Primp. Impressive is that all runs of TrustPal achieve a lower approximation error than Primp an Panpal. Looking at the amount of traceable wrong recommendations, we see that Panpal with its very conservative selection of tiles achieves a generally low amount of wrong recommendations, which is only topped by one run of TrustPal. Primp exhibits a comparably low amount of traceable wrong recommendations as well. Only two of the six runs of TrustPal achieve an equal or lower amount of wrong recommendations than Primp. By and large, the results show that the effect of random initialization is stronger than the the effect of the estimated noise p_+.

7.5 FDR Control in Clustering and Pattern Mining

False discovery control in unsupervised settings is basically unexplored. One notable approach is scan clustering (Pacifico et al., 2007), focusing on one- or two-

dimensional spatial density clustering. The authors control the area of discovered clusters by the FDR, addressing Gaussian processes in continuous data. Thus, this approach cannot be applied to binary or discrete data in general.

In the pattern mining literature, a standard framework for handling false discoveries is the Significant Pattern Mining proposed by Webb (2007). It assesses individual patterns, handling the pattern explosion problem by Bonferroni-like corrections on the significance level. The significant pattern mining framework can work with any null hypothesis to be tested on patterns. One considerable approach that works in this setting is statistical significant pattern mining via permutation testing (see (Llinares-López et al., 2015) and references therein). The major difference to our scenario is the supervision of the mining procedure. In the significant pattern mining scenario, patterns are annotated by class labels, and the task is to identify those patterns which appear significantly more often in one class than in the other class(es). State-of-the-art approaches rely on (variants of) Westfall-Young permutation based hypothesis testing. In a similar line of research, namely statistical emerging pattern mining (Komiyama et al., 2017), patterns from different sources (e.g., databases) are considered. The goal is to find patterns which appear significantly more often in one database than in another. Multiple hypothesis testing is applied to control the FDR, and to provide other statistical guarantees.

A method designed to test one specific null hypothesis on supervised pattern mining results (such as Subgroup Discovery and Exceptional Model Mining) is DFD Validation by Duivesteijn and Knobbe (2011). In what essentially boils down to a permutation test, a Distribution of artificial False Discoveries (DFD) is generated. Subgroups resulting from the actual supervised local pattern mining run are then accepted only if they refute the null hypothesis that they are generated by the DFD. This provides evidence that the subgroups are deemed interesting by more than solely random effects, but the method is specific to the supervised local pattern mining setting. All approaches make heavy use of the fact that data comes from multiple classes or sources and are not easily transferred to the unsupervised setting.

7.6 Discussion

We introduce a method to control the false discovery rate in Boolean matrix factorization and prove two bounds to estimate the probability that a tile minimizes the objective while covering (mostly) noise. A theoretical comparison of our bounds characterizes the tiles which are regarded as false discoveries (cf. Figure 7.1). We explain how FDR control can be integrated into existing BMF algorithms—this

improves the theoretical properties of algorithms and takes away the need to regularize the model complexity. An empirical study on synthetic and real-world data demonstrates its practical utility.

In conclusion, FDR control takes the concern about too noisy results off the researcher's hand. The remaining question is how to derive tiles which approach the underlying model best, e.g., which do not split *true* tiles? In this respect, the suitable application of regularizers is still important. Another arising question is if we can incorporate other noise distributions or how we can test if the noise is, e.g., actually Bernoulli distributed. Multiple avenues of research are opened now.

CHAPTER 8

Mining Class-Specific Alterations with BMF

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 chapter we focus on the characterization of the data with respect to the label in order to find 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 (Schramm et al., 2015). 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 (Vreeken et al., 2007) and Boolean matrix factorization problems (Miettinen, 2012). Both approaches search for factors or patterns which occur in both or only one of the classes. This is illustrated in Figure 8.1; a data matrix is indicated on the left, whose rows are assigned to one class, A or B. While the green outer product spreads over both classes, the blue 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.

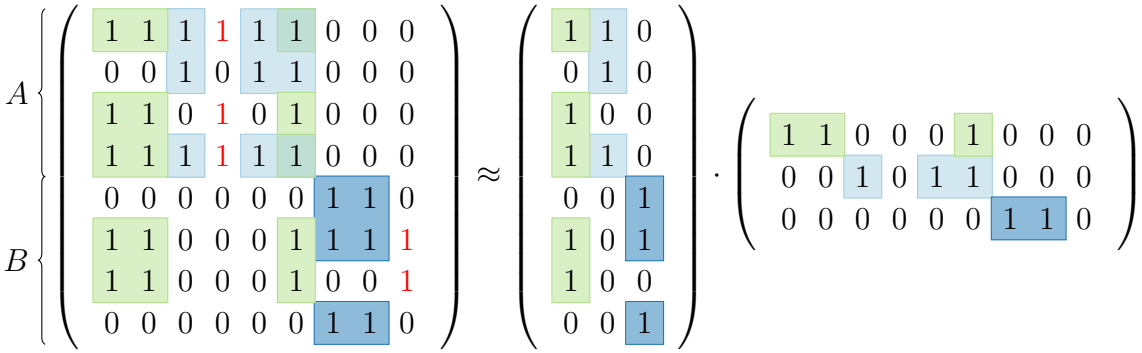

FIGURE 8.1: A Boolean factorization of rank three. The data matrix on the left is composed by transactions belonging to two classes A and B. Each outer product is highlighted. Best viewed in color.

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 Figure 8.1 is not exact, we can see that the red colored ones in the data matrix are not taken into account by the model (the factorization on the right). This is partially desired as the data is expected to contain noise which is supposedly filtered out. On the other hand, we can observe concurrence of the red ones and the pattern of the green tile – in each class. In this chapter we propose a novel Boolean matrix factorization 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:

- the method can be applied to compare any number of classes or sources,
- the factorization rank is automatically determined; this includes the automatic identification of tiles being common among multiple classes as well as the identification of discriminative tiles occurring in only one class,
- in addition to discriminative tiles, more subtle characteristics of classes can be derived, pointing out the features where common tiles deviate among the classes.

While works exist which approach one of the first two points mentioned above (cf. Section 8.1), the focus on subtle deviations among the classes as addressed in the third point is entirely new. This expands the applicability of the new method to datasets where deviations among the classes have a more complex structure.

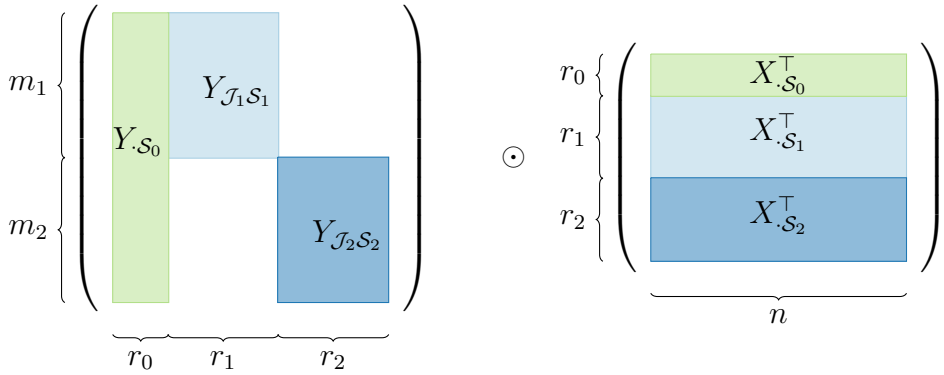

FIGURE 8.2: A Boolean product identifying common (green) and discriminative outer products (blue). Best viewed in color.

8.1 Integrating Class Labels into BMF

We assume that the data matrix is composed of various sources, represented by an assignment of transactions to c classes. We denote the transactions belonging to class a with the set set of indices $\mathcal{J}_a \subset \{1, \ldots, m\}$. The cardinality of set $\mathcal{J}_a$ is m_a, hence we have $m = m_1 + \ldots + m_c$. If the given data matrix is partitioned into classes, a first approach for finding class-defining characteristics is to separately derive factorizations for each class. However, simple approximation measurements such as the residual sum of squares of a Boolean factorization are already nonconvex and have multiple local optima. Due to this local view 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 not trivial. Therefore, finding suitable factorizations which reflect dependencies between the classes is a problem on its own.

In the case of nonnegative, labeled data matrices, measures such as Fisher's linear discriminant criterion are minimized to derive weighted feature vectors, that are patterns in the binary case, which discriminate most between classes. This variant of NMF is successfully implemented for classification problems such as face recognition (Nikitidis et al., 2014) and identification of cancer-associated genes (Odibat and Reddy, 2014).

For social media retrieval, Gupta et al. (2010) introduce Joint Subspace Matrix Factorization (JSMF). Focusing on the two-class setting, they assume that data points 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 Figure 8.2. Note that there are exactly r_0 columns in the factor matrix Y spanning over points from both classes and r_1 respectively r_2 columns whose nonzero entries are restricted to class one or two. Multiplicative

updates are used to minimize a weighted sum of class-wise computed residual sums of squares. In Regularized JSNMF (RJSNMF), a regularization term is used to prevent that shared feature vectors (in $X^\top_{\cdot\mathcal{S}_0}$) swap into discriminative subspaces and vice versa (Gupta et al., 2013). This happens if a column in $Y_{\cdot\mathcal{S}_0}$ is divided into two columns, one in $Y_{\mathcal{J}_1\mathcal{S}_1}$ and the other in $Y_{\mathcal{J}_2\mathcal{S}_2}$. Correspondingly, a discriminative feature vector in $X_{\cdot\mathcal{S}_1}$ or $X_{\cdot\mathcal{S}_2}$ could be represented as one of the shared vectors. 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 numbers of shared and discriminative subspaces, until the approximation error barely decreases. A pioneering extension of this method to the multi-class case ($c \geq 2$) is provided by Gupta et al. (2014).

Miettinen (2012) transfers the objective of JSMF into Boolean algebra, solving

$$\min_{X,Y} \sum_{a\in\{1,2\}} \frac{\mu_a}{2} \left| D_{\mathcal{J}_a\cdot} - Y_{\mathcal{J}_a\mathcal{S}_0\cup\mathcal{S}_a} \odot X^\top_{\cdot\mathcal{S}_0\cup\mathcal{S}_a} \right|$$

for binary matrices D, X and Y, and normalizing constants $\mu^{-1}_{1/2} = \left|D_{\mathcal{J}_{1/2}}\right|$. A variant of the BMF algorithm Asso governs the minimization. A provisional determination of ranks based on the MDL principle is proposed, computing which of the candidate rank constellations yields the lowest description length.

Vreeken et al. (2007) pursue the idea of MDL in the context of deriving 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, as known from pattern mining, restricting the usage to the supporting transactions often results in a vast amount of returned patterns.

In the case where two-classes are given for a nonnegative data matrix, Kim et al. (2015) 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 work, 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.

8.2 Mining Class-Specific Alterations

Given a binary data matrix composed from multiple classes, we assume that the data has an underlying model similar to the one in Figure 8.1. There are common or shared patterns (green) and class-specific patterns (blue). 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 them.

DEFINITION 8.1 (CLASS-SPECIFIC ALTERATIONS). Let $X \in \{0,1\}^{n\times r}$ and $V \in \{0,1\}^{cn\times r}$ be binary matrices, where V stacks the $n \times r$ binary matrices V_a for $1 \leq a \leq c$ on top of each other. 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 $\|V_1 \circ \ldots \circ V_c\| = 0$.

We assume that the 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, the pattern matrix, usage and the noise matrix $N \in \{-1,0,1\}^{m\times n}$, such that for $1 \leq a \leq c$ the data is decomposed as

$$D_{\mathcal{J}_a\cdot} = \theta\left(Y_{\mathcal{J}_a\cdot}(X + V_a)^\top\right) + N_{\mathcal{J}_a\cdot}. \tag{8.1}$$

Given a class-wise decomposed binary data matrix as shown above, we wish to filter the factorization, defined by X, Y and V, from the noise.

8.2.1 C-Salt

We propose to capture class-defining characteristics in the framework of PALTiling, for which 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 penalize nonconformity to this request with the function

$$\begin{aligned} S(Y,V) &= \sum_{s=1}^{r}\sum_{a=1}^{c}\left(|Y_{\mathcal{J}_a s}|\,|V_{a\cdot s}| - {Y_{\mathcal{J}_a s}}^\top D_{\mathcal{J}_a\cdot} V_{a\cdot s}\right) + \sum_{b\neq a} Y_{\mathcal{J}_b s}^\top D_{\mathcal{J}_b\cdot} V_{a\cdot s} \\ &= \sum_{a=1}^{c} \operatorname{tr}\left(\left({Y_{\mathcal{J}_a\cdot}}^\top(\mathbf{1} - 2D_{\mathcal{J}_a\cdot}) + Y^\top D\right) V_a\right). \end{aligned}$$

The left summand in the equation on the top becomes smaller the more often the class-specific alteration occurs in used transactions of that class. Conversely, as the right summand decreases, the class-specific alteration matches less to foreign classes. As a result, the regularizing function $S(Y,V)$ returns zero if every class specific alteration occurs exactly at those transactions where the pattern is used, but only in the corresponding class.

ALGORITHM 11 The PAL-TILING extension C-SALT, mining class-specific alterations in a labeled binary database.

FUNCTION C-SALT$(D; \Delta_r = 10)$
  $(X_K, V_K, Y_K) \leftarrow (\emptyset, \emptyset, \emptyset)$
  FOR $r \in \{\Delta_r, 2\Delta_r, 3\Delta_r, \ldots\}$ **DO**
    $(X_0, V_0, Y_0) \leftarrow$INCREASERANK$(X_K, V_K, Y_K, \Delta_r)$
    FOR $k = 0, 1, \ldots$ **DO** ▷ Select stopping criterion
      $\alpha_k^{-1} \leftarrow M_{\nabla_X F}(V_k, Y_k)$
      $X_{k+1} \leftarrow \text{prox}_{\alpha_k \phi_B}\left(X_k - \alpha_k \nabla_X F(X_k, V_k, Y_k)\right)$
      $\gamma_{ak}^{-1} \leftarrow M_{\nabla_{V_a} F}(X_{k+1}, Y_k)$ ▷ $1 \le a \le c$
      $V_{ak+1} \leftarrow \text{prox}_{\gamma_{ak} \phi_B}\left(V_{ak} - \gamma_{ak} \nabla_{V_a} F(X_{k+1}, V_{ak}, Y_k)\right)$ ▷ $1 \le a \le c$
      $\beta_k^{-1} \leftarrow M_{\nabla_Y F}(X_{k+1}, V_{k+1})$
      $Y_{k+1} \leftarrow \text{prox}_{\beta_k \phi_B}\left(Y_k - \beta_k \nabla_Y F(X_{k+1}, V_{k+1}, Y_k)\right)$
    END FOR
    $(X, V, Y) \leftarrow$ROUND(L, X_k, V_k, Y_k, D)
    IF RANKGAP(X, V, Y, r) **THEN RETURN** (X, V, Y) **END IF**
  END FOR
END FUNCTION

FUNCTION ROUND(L, X_k, Y_k, V_k, D)
  $(X^*, V^*, Y^*, L^*) \leftarrow (0, 0, 0, \infty)$
  FOR $t_x, t_y \in \{0, 0.05, 0.1, \ldots 1\}$ **DO**
    $(X, V, Y) \leftarrow (\theta_{t_x}(X_k), \theta_{t_x}(V_k), \theta_{t_y}(Y_k))$
    $V_a \leftarrow V_a - X \circ V_a$ ▷ $1 \le a \le c$
    $U \leftarrow V_1 \circ \ldots \circ V_c$
    $X \leftarrow \theta(X + U)$
    $V_a \leftarrow V_a - U$ ▷ $1 \le a \le c$
    FOR $s \in \{1, \ldots, r\}, a \in \{1, \ldots, c\}$ **DO**
      IF TOSS$(X_{\cdot s} + V_{a \cdot s}, Y_{\mathcal{J}_a s}, D_{\mathcal{J}_a \cdot})$ **THEN** $(X_{\cdot s}, V_{a \cdot s}, Y_{\mathcal{J}_a s}) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$ **END IF**
    END FOR
    IF $L(X, V, Y) < L^*$ **THEN** $(X^*, V^*, Y^*, L^*) \leftarrow (X, Y, L(X, V, Y))$ **END IF**
  END FOR
  RETURN (X^*, V^*, Y^*)
END FUNCTION

We include this regularizing function in the description length $L(X, Y, V)$ and its relaxation $F(X, Y, V)$. Since the function $S(Y, V)$ is a polynomial and thus semi-algebraic, its composition with definable functions returns a Kurdyka-

Łojasiewicz function, suitable for the optimization via PALM. The overall scheme to derive alternating derivations in a binary data matrix is sketched in Algorithm 11. The algorithm C-Salt largely follows the framework of PAL-Tiling, extending the optimization to more than two matrices which are involved in the factorization. The input of C-Salt is the data matrix D and the rank increment Δ_r, where the transactions belonging to class a are indicated within the chunk $D_{\mathcal{J}_a\cdot}$. For each considered rank, the optimization scheme of PALM is applied to the nonbinary penalization of the relaxed objective (line 5-12). The alternating updates with respect to more than two matrices corresponds to the extension of PALM for multiple blocks, as discussed by Bolte et al. (2014). Subsequently, a rounding procedure is applied. Within the rounding procedure, the validity of Definition 8.1 to the class-specific alteration matrix V is ensured (line 20-24). Once more, we apply a function Toss to remove trivial outer products. The number of remaining outer products defines the rank. If the gap between the number of possibly and actually modeled tiles is larger than one, the current factorization is returned (line 14).

ALGORITHM SPECIFICATION 4 (C-SALT). Apply PAL-Tiling with the function Toss from Algorithm 8 and the following functions, where the function L_{CT} and G are the same as denoted for Primp (Algorithm Specification 3). We employ the following constants, objective L and its smooth part F:

$$c_i = -\log\left(\frac{|D_{\cdot i}|}{|D|}\right),\ \mu = 1+\log(n)$$
$$L(X,V,Y) = L_{\mathsf{CT}}\left(\left[X\ V_1 \dots V_c\right], Y\right) + S(Y,V)$$
$$F(X,V,Y) = \frac{\mu}{2}\sum_{a=1}^{c}\left\|D_{\mathcal{J}_a\cdot} - Y_{\mathcal{J}_a\cdot}\left(X+V_a\right)^\top\right\|^2 + \frac{1}{2}G\left(\left[X\ V_1 \dots V_c\right], Y\right) + S(Y,V).$$

The partial gradients are given as follows:

$$\nabla_X F(X,V,Y) = \mu\sum_{a=1}^{c}\left(Y_{\mathcal{J}_a\cdot}(X+V_a)^\top - D_{\mathcal{J}_a\cdot}\right)^\top Y_{\mathcal{J}_a\cdot} + \frac{\mathbf{c}\mathbf{1}^\top}{2}$$
$$\nabla_{V_a} F(X,V,Y) = \mu\left(Y_{\mathcal{J}_a\cdot}(X+V_a)^\top - D_{\mathcal{J}_a\cdot}\right)^\top Y_{\mathcal{J}_a\cdot} + \frac{\mathbf{c}\mathbf{1}^\top}{2} + D^\top Y + \left(\mathbf{1} - 2D_{\mathcal{J}_a\cdot}\right)^\top Y_{\mathcal{J}_a\cdot}$$
$$\left(\nabla_Y F(X,V,Y)\right)_{\mathcal{J}_a\cdot} = \mu\left(Y_{\mathcal{J}_a\cdot}(X+V_a)^\top - D_{\mathcal{J}_a\cdot}\right)^\top X - \frac{1}{2}\left(\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) - 1\right)_{j\in\mathcal{J}_a, s} + \sum_{b=1}^{c} D_{\mathcal{J}_a\cdot}V_b + \left(\mathbf{1} - 2D_{\mathcal{J}_a\cdot}\right)V_a.$$

Further, we specify the employed Lipschitz moduli:

$$M_{\nabla_X F}(Y,V) > \mu\left\|YY^\top\right\|, \qquad M_{\nabla_{V_a} F}(X,Y) > \mu\left\|Y_{\mathcal{J}_a \cdot} Y_{\mathcal{J}_a \cdot}{}^\top\right\|$$
$$M_{\nabla_Y F}(X,V) > \|(M_{Ya}(X,V))_a\|, \qquad M_{Ya}(X,V) = \mu\left\|(X+V_a)(X+V_a)^\top\right\| + m_a.$$

We follow the automatic rank determination scheme by minimizing the description length of the encoding by a code table, as pursued by Primp. The class-specific alterations are denoted by means of standard codes, just as patterns are stated in the code table. Therewith, we obtain an extension of this description length to class-specific alterations as denoted in Algorithm Specification 4. Correspondingly, we employ the simplistic tossing function from Algorithm 8. An automatic rank determination by FDR control is also applicable in this setting. We check for the FDR probability bounds whenever the application domain is sensitive to false discoveries.

8.3 Experiments

The experimental evaluations concern the following research questions: First, given that the data matrix is generated as stated in Eq. (8.1), does C-Salt find the original data structure? Second, is the assumption that real-world data emerge as stated in Eq. (8.1) 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 Miettinen (2012), and Primp. The first question is approached by a series of synthetic datasets, generated according to Eq. (8.1). To address the second question, we compare on real-world datasets the residual sum of squares, computed factorization ranks and visually inspect derived patterns. Furthermore, we discuss an application in genome analysis where none of the existing methods is able to provide the crucial information.

For C-Salt and Primp we use as stopping criterion a minimum average function decrease (of last 500 iterations) of 0.005 and maximal $10,000$ 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 $\{2,\ldots,8\}$.

C-Salt is implemented for GPU. We provide the source code of our algorithms together with the data generating script [1].

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

8.3.1 Experiments on Synthetic Data

EVALUATION For synthetic datasets, we compare the computed models against the planted structure by the known adaptation of the micro-averaged F-measure. However, here we compute the F-measure for every class with respect to the matrices $Y_{\mathcal{J}_a \cdot}$ and $X_a = X + V_a$. We denote the class-wise calculated F-measure with F_a.

Since class-specific alterations of patterns, reflected by the matrix V, are particularly interesting in the scope of this work, we additionally state the recall of V^*, denoted by rec_V. Therefore, we compute another maximum matching σ_V between generated class alterations V^* with usage Y^* and computed patterns $X_V = [X\ V_1\ \ldots V_c]$ (setting V to the $cn \times r$ zero matrix for other algorithms than C-SALT) with usage $Y_V = [Y \ldots Y]$ (concatenating c times). The recall rec_{V_a} is then computed with respect to the matrices V^*, Y^*, 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

$$\mathsf{F} = \frac{1}{c}\sum_a \mathsf{F}_a, \quad \mathrm{rec}_V = \frac{1}{c}\sum_a \mathrm{rec}_{V_a} \text{ and } r = \frac{1}{c}\sum_a r_a.$$

Therewith, the size of the class is not taken into account; the discovery of planted structure is considered equally important for every class.

DATA GENERATION Algorithm 12 states the synthetic data generation as a procedure which receives the class labels in form of indices $(\mathcal{J}_a)_a$ and n, the factorization rank r^*, matrix $C \in \{0,1\}^{c \times v}$ and noise probability p^* as input. The matrix C indicates which patterns are common over all classes and which occur in only some classes. We define for our experiments three instances, one for each considered number of classes $c \in \{2,3,4\}$

$$C_2 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \quad C_3 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \quad C_4 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Every zero in the matrix C determines an $m_a \times \frac{r^*}{v}$ submatrix of Y which is entirely set to zero. That is, the entry C_{au} indicates whether the patterns $X_{\cdot s}$ with $(u-1)\frac{r^*}{u} < s \leq u\frac{r^*}{v}$ are used at all in class a. For example, the shape of Y depicted in Figure 8.2 for $r_0 = r_1 = r_2 = \frac{r^*}{v}$ would be generated when using C_2. By default, we set the parameters $r^* = 24$, $m_a = \frac{m}{c}$ and the noise flipping probability $p^* = 0.1$.

ALGORITHM 12 Generation of synthetic datasets for Boolean matrix factorizations with class-specific alterations.

FUNCTION GENERATENOISYCSBMF$(n, (\mathcal{J}_a)_a, r^*, C, p^*)$ ▷ $C \in \{0,1\}^{c \times v}$
  $X^* \leftarrow$ UNIFORMRAND$(\left\{ X \in \{0,1\}^{n \times r} \;\middle|\; 0.01 \leq \frac{|X_{\cdot s}|}{n} \leq 0.1 \right\})$
  $V^* \leftarrow$ UNIFORMRAND$(\left\{ V \in \{0,1\}^{cn \times r} \;\middle|\; 0.01 \leq \frac{|V_{a \cdot s}|}{n}, \frac{|V_{\cdot s}|}{|X^*_{\cdot s}|} \leq \frac{2}{3} \right\})$
  $Y^* \leftarrow$ UNIFORMRAND$(\left\{ Y \in \{0,1\}^{m \times r^*} \;\middle|\; 0.01 \leq \frac{|Y_{\mathcal{J}_a s}|}{m_a} \leq 0.1, 1 \leq a \leq c \right\})$
  FOR $(a, s) \in \{1, \ldots, c\} \times \{1, \ldots, r^*\}$ **DO**
    ${Y_{\mathcal{J}_a s}}^* \leftarrow C_{au} {Y_{\mathcal{J}_a s}}^*$ where $u = \left\lceil s \frac{v}{r^*} \right\rceil$
  END FOR
  $D_{\mathcal{J}_a \cdot} \leftarrow {Y_{\mathcal{J}_a \cdot}}^* \odot ({X^*} + {V_a}^*)^\top$ ▷ Ensure Def. 8.1
  FOR $1 \leq j \leq m, 1 \leq i \leq n$ **DO**
    $D_{ji} \leftarrow 1 - D_{ji}$ with probability p^*
  END FOR
  RETURN (X^*, V^*, Y^*)
END FUNCTION

SENSITIVITY TO NOISE AND CLASS DISTRIBUTIONS We plot for the following series of experiments the averaged F-measure, recall rec_V, and the rank against the varied data generation parameter. For every experiment, we generate eight matrices: two for each combination of dimensions $(n, m) \in \{(500, 1600), (1600, 500), (800, 1000), (1000, 800)\}$.

Figure 8.3 contrasts the results of C-SALT, PRIMP and DBSSL in the two-class setting. For DBSSL, we evaluate two parameter settings. We recall that DBSSL requires a specification of the number of shared and discriminative patterns. We generate the usage matrix according to matrix C_2, where we have $v = 3$ types of patterns: $\frac{r^*}{3}$ shared patterns and just as many discriminative patterns for each class. However, the question is to which of the sets (shared or discriminative) we assign class-specific alterations. In one perspective, class-specific alterations state discriminative patterns on their own. In this view, we have $\frac{r^*}{3}$ shared patterns and $2\frac{r^*}{3}$ discriminative patterns for each class. Another perspective is to interpret a shared pattern together with its class-specific alterations as one discriminative pattern. In this view, we have zero shared patterns and $2\frac{r^*}{3}$ discriminative patterns for each class. Both settings correctly reflect the number of planted discriminative and shared patterns. In the experiments varying the rank, we employ the MDL-based selection of the rank proposed for DBSSL.

Figure 8.3 shows the performance measures of the competing algorithms when varying three parameters: noise p^*, ratio of transactions per class $\frac{m_1}{m}$ and rank r^*. We observe an overall high F-measure of C-SALT and PRIMP. Both DBSSL

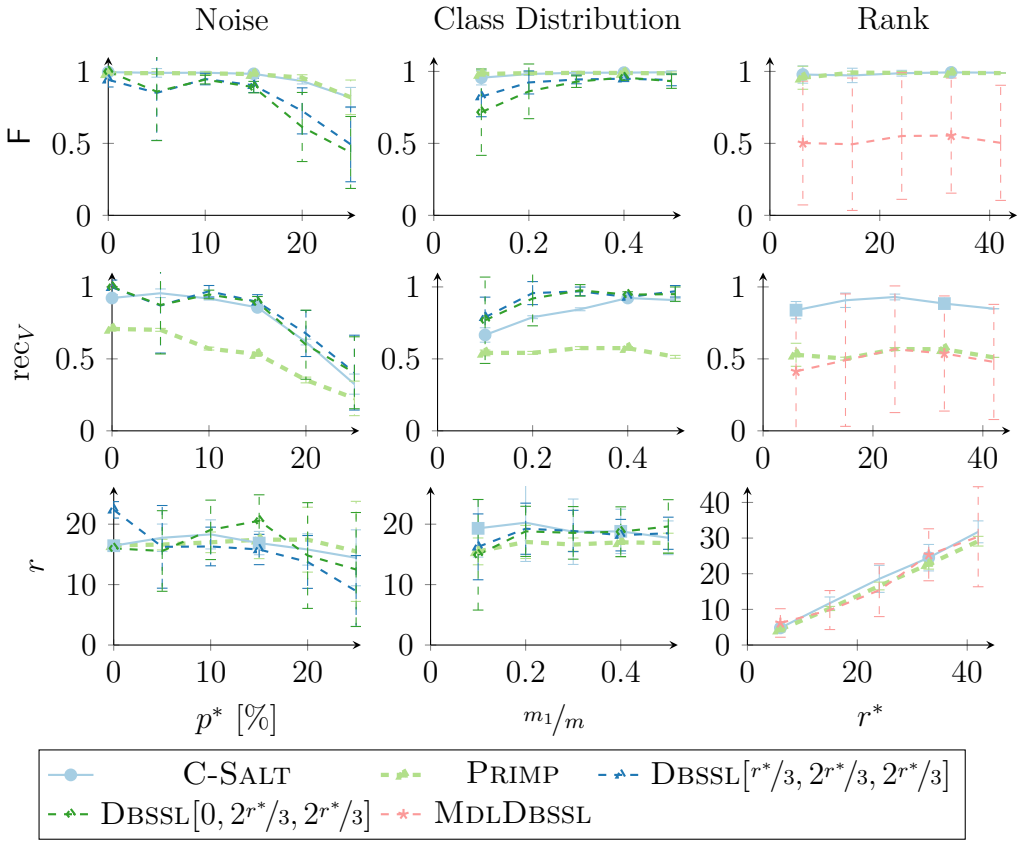

FIGURE 8.3: Variation of noise, class distribution $\frac{m_1}{m}$ and the rank. The F-measure, recall of the matrix V (both the higher the better) and the class-wise estimated rank of the calculated factorization is plotted against the varied parameter. Best viewed in color.

instantiations also obtain 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^* decrease to 0.5 if the rank is not set to the correct parameters for Dbssl.

Figure 8.4 displays the results of Primp and C-Salt when varying the noise for generated factorizations according to three (using matrix C_3) and four classes

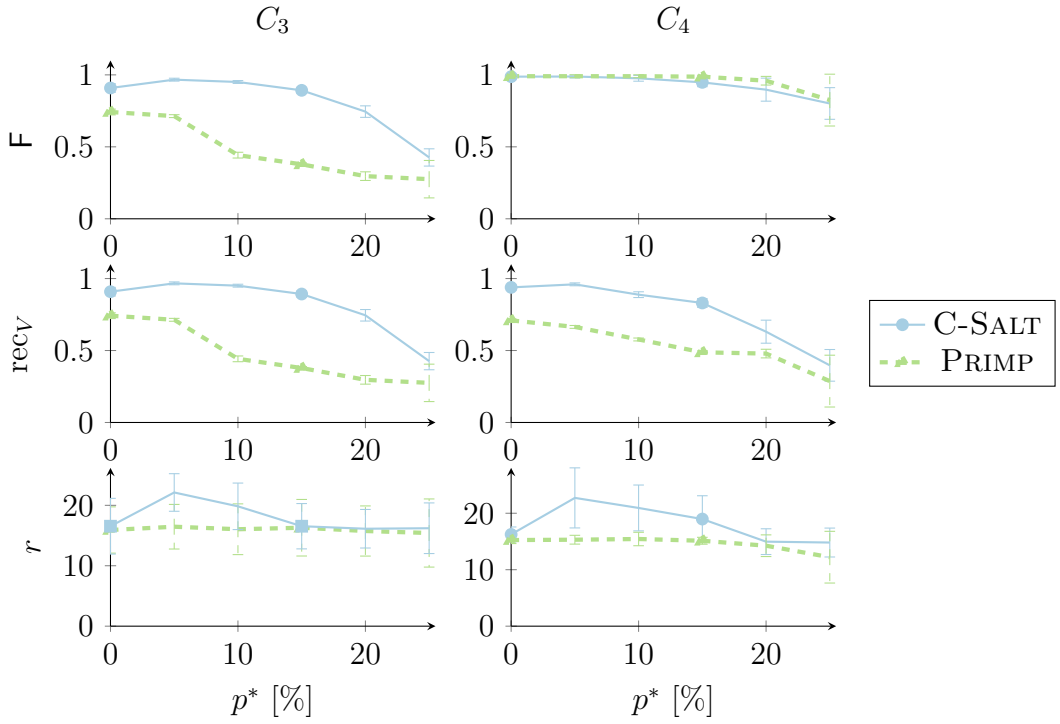

FIGURE 8.4: Variation of noise for generated data matrices with three (left) and four classes (right). The F-measure, recall of the matrix V (both the higher the better) and the class-wise estimated rank of the calculated factorization (between 16 and 24 can be considered correct) is plotted against the varied parameter. Best viewed in color.

(using matrix C_4). The plots are similar to Figure 8.3. The more complex constellations of class-overarching tiles, 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.

8.3.2 Real-World Data Experiments

We explore the algorithms' behavior by three interpretable text datasets depicted in Table 8.1. The datasets are composed by two classes, allowing 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 Newsgroup corpus[2], we compose the *Space-Rel* dataset by posts from `sci.space` and

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

Dataset	m_1	m_2	n	Density [%]
Space-Rel	622	980	2244	2.27
Politics	936	775	2985	2.64
Movies	998	997	4442	3.68

TABLE 8.1: Characteristics of considered datasets: Number of rows belonging to class one m_1 and two m_2, number of columns n and density $|D|/(nm)$ in percent.

Data	Algorithm	r	r_0	r_1	r_2	$\%RSS$
Space-Rel	C-SALT	29(28)	4(3)	9(9)	16(16)	93.1(94.3)
	PRIMP	30(30)	8(8)	8(8)	14(14)	93.1(**93.1**)
	DBSSL1	40(7)	19(1)	13(4)	8(2)	**89.5**(96.8)
	DBSSL2	18(6)	7(1)	8(4)	3(1)	93.1(96.8)
Politics	C-SALT	41(40)	10(10)	19(18)	12(12)	88.2(**88.3**)
	PRIMP	30(30)	8(8)	15(15)	7(7)	90.5(90.5)
	DBSSL1	57(20)	16(2)	27(14)	14(4)	**81.6**(90.5)
	DBSSL2	42(15)	5(0)	18(11)	19(4)	86.0(91.2)
Movies	C-SALT	26(25)	25(25)	1(0)	0(0)	98.1(98.2)
	PRIMP	30(27)	29(27)	1(0)	0(0)	97.8(98.0)
	DBSSL1	27(4)	21(1)	3(1)	3(2)	**96.6**(**97.5**)
	DBSSL2	12(4)	6(0)	3(1)	3(3)	96.9(97.6)

TABLE 8.2: Comparison of the amount of derived discriminative (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 for real-world datasets. Values in parentheses correspond to factorizations where outer products with less than four items or transactions are discarded.

`talk.religion.misc`, and the *Politics* dataset from `talk.politics.mideast` and `talk.politics.misc`. The *Movies* dataset is prepared from a collection of 1000 negative and 1000 positive movie reviews[3].

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 $V_{a\cdot s}+X_{\cdot s}$ as individual class-specific patterns if $|V_{a\cdot s}| > |X_{\cdot s}|$.

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

FIGURE 8.5: Illustration of a selection of derived topics for the 20 News articles from classes *space* and *religion*. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^{\top} D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a space and green a religion article. Best viewed in color.

Table 8.2 shows the number of discriminative and common patterns, and the resulting residual sum of squares. 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*, discarding outer products reflecting less than four items or transactions (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 Newsgroup 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 Movies dataset only by class-common patterns. We inspect these results more closely in the next paragraph, showing that mining class-specific alterations points at exclusively derived class characteristics, especially for the Movies dataset.

ILLUSTRATION OF FACTORIZATIONS Let us inspect the derived most prevalent topics in the form of word clouds. Figures 8.5, 8.6 and 8.7 display for every algorithm the top four topics, whose outer product (tile) 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-Newsgroup data are mostly attributed to one of the classes. The topics are generally interpretable and even comparable among the algorithms (cf. the first

FIGURE 8.6: Illustration of a selection of derived topics for the 20 News articles from classes *mideast politics* and *miscellaneous politics*. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^{\top} D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a mideast and green a miscellaneous politics article. Best viewed in color.

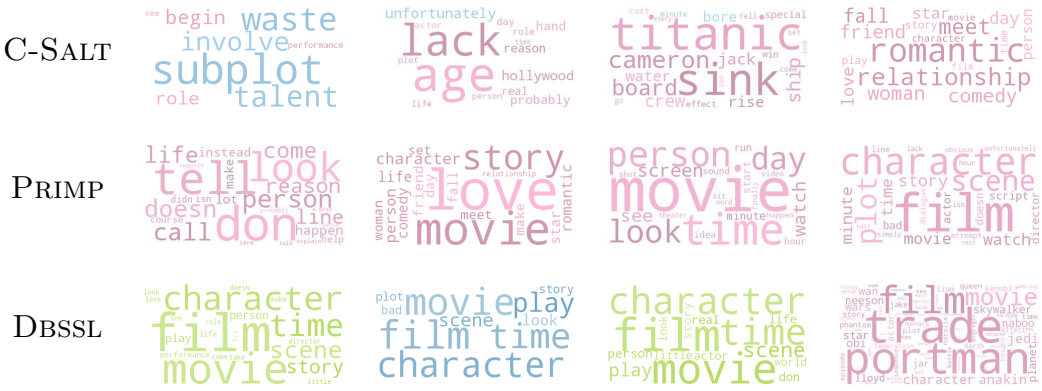

FIGURE 8.7: Illustration of a selection of derived topics for the Movies dataset, comprising *bad* and *good* reviews. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^{\top} D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a bad and green a good review. Best viewed in color.

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 Movies 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 gives the decisive hint which additional words indicate the class membership. We recall from Table 8.2 that DBSSL returns

FIGURE 8.8: Transposed usage matrix returned by C-SALT on the genome dataset. Class-memberships are signalized by colors.

TABLE 8.3: 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

in total four truncated topics for the Movies dataset. Thus, the displayed topics for the Movies 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 common patterns denote for some datasets the sole class-distinctions.

8.3.3 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 Schramm et al. (2015), 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 $A \to C$, insertions ($C \to AC$) and deletions ($AC \to A$). 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 32, of which we omit 20 patterns by means of the density bound (Theorem 7.4) with a noise assumption of 20%. Figure 8.8 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 8.3 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 Schramm et al. (2015).

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 the class. Running Primp separately on each class-related part $D_{\mathcal{J}_a\cdot}$ yields factorizations of rank zero – the genomic alignments between patients can not be differentiated from noise for such few samples. However, applying Vanilla PAL-Tiling separately, minimizing the RSS without any regularization for a specified rank of 15, yields patterns for each part $D_{\mathcal{J}_a\cdot}$. 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. This illustrates the importance of a global view on shared, discriminative and alterating patterns.

8.4 Discussion

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 by means of numerical optimization, extending the method Primp to incorporate class assignments of the transactions. We integrate a factor matrix reflecting class-specific alterations of outer products from a BMF (cf. Definition 8.1). 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. Figure 8.3). Even in the case of more than two classes, C-Salt filters complex dependencies among patterns and the classes (cf. Figure 8.4). These experiments also show that the rank is suitably estimated. On interpretable text data, C-Salt derives meaningful factorizations which provide valuable insight into prevalent topics and their class specific characteristics (cf. Table 8.2 and Figures 8.5, 8.6 and 8.7). An analysis of genomic data underlines the usefulness of our new factorization method, yielding information which none if the existing algorithms can provide (cf. Section 8.3.3).

CHAPTER 9

Conclusions

Making decisions is hard – however, with respect to machine learning, we have seen that there are some tools to reach suitable binary solutions by means of theoretically sound and empirically robust methods. With this work, we contribute to the theory of making hard decisions in the scope of clustering. Our overview of state-of-the-art clustering methods shows that there is a variety of clustering tasks equivalent to matrix factorization problems involving binary constraints on one or two of the factor matrices. Summarizing clustering problems in the framework of matrix factorization has proven useful for deriving characteristics of solutions and comparing approaches. For instance, we lay out in which sense the famous k-means clustering task corresponds to learning orthogonal projections, eigenvectors, and nonnegative factorizations in Theorem 2.1.

The overview of clustering tasks as matrix factorization has additionally proven useful to provide deeper insights into the state of the art. As such, we lay out how the seemingly arbitrary step of discretization via k-means in spectral clustering can actually be made theoretically justified. Here, the formalization of the minimum cut objective as matrix factorization displays its relation to kernel k-means, yielding the equivalence of minimum cut and k-means clustering, given a symmetric decomposition of the data matrix such as the eigendecomposition. Based on this observation, we have developed the algorithm SpectACl, tackling two of the main issues in spectral clustering: robustness to noise and interpretability of the embedding. Looking at visualizations of the embedding used in SpectACl (cf. Figure 4.2), we see how every projected eigenvector corresponds to one cluster component of a specific density. The exploration of density-based clustering in the framework of spectral clustering furthermore builds a bridge to the other main approach in nonconvex clustering, which is represented by the DBSCAN algorithm. In conclusion, the algorithm SpectACl and its normalized version is an efficient, robust, and versatile tool to derive nonconvex clusters.

Our comparison of popular clustering objectives shows that the majority of methods is designed to find partitioning clusters. Suitable adaptations of Lloyd's algorithm (cf. Section 3.2) guarantee the convergence to a local optimum of the objective function subject to the partitioning and particularly binary constraints. This offers an undeniable advantage over other practical alternatives, requiring a relaxation of the binary constraint. The major drawback of relaxing approaches is the discretization step, in which theoretical guarantees, which might be provided for solutions of the relaxed objective, are usually lost. However, the partitioning constraint, enabling the alternating minimization according to Lloyd, is not feasible in some applications. Overlapping and nonexhaustive clusterings are more likely to represent the *true* model when it comes among other things to the clustering of text or genomic data. In this case, the theoretical foundation regarding the efficient optimization of corresponding objectives is leaky.

In this work, we mainly focus on the optimization of Boolean matrix factorizations, which is explicitly designed to model overlapping clusters of binary data. The optimization of this clustering objective is provisional, since so far only greedy heuristics prevail. This results in a strong preference bias towards factorizations where the first tile covers the largest chunk of the data, and consecutive tiles cover less and less. By contrast, we propose the simultaneous optimization of all clusters via an efficient proximal gradient procedure, based on a nonnegative relaxation of the binary values. We specifically incorporate a penalty term to regulate the convergence towards approximately binary values. This way, we still can not guarantee that the discretized solutions are actual local minima of the objective function, but we can guarantee that the approximately binary result is a local minimum of the relaxed objective.

The approach to add a penalty term for nonbinary values is not new (cf. Section 3.5). In particular, the optimization scheme proposed by Zhang et al. (2007) incorporates nonbinary penalization by means of the Mexican hat function in the scope of binary matrix factorization, aiming once again for nonoverlapping clusters. However, the proposed optimization scheme is based on multiplicative updates, which are slow in practice due to a very conservative choice of the stepsize. We propose the optimization of Boolean matrix factorization via PALM, a universal approach which is applicable to solving constrained matrix factorizations. The constraints are thereby required to be convertible to a simple but possibly nonsmooth regularization term whose proximal mapping is efficient to compute. We have derived a closed form for the proximal mapping of a nonbinary penalization term (cf. Section 5.3.1). Hence, we contribute to the theory of proximal optimization, extending its applications to problems involving binary or more general integer constraints.

The task of Boolean matrix factorization is moreover interesting due to its established methodology to estimate the rank of the factorization. In particular, the Minimum Description Length (MDL) principle is used for that purpose. Originating from the selection of interesting patterns in pattern mining, the MDL principle has been transferred to the more general notion of Boolean matrix factorization. In this work, we explore two directions in order to automatically select the rank of Boolean factorizations: the one is to minimize the description length via a suitable relaxation as a smooth Kurdyka-Łojasiewicz function, and the other is based on controlling the false discovery rate. We propose two instances of the first direction with the algorithms Primp and Panpal. Our experimental evaluation shows that the method Primp, employing a description length based on a compression by code tables, delivers correct rank estimations and is able to filter the haphazard from the formative structure. However, there are some settings where Primp tends to overestimate the number of clusters. In this case, the result is likely to contain false discoveries, that are tiles which mostly cover noise. For some applications, controlling the amount of the false discoveries is crucial. For such cases, we establish the estimation of the probability that particular tiles are generated by Bernoulli distributed noise (cf. Section 7.1.2). Our method TrustPal incorporates the novel model selection by the false discovery rate into our optimization framework PAL-Tiling. Here, we furthermore contribute towards the aspect of explainability. In a theoretical analysis of our false discovery controlling mechanism, we derive characteristics of local minima of the Boolean factorization's residual sum of squares. As a result, we have found out that every item selected in a Boolean bicluster occurs in at least half of the data points assigned to that cluster; hence, giving first explanations of what makes a tile a tile.

Last but not least, we have explored the possibility to derive descriptions of clusters with regard to predefined classes. Identifying similarly expressed features together with class-discriminating features in one cluster helps the practitioner to explore a dataset in a supervised setting. A motivating example is the analysis of genomic mutations, and the development of genomic variations from normal to tumor cells and a possible relapse. The method C-Salt extends the tradition model of matrix factorization with an additional matrix, reflecting class-specific feature expressions for each cluster. Therewith, we find patterns in a database which cannot be derived with state-of-the-art methods. Due to the near-orthogonality of matrix factorizations (cf. Section 2.2.1), deviations from the patterns which identify cluster membership are hardly recovered by general matrix factorization approaches. The model assumption of C-Salt is a fundamental change in the core of the matrix factorization scheme. It takes into account that class-defining characteristics may lie in the alteration of a theme and do not pose a theme by itself. Hence, groups are identified together with their class-indicating features.

With regard to our motivating example, the model assumption of C-Salt takes into account that there may not be one therapy for all, but multiple treatments, each one tailored to features of people belonging to one cluster.

9.1 Impact and Future Directions

We have discussed and evaluated in which ways deriving Boolean matrix factorizations by advanced numerical optimization methods is preferable over greedy optimization. This poses only the beginning of a potentially versatile theory of numerical optimization for binary learning tasks. The theoretical progress in the field of nonconvex optimization provides new insights, which are now transferable to Boolean factorization (and perhaps beyond Boolean). Meanwhile, stochastic (Davis et al., 2016), accelerated (Li and Lin, 2015), and inertial (Pock and Sabach, 2016) variants exist of the PALM optimization scheme. The exploration of stochastic gradient descent for matrix factorization is particularly promising with regard to its generalizing properties (Hardt et al., 2016; Hoffer et al., 2017).

Inertial methods are interesting since they can find *good* local optima, even if the objective functions' landscape is rough (Goh, 2017). Inertial methods integrate the direction of the previous gradient descent update into the current step, which is sometimes referred to as giving gradient descent a short-time memory. This is also the foundation of accelerating methods, which can improve the convergence rate. Considering that the framework PAL-Tiling is basically applicable to all discussed matrix factorizations from Chapter 2 (discarding the partitioning requirement), PAL-Tiling and possible extensions potentially provide a general optimization scheme for hard clusterings.

If the matrix product is not approximated in another algebra (as in Boolean matrix factorization), then we can adapt the optimization scheme of PAL-Tiling such that the thresholding step at the end is no longer necessary. For example, consider the optimization of the objective of k-means by PAL-Tiling. One could derive binary solutions via PAL-Tiling, by increasing the weight of the nonbinary penalization such that only binary values in the cluster assignment matrix are possible. Although this optimization scheme comes with theoretical convergence guarantees to a local optimum, the result will probably not be suitable due to the vast amount of local minima in k-means clustering. On the contrary, the method is likely to be stuck at a local optimum close to the initialization matrix. This issue can be addressed by providing better initializations. Those initializations may themselves be derived by the application of PAL-Tiling, employing a lower scale nonbinary penalty term. Hence, we could also gradually increase the nonbi-

nary penalty term, starting with an unconstrained matrix factorization. This is a promising approach to derive matrix factorizations with binary constraints under convergence guarantees.

With regard to our evaluations of automatic determinations of the factorization rank, extending clustering from binary to real-valued data would be of interest. In particular, the experiments of SpectACl display potential to integrate model selection techniques into this nonconvex clustering task. Looking at the visualization of the clustering of the three blobs dataset in Figure 4.4, we see that SpectACl is able to identify a smaller and dense cluster on top of a larger cluster. Imagine that we compare SpectACl's clustering of two clusters with the one shown, assuming three clusters. It is reasonable to assume that SpectACl returns the two point clouds at both edges just as DBSCAN if the number of clusters is set to two. If we follow the approach of hypothesis testing from TrustPal, then a bound on the probability that the smaller, denser cluster appears due to noise effects from a larger cluster would deliver a theoretically founded way of determining whether the rank of this three blobs dataset is equal to two or three.

APPENDIX A

Proofs of Chapter 2

We state here the more lengthy proofs of results presented in Chapter 2.

THEOREM 2.1. *The following optimization problems are equivalent to objective* (KM)

$$\min_{Y} \left\|D - YX^\top\right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r}, X = D^\top Y\left(Y^\top Y\right)^{-1} \tag{2.7}$$

$$\min_{Y} \left\|D - YY^\dagger D\right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r} \tag{2.8}$$

$$\min_{Y,X} \left\|D - YX^\top\right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r}, X \in \mathbb{R}^{n\times r} \tag{2.9}$$

$$\max_{Y} \operatorname{tr}\left(Z^\top DD^\top Z\right) \qquad s.t.\ Z = Y\left(Y^\top Y\right)^{-1/2}, Y \in \mathbb{1}^{m\times r} \tag{2.10}$$

$$\min_{Y} \left\|DD^\top - YY^\dagger\right\|^2 \qquad s.t.\ Y \in \mathbb{1}^{m\times r} \tag{2.11}$$

Proof. We have already shown in Section 2.3.1 that problem (KM) is equivalent to Eq. (2.7). The equivalence of Eq. (2.7) and Eq. (2.8) follows by definition of the matrix X in Eq. (2.7). The equivalence of problem (KM) and Eq. (2.9) follows from the convexity of the matrix factorization problem when the matrix Y is fixed. We show that for a given matrix $Y \in \mathbb{1}^{m\times r}$, the matrix $X = D^\top Y\left(Y^\top Y\right)^{-1}$ minimizes the residual sum of squares.

$$\begin{aligned}
\arg\min_X \left\|D - YX^\top\right\|^2 &= \arg\min_X -2\operatorname{tr}\left(X^\top D^\top Y\right) + \operatorname{tr}\left(XY^\top YX^\top\right) \\
&= \arg\min_X \sum_s -2X_{\cdot s}^\top D^\top Y_{\cdot s} + \|X_{\cdot s}\|^2 |Y_{\cdot s}| \\
&= \arg\min_X \sum_s |Y_{\cdot s}| \left\| D^\top \frac{Y_{\cdot s}}{|Y_{\cdot s}|} - X_{\cdot s} \right\|^2 .
\end{aligned} \tag{A.1}$$

The last equation shows that the minimum is attained when $X_{\cdot s} = D^\top \frac{Y_{\cdot s}}{|Y_{\cdot s}|}$.

The equivalence of Eq. (2.7) and Eq. (2.10) follows from the expansion

$$\left\|D - YY^{\dagger}D\right\|^2 = \|D\|^2 - 2\,\mathrm{tr}\left(D^{\top}Y\left(Y^{\top}Y\right)^{-1}Y^{\top}D\right) + \left\|Y\left(Y^{\top}Y\right)^{-1}Y^{\top}D\right\|^2,$$

and the following equation:

$$\begin{aligned}\left\|Y\left(Y^{\top}Y\right)^{-1}Y^{\top}D\right\|^2 &= \mathrm{tr}\left(D^{\top}Y\left(Y^{\top}Y\right)^{-1}Y^{\top}Y\left(Y^{\top}Y\right)^{-1}Y^{\top}D\right)\\ &= \mathrm{tr}\left(D^{\top}Y\left(Y^{\top}Y\right)^{-1}Y^{\top}D\right).\end{aligned}$$

Likewise, we can add the term

$$\left\|Y\left(Y^{\top}Y\right)^{-1}Y^{\top}\right\|^2 = \mathrm{tr}\left(Y\left(Y^{\top}Y\right)^{-1}Y^{\top}Y\left(Y^{\top}Y\right)^{-1}Y^{\top}\right) = \mathrm{tr}(I_r) = r$$

to the optimization problem in Eq. (2.10) and complete the square such that we obtain Eq. (2.11). □

THEOREM 2.7. *Define* $\Theta(A)$ *as the set*

$$\Theta(A) = \{B | B_{ji} = \theta(A_{ji}) \text{ for } A_{ji} \neq 1/2, B_{ji} \in \{0,1\} \text{ for } A_{ji} = 1/2\}$$

The following optimization problems are equivalent:

$$\min_{X,Y}\left\|D - YX^{\top}\right\|^2 \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \mathbb{1}^{m\times r} \tag{2.32}$$

$$\min_{Y}\left\|D - YX^{\top}\right\|^2 \qquad s.t.\ X \in \Theta\left(D^{\top}Y\left(Y^{\top}Y\right)^{-1}\right), Y \in \mathbb{1}^{m\times r} \tag{2.33}$$

$$\max_{X,Y}\mathrm{tr}\left(Y^{\top}(2D - \mathbf{1})X\right) \qquad s.t.\ X \in \{0,1\}^{n\times r}, Y \in \mathbb{1}^{m\times r} \tag{2.34}$$

Proof. Let (X, Y) be the solution to the problem of Eq. (2.32). From the convexity of the objective function when fixing Y, we derive from Eq. (A.1) that X is given as the binary matrix minimizing

$$\begin{aligned}&\operatorname*{arg\,min}_{X\in\{0,1\}^{n\times r}}\sum_{s,i}|Y_{\cdot s}|\left(D_{\cdot i}^{\top}\frac{Y_{\cdot s}}{|Y_{\cdot s}|} - X_{is}\right)^2\\ &= \left\{X\,\middle|\,X_{is}\begin{cases}= \theta(D_{i\cdot}^{\top}Y_{\cdot s}|Y_{\cdot s}|^{-1}) & \text{if } D_{i\cdot}^{\top}Y_{\cdot s}|Y_{\cdot s}|^{-1} \neq 1/2,\\ \in \{0,1\} & \text{otherwise}\end{cases}\right\}.\end{aligned}$$

The equivalence of Eq. (2.33) and Eq. (2.34) follows from the definition of the Frobenius inner product via the trace. □

APPENDIX B

Proofs of Chapter 5

We state here the more lengthy proofs of results presented in Chapter 6.

LEMMA 6.1. *Let D be a data matrix. For any code table CT and its cover function there exists a Boolean matrix factorization $D = \theta\big(Y X^\top\big) + N$ such that non-singleton patterns in CT are mirrored in X and the cover function is reflected by Y. The description lengths correspond to each other, such that*

$$L_{\mathsf{CT}}(CT) = L_{\mathsf{CT}}(X, Y) = L^D_{\mathsf{CT}}(X, Y) + L^M_{\mathsf{CT}}(X, Y),$$

where the functions returning the model and the data description size are given as

$$\begin{aligned} L^D_{\mathsf{CT}}(X, Y) &= -\sum_{s=1}^{r} |Y_{\cdot s}| \cdot \log(p_s) - \sum_{i=1}^{n} |N_{\cdot i}| \cdot \log(p_{r+i}) && = L^D_{\mathsf{CT}}(CT) \\ L^M_{\mathsf{CT}}(X, Y) &= \sum_{s:|Y_{\cdot s}|>0} \big(X_{\cdot s}^\top \mathbf{c} - \log(p_s)\big) + \sum_{i:|N_{\cdot i}|>0} (\mathbf{c}_i - \log(p_{r+i})) && = L^M_{\mathsf{CT}}(CT). \end{aligned}$$

The probabilities p_s and p_{r+i} indicate the relational usage of non-singleton patterns $X_{\cdot s}$ and singletons $\{i\}$,

$$p_s = \frac{|Y_{\cdot s}|}{|Y| + |N|}, \;\; p_{r+i} = \frac{|N_{\cdot i}|}{|Y| + |N|}.$$

We denote with $\mathbf{c} \in \mathbb{R}^n_+$ the vector of standard code lengths for each item, i.e.,

$$\mathbf{c}_i = -\log\left(\frac{|D_{\cdot i}|}{|D|}\right).$$

Proof. Let D be a data matrix, $CT = \{(\mathcal{X}_s, C_s) \mid 1 \leq s \leq r_0\}$ an r_0-element code table and *cover* the cover function. Let r be the number of non-singleton patterns

in CT and assume w.l.o.g. that CT is indexed such that these non-singleton patterns have an index $1 \leq s \leq r$. We construct the pattern matrix $X \in \{0,1\}^{n \times r}$ and usage matrix $Y \in \{0,1\}^{m \times r}$ such that for $1 \leq s \leq r$ it holds that

$$\begin{aligned} X_{is} = 1 &\Leftrightarrow i \in \mathcal{X}_s \\ Y_{js} = 1 &\Leftrightarrow \mathcal{X}_s \in \mathit{cover}(CT, D_{j\cdot}). \end{aligned}$$

The Boolean product $Y \odot X^\top$ indicates the entries of D which are covered by non-singleton patterns of CT. Nonzero elements in the noise matrix $N = D - \theta(YX^\top)$ are covered by singletons,

$$N_{ji} \neq 0 \Leftrightarrow \{i\} \in \mathit{cover}(CT, D_{j\cdot}).$$

The usage of a non-singleton pattern $\mathcal{X}_s$ is then computed as

$$\begin{aligned} \mathit{usage}(\mathcal{X}_s) &= |\{j \in \{1,\ldots,m\} \mid \mathcal{X}_s \in \mathit{cover}(CT, D_{j\cdot})\}| \\ &= |\{j \in \{1,\ldots,m\} \mid Y_{js} = 1\}| \\ &= |Y_{\cdot s}|. \end{aligned}$$

Correspondingly follows that $\mathit{usage}(\{i\}) = |N_{\cdot i}|$. The calculation of the probabilities p_s for $1 \leq s \leq r + n$ is directly obtained by inserting this usage calculation in the definition of code-usage-probabilities of Eq. (6.1). Likewise follow the definitions of the matrix functions $L^M_{\mathsf{CT}}(X,Y)$ and $L^D_{\mathsf{CT}}(X,Y)$ from the definitions of the description sizes $L^M_{\mathsf{CT}}(CT)$ and $L^D_{\mathsf{CT}}(CT)$. □

LEMMA B.1. *Let (a_s) be a finite sequence of r nonnegative scalars such that $S_r = \sum_{s=1}^r a_s > 0$. The function $g : [0,\infty) \to [0,\infty)$ defined by*

$$g(x; a_1, \ldots, a_r, S_r) = -\sum_{s=1}^{r} (a_s + x) \log\left(\frac{a_s + x}{S_r + rx}\right)$$

is monotonically increasing in x.

Proof. W.l.o.g., let $a_1, \ldots, a_{r_0} > 0$ and $a_{r_0+1} = \ldots = a_r = 0$ for some $r_0 \in \mathbb{N}$. We rewrite the function g as

$$g(x; a_1, \ldots, a_r, S_r) = g(x; a_1, \ldots, a_{r_0}, S_r) + g(x; a_{r_0+1}, \ldots, a_r, S_r)$$

and show that each of the subfunctions is monotonically increasing. The first subfunction is differentiable and its derivative is nonnegative

$$\begin{aligned}
&\frac{d}{dx} g(x; a_1, \ldots, a_{r_0}, S_r) \\
&= -\sum_{s=1}^{r} \left(\log\left(\frac{a_s + x}{S_r + rx}\right) + (a_s + x) \frac{S_r + rx}{a_s + x} \frac{S_r + rx - r(a_s + x)}{(S_r + rx)^2} \right) \\
&= -\sum_{s=1}^{r} \log\left(\frac{a_s + x}{S_r + rx}\right) + \sum_{s=1}^{r} \frac{S_r - ra_s}{S_r + rx} \\
&= -\sum_{s=1}^{r} \log\left(\frac{a_s + x}{S_r + rx}\right) \geq 0.
\end{aligned}$$

The second subfunction is monotonically increasing, since for $a_s = 0$ and all $x \geq 0$ it holds that

$$-a_s \log\left(\frac{a_s}{S_r}\right) = 0 \leq -(a_s + x) \log\left(\frac{a_s + x}{S_r + rx}\right).$$

□

THEOREM 6.2. *Given binary matrices X and Y and $\mu = 1 + \log(n)$, it holds that*

$$L^D_{\mathsf{CT}}(X, Y) \leq \mu \left\| D - YX^\top \right\|^2 - \sum_{s=1}^{r} (|Y_{\cdot s}| + 1) \log\left(\frac{|Y_{\cdot s}| + 1}{|Y| + r}\right) + |Y| \tag{6.3}$$

Proof. We recall that the description size of the data is computed as

$$L^D_{\mathsf{CT}}(X, Y) = \underbrace{-\sum_{s=1}^{r} |Y_{\cdot s}| \cdot \log\left(\frac{|Y_{\cdot s}|}{|Y| + |N|}\right)}_{=L_1(X,Y)} \underbrace{-\sum_{i=1}^{n} |N_{\cdot i}| \cdot \log\left(\frac{|N_{\cdot i}|}{|N| + |Y|}\right)}_{=L_2(X,Y)}.$$

Applying the logarithmic properties, we rewrite the first sum

$$\begin{aligned}
L_1(X, Y) &= -\sum_{s=1}^{r} |Y_{\cdot s}| \log\left(\frac{|Y_{\cdot s}|}{|Y|} \frac{|Y|}{|Y| + |N|}\right) \\
&= -\sum_{s=1}^{r} |Y_{\cdot s}| \log\left(\frac{|Y_{\cdot s}|}{|Y|}\right) + \sum_{s=1}^{r} |Y_{\cdot s}| \log\left(\frac{|Y| + |N|}{|Y|}\right) \\
&= g(0; |Y_{\cdot 1}|, \ldots, |Y_{\cdot r}|, |Y|) + |Y| \log\left(1 + \frac{|N|}{|Y|}\right).
\end{aligned}$$

From the monotonicity of g (Lemma B.1) and the logarithm inequality ($\log(1+x) \leq x, \forall x \geq 0$) follows that L_1 is upper bounded by

$$L_1(X,Y) \leq -\sum_{s=1}^{r}(|Y_{\cdot s}|+1)\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) + |N|.$$

The second term L_2 is transformed into

$$\begin{aligned} L_2(X,Y) &= -\sum_{i=1}^{n}|N_{\cdot i}| \cdot \log\left(|N_{\cdot i}|\right) + \sum_{i=1}^{n}|N_{\cdot i}| \cdot \log\left(|N|+|Y|\right) \\ &= \sum_{i=1}^{n}|N_{\cdot i}| \log\frac{1}{|N_{\cdot i}|} + |N|\log(|N|+|Y|). \end{aligned}$$

Subsequently, we show that $L_2(X,Y) \leq |N|\log(n)+|Y|$. This inequality trivially holds if $|N|=0$. Otherwise, we apply Jensen's inequality to the concave logarithm function

$$|N|\sum_{i=1}^{n}\frac{|N_{\cdot i}|}{|N|}\log\frac{1}{|N_{\cdot i}|} \leq |N|\log\left(\frac{n}{|N|}\right).$$

Therewith, we obtain

$$\begin{aligned} L_2(X,Y) &\leq |N|\log\left(\frac{n}{|N|}\right) + |N|\log(|N|+|Y|) \\ &= |N|\log(n) + |N|\log\left(1+\frac{|Y|}{|N|}\right) \\ &\leq |N|\log(n) + |Y|, \end{aligned}$$

where the last equality again follows from the logarithm inequality. We derive the final inequality

$$\begin{aligned} L^D_{\mathsf{CT}}(X,Y) &= L_1(X,Y) + L_2(X,Y) \\ &\leq (1+\log(n))|N| - \sum_{s=1}^{r}(|Y_{\cdot s}|+1)\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right) + |Y| \end{aligned}$$

□

PRIMP'S LIPSCHITZ MODULI We study the partial gradients of the regularization term used in PRIMP (Section 6.2.2)

$$\nabla_X G(X,Y) = \mathbf{c}\mathbf{1}^\top, \qquad \nabla_Y G(X,Y) = -\left(\log\left(\frac{|Y_{\cdot s}|+1}{|Y|+r}\right)\right)_{js} + \mathbf{1}.$$

The partial gradient with respect to X is constant and has a Lipschitz constant of zero. The partial gradient with respect to Y is equal to the sum

$$\nabla_Y G(X,Y) = -(\underbrace{(\log(|Y_{\cdot s}|+1))_{js}}_{=A(Y)} - \underbrace{(\log(|Y|+r))_{js}}_{=B(Y)}) + \mathbf{1}.$$

From the triangle inequality follows that the gradient with respect to Y is Lipschitz continuous with modulus $M_{\nabla_Y G}(X) = M_A + M_B$, if the functions A and B are Lipschitz continuous with moduli M_A and M_B:

$$\begin{aligned}\|\nabla_Y G(X,Y) - \nabla_V G(X,V)\| &= \|A(Y) - A(V) + B(Y) - B(V)\| \\ &\leq \|A(Y) - A(V)\| + \|B(Y) - B(V)\| \\ &\leq (M_A + M_B)\|Y - V\|.\end{aligned}$$

The one-dimensional function $x \mapsto \log(x+\delta)$, $x \in \mathbb{R}_+$ is for any $\delta > 0$ Lipschitz continuous with modulus δ^{-1}. This is easily derived by the mean value theorem and the bound

$$\frac{d}{dx}\log(x+\delta) = \frac{1}{x+\delta} \leq \frac{1}{\delta}$$

for all $x \geq 0$. We show with the following equations, that $M_A = M_B = m$. For improved readability, we use the squared Lipschitz inequality, i.e.,

$$\begin{aligned}\|A(Y) - A(V)\|^2 &= \sum_{s,j}(\log(|Y_{\cdot s}|+1) - \log(|V_{\cdot s}|+1))^2 \\ &= m\sum_{s=1}^{r}(\log(|Y_{\cdot s}|+1) - \log(|V_{\cdot s}|+1))^2 \\ &\leq m\sum_{s=1}^{r}(|Y_{\cdot s}| - |V_{\cdot s}|)^2 \qquad \text{(B.1)} \\ &= m\sum_{s=1}^{r}\left(\sum_{j=1}^{m}(Y_{js} - V_{js})\right)^2 \\ &\leq m^2\sum_{s,j}(Y_{js} - V_{js})^2 = m^2\|Y - V\|^2, \qquad \text{(B.2)}\end{aligned}$$

where Eq. (B.1) follows from the Lipschitz continuity of the logarithmic function as discussed above for $\delta = 1$ and Eq. (B.2) follows from the Cauchy-Schwarz inequality. Similar steps yield the Lipschitz modulus of B,

$$\begin{aligned}
\|B(Y) - B(V)\|^2 &= \sum_{s,j} (\log(|Y| + r) - \log(|V| + r))^2 \\
&= mr(\log(|Y| + r) - \log(|V| + r))^2 \\
&\leq \frac{mr}{r^2}(|Y| - |V|)^2 \\
&= \frac{m}{r}\left(\sum_{s,j}(Y_{js} - V_{js})\right)^2 \\
&\leq m^2 \sum_{s,j}(Y_{js} - V_{js})^2.
\end{aligned}$$

We conclude that the Lipschitz moduli of the gradients are given as

$$M_{\nabla_X G}(Y) = 0 \quad M_{\nabla_Y G}(X) = 2m.$$

Acknowledgement

I would like to thank and acknowledge first and foremost my awesome co-authors, from whom I learned so much just by working on a project together with them. This particularly includes my doctor mother Katharina Morik, who has such a broad knowledge of the disciplines in AI that I never ran out of interesting problems under her supervision. Her vision of data mining algorithms and their application in the future is often two steps ahead and inspired me to aim for the big questions in our field. She also introduced me to other inspiring scientists like Arno Siebes, whose elegant yet theoretically founded approach to pattern mining I already have been fangirling as a student. Arno later became my second dissertation reviewer (thank you) and I am glad to be therewith also a part of his academic family. The third member of my dissertation committee is Erich Schubert, whom I want to thank for animating me to go down the rabbit hole of nonconvex clustering of real-valued data. I am glad to have found another clustering enthusiast with him.

Further, I want to acknowledge the awesome team of LS8 at TU Dortmund with their desire for optimization inside and out: numerical, combinatorial, coffee-preparatory and pull-ups-executional. Part of that team is my long-term co-author Nico Piatkowski, whom I want to thank for his enthusiasm and his *good nose* for interesting methods and approaches. Discussing with him is always an overlength but delightful experience. With my other co-author Wouter Duivesteijn, I later did more than just research. Here, I want to thank him for teaching me writing skills and for having an open ear when I want to discuss theory during the weekend <3. Also, as always, one thing is important to mention:

My work has been supported by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, project C1.

List of Figures

2.1 Variants of row- and column-partitioning biclusters: checkerboard model (left), plaid model (middle) and block-diagonal model (right). Best viewed in color. 22

3.1 Plot of the Mexican hat function used as a penalizing function for nonbinary values. 40

4.1 Performance of spectral clustering, DBSCAN, and SpectACl on two concentric circles. Best viewed in color. 42
4.2 Visualization of every fifth eigenvector for the two circles dataset. The size of each point is proportional to the absolute value of the corresponding entry in the eigenvector. 46
4.3 Variation of noise, comparison of F-measures (the higher the better) for the two moons (left), the two circles (middle) and three blobs (right) datasets. 49
4.4 Cluster visualizations of regarded algorithms and synthetic datasets. Best viewed in color. 50
4.5 Variation of latent parameters used for SpectACl. The neighborhood radius ϵ (top left) and the number of nearest neighbors k (top right) determine the adjacency matrix for SpectACl and its normalized version. The parameter d (bottom left and right) specifies the number of projected eigenvectors. We plot the F-measure (the higher the better) against the variation of the parameter. 51
4.6 SpectACl clusters individual handwriting styles instead of cipher shapes. 53

5.1 Results of a Boolean decomposition of rank three on a picture dataset, depicted on the left. On the top row, the picture is approximated by a greedy approach, on the bottom we display the result of the proposed proximal optimization scheme for BMF. Best viewed in color. 56

5.2 Approximation of a binary matrix D with two overlapping tiles (top) applying NMF (second from above) and the factorizations resulting from thresholding the factor matrices to binary matrices in elementary algebra (second from below) and Boolean algebra (below). Tiles are highlighted. 57
5.3 A lower semi-continuous function. 60
5.4 The function Λ penalizing nonbinary values. 63

6.1 Variation of uniform noise for 800×1000- and 1000×800-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., $p^*_+ = p^*_-$ is indicated on the x-axis (best viewed in color). 84
6.2 Variation of uniform noise for 500×1600- and 1600×500-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., $p^*_+ = p^*_-$ is indicated on the x-axis (best viewed in color). 85
6.3 Variation of uniform, positive and negative noise. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise, i.e., p^*_+ and p^*_- are indicated on the x-axis (best viewed in color). 86
6.4 Variation of the rank $r^* \in \{5, \ldots, 45\}$ of the generated factorization. Comparison of F-measures (the higher the better) and estimated rank (the closer to the identity function the better) of calculated Boolean matrix factorizations for uniform noise of $p^*_+ = p^*_- = 10\%$ (best viewed in color). 87
6.5 Variation of density and overlap influencing parameter $\xi_{max} \in [0.1, \ldots, 0.3]$. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for uniform noise of $p^*_+ = p^*_- = 10\%$ (best viewed in color). . 88
6.6 Variation of rank increment $\Delta_r \in \{2, 5, 10, 20\}$. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for uniform noise of $p^*_+ = p^*_-$ indicated by the x-axis (best viewed in color). 89
6.7 Reconstructions of the Alice image and visualizations of the top-4 outer products. Best viewed in color. 95
6.8 Reconstructions of the Space Invaders image and visualizations of the top-4 outer products. Best viewed in color. 96

7.1 Minimum relative size $|Y_{\cdot s}|/m$, depending on $|X_{\cdot s}|/n$, for which $P(\mathbf{Z}_s = 1) \leq 0.01$, based on density (blue) and coherence (green). 108
7.2 Variation of uniform noise for 1600×500- and 1000×800-dimensional data. Comparison of F-measures (the higher the better) and the estimated rank of the calculated Boolean factorization (the closer to 25 the better) for varying levels of noise indicated on the x-axis. . . 111
7.3 Variation of density and overlap influencing parameter $\xi_{max} \in [0.1, \ldots, 0.3]$. Comparison of F-measures (the higher the better) and the estimated rank (the closer to 25 the better) for uniform noise of $p^*_{\pm} = 10\%$. . . 112

8.1 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. 118
8.2 A Boolean product identifying common (green) and discriminative outer products (blue). Best viewed in color. 119
8.3 Variation of noise, class distribution $\frac{m_1}{m}$ and the rank. 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. 127
8.4 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. 128
8.5 Illustration of a selection of derived topics for the 20 News articles from classes *space* and *religion*. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^\top D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a space and green a religion article. Best viewed in color. 130
8.6 Illustration of a selection of derived topics for the 20 News articles from classes *mideast politics* and *miscellaneous politics*. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^\top D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a mideast and green a miscellaneous politics article. Best viewed in color. 131
8.7 Illustration of a selection of derived topics for the Movies dataset, comprising *bad* and *good* reviews. The size of a word reflects its frequency in the topic ($\sim Y_{\cdot s}^\top D_{\cdot i}$) and the color its class affiliation: pink words are common among both classes, blue indicates a bad and green a good review. Best viewed in color. 131

8.8 Transposed usage matrix returned by C-SALT on the genome dataset. Class-memberships are signalized by colors. 132

List of Algorithms

1 Computing a solution to the binary matrix factorization problem in Eq. (2.32) via the greedy approach. 34
2 Lloyd's Alternating Minimization for k-means 35
3 Clustering via orthogonal nonnegative matrix factorization 37
4 The orthogonal relaxation of spectral clustering. 38
5 Binary Penalty Algorithm 39
6 Spectral Averagely Dense Clustering 46
7 The proposed general optimization scheme PAL-Tiling for Boolean matrix factorizations. The implementation of the function Toss, used in the function Round, specifies which tiles are kept. the specification of this method determines the rank selection strategy. 66
8 The tossing function for the application of PAL-Tiling implementing the determination of the rank via the minimum description length. 77
9 Generation of synthetic datasets for Boolean matrix factorizations. 82
10 The tossing functions for the application of PAL-Tiling implementing the determination of the rank via FDR control. 109
11 The PAL-Tiling extension C-Salt, mining class-specific alterations in a labeled binary database. 122
12 Generation of synthetic datasets for Boolean matrix factorizations with class-specific alterations. 126

Bibliography

C. C. Aggarwal and J. Han. *Frequent pattern mining*. Springer, 2014.

C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.

M. Asteris, D. Papailiopoulos, and A. G. Dimakis. Orthogonal NMF through subspace exploration. In *Advances in neural information processing systems (NIPS)*, pages 343–351, 2015.

H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of operations research*, 35 (2):438–457, 2010.

R. Belohlavek and M. Trnecka. From-below approximations in Boolean matrix factorization: geometry and new algorithm. *Journal of computer and system sciences*, 81(8):1678–1697, 2015.

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*, 76(1):3–20, 2010.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of machine learning research*, 3(Jan):993–1022, 2003.

A. Bojchevski, Y. Matkovic, and S. Günnemann. Robust spectral clustering for noisy data: modeling sparse corruptions improves latent embeddings. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 737–746, 2017.

J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical programming*, 146(1-2): 459–494, 2014.

S. Busygin, O. Prokopyev, and P. M. Pardalos. Biclustering in data mining. *Computers & operations research*, 35(9):2964–2987, 2008.

M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013.

S. Chawla and A. Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 189–197. SIAM, 2013.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the eighth international conference on intelligent systems for molecular biology*, volume 8, pages 93–103, 2000.

H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 114–125. SIAM, 2004.

F. R. Chung. *Spectral graph theory*. American Mathematical Soc., 1997.

A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

L. Collatz. Spektren periodischer Graphen. *Results in mathematics*, 1(1-2):42–53, 1978.

T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

D. Davis, B. Edmunds, and M. Udell. The sound of APALM clapping: faster nonsmooth nonconvex optimization with stochastic asynchronous PALM. In *Advances in neural information processing systems (NIPS)*, pages 226–234, 2016.

T. De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data mining and knowledge discovery (DAMI)*, 23 (3):407–446, 2011.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 269–274. ACM, 2001.

C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 606–610. SIAM, 2005.

C. Ding, T. Li, and W. Peng. Nonnegative matrix factorization and probabilistic latent semantic indexing: equivalence chi-square statistic, and a hybrid method. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 342–347, 2006a.

C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 126–135. ACM, 2006b.

C. Ding, T. Li, and M. I. Jordan. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *IEEE international conference on data mining (ICDM)*, pages 183–192. IEEE, 2008.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

H. E. Driver and A. L. Kroeber. *Quantitative expression of cultural relationships*, volume 31. University of California Press, 1932.

W. Duivesteijn and A. Knobbe. Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In *IEEE international conference on data mining (ICDM)*, pages 151–160. IEEE, 2011.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, volume 96, pages 226–231, 1996.

L. M. Fagan, E. H. Shortliffe, and B. G. Buchanan. Computer-based medical decision making: from MYCIN to VM. *Automedica*, 3(2):97–108, 1980.

K. Fan. On a theorem of Weyl concerning eigenvalues of linear transformations I. *Proceedings of the national academy of sciences*, 35(11):652–655, 1949.

S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

W. Gaul and M. Schader. A new algorithm for two-mode clustering. In *Data analysis and information systems*, pages 15–23. Springer, 1996.

E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *Proceedings of the annual international ACM SIGIR conference on research and development in information retrieval*, pages 601–602. ACM, 2005.

F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *International conference on discovery science (DS)*, pages 278–289. Springer, 2004.

G. Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL `http://distill.pub/2017/momentum`.

L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. *Operations research letters*, 26(3): 127–136, 2000.

P. D. Grünwald. *The minimum description length principle*. MIT press, 2007.

S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM journal on matrix analysis and applications*, 19(3):701–719, 1998.

S. Gudder and F. Latrémolière. Boolean inner-product spaces and Boolean matrices. *Linear algebra and its applications*, 431(1-2):274–296, 2009.

S. K. Gupta, D. Phung, B. Adams, T. Tran, and S. Venkatesh. Nonnegative shared subspace learning and its application to social media retrieval. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 1169–1178, 2010.

S. K. Gupta, D. Phung, B. Adams, and S. Venkatesh. Regularized nonnegative shared subspace learning. *Data mining and knowledge discovery (DAMI)*, 26(1): 57–97, 2013.

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.

L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

J. Han, K. Song, F. Nie, and X. Li. Bilateral k-means algorithm for fast co-clustering. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 1969–1975, 2017.

M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the international conference on machine learning (ICML)*, pages 1225–1234, 2016.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American statistical association*, 67(337):123–129, 1972.

P. Hennig. Fast probabilistic optimization from noisy gradients. In *Proceedings of the international conference on machine learning (ICML)*, pages 62–70, 2013.

P. Hennig. Probabilistic interpretation of linear solvers. *SIAM journal on optimization*, 25, Jan. 2015.

P. Hennig and M. Kiefel. Quasi-Newton methods – a new direction. In *Proceedings of the international conference on machine learning (ICML)*, 2012.

S. Hess and K. Morik. C-SALT: mining class-specific alterations in Boolean matrix factorization. In *Joint European conference on machine learning and knowledge discovery in databases (ECML PKDD)*, pages 547–563. Springer, 2017.

S. Hess, N. Piatkowski, and K. Morik. SHrimp: descriptive patterns in a tree. In *Proceedings of the LWA workshops: KDML, IR and FGWM.*, pages 181–192, 2014.

S. Hess, K. Morik, and N. Piatkowski. The PRIMPING routine – tiling through proximal alternating linearized minimization. *Data mining and knowledge discovery (DAMI)*, 31(4):1090–1131, 2017.

S. Hess, N. Piatkowski, and K. Morik. The trustworthy pal: controlling the false discovery rate in Boolean matrix factorization. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 405–413. SIAM, 2018.

S. Hess, W. Duivesteijn, P. Honysz, and K. Morik. The SpectACl of nonconvex clustering: a spectral approach to density-based clustering. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 3788–3795, 2019.

E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1731–1741, 2017.

A. Holzinger, R. Goebel, V. Palade, and M. Ferri. Towards integrative machine learning and knowledge extraction. In *Towards integrative machine learning and knowledge extraction*, pages 1–12. Springer, 2017.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *IEEE international conference on computer vision (ICCV)*, pages 2146–2153. IEEE Computer Society, 2009.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In *Proceedings of the international conference on machine learning (ICML)*, pages 1724–1732, 2017.

Z. Kang, C. Peng, Q. Cheng, and Z. Xu. Unified spectral clustering with optimal graph. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 3366–3373, 2018.

S. Karaev, P. Miettinen, and J. Vreeken. Getting to know the unknown unknowns: destructive-noise resistant Boolean matrix factorization. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 325–333. SIAM, 2015.

J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine learning*, 14(2):193–217, 1994.

J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive logic programming*, number 38, pages 335–360. 1992.

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 *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 567–576, 2015.

J. Kim, Y. He, and H. Park. Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. *Journal of global optimization*, 58(2):285–319, 2014.

S. Klimek. *Culture element distributions: I. The structure of California Indian culture*. University of California Publications in American Archaeology and Ethnology, 1935.

J. Komiyama, M. Ishihata, H. Arimura, T. Nishibayashi, and S.-i. Minato. Statistical emerging pattern mining with multiple testing correction. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 897–906. ACM, 2017.

K.-N. Kontonasios and T. De Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 153–164. SIAM, 2010.

M. Koyutürk and A. Grama. PROXIMUS: a framework for analyzing very high dimensional discrete-attributed datasets. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 147–156. ACM, 2003.

H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM transactions on knowledge discovery from data (TKDD)*, 3(1):1, 2009.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica sinica*, pages 61–86, 2002.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems (NIPS)*, pages 556–562, 2001.

M. v. Leeuwen and A. Siebes. Streamkrimp: detecting change in data streams. In *Joint European conference on machine learning and knowledge discovery in databases (ECML PKDD)*, pages 672–687, 2008.

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in neural information processing systems (NIPS)*, pages 379–387, 2015.

M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.*, volume 9. Springer, New York, 2008.

T. Li. A general model for clustering binary data. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery in data mining (KDD)*, pages 188–197. ACM, 2005.

T. Li and C. Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *IEEE international conference on data mining (ICDM)*, pages 362–371. IEEE, 2006.

C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

F. Llinares-López, M. Sugiyama, L. Papaxanthos, and K. Borgwardt. Fast and memory-efficient significant pattern mining via permutation testing. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 725–734. ACM, 2015.

S. Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

C. Lucchese, S. Orlando, and R. Perego. Mining top-k patterns from binary datasets in presence of noise. In *Proceedings of the SIAM international conference on data mining (SDM)*, volume 10, pages 165–176. SIAM, 2010.

C. Lucchese, S. Orlando, and R. Perego. A unifying framework for mining approximate top-k binary patterns. *IEEE transactions on knowledge and data engineering (TKDE)*, 26(12):2900–2913, 2014.

U. v. Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics (TCBB)*, 1(1):24–45, 2004.

V. Maurizio. Double k-means clustering for simultaneous classification of objects and variables. In *Advances in classification and data analysis*, pages 43–52. Springer, 2001.

I. v. Mechelen, H.-H. Bock, and P. De Boeck. Two-mode clustering methods: a structured overview. *Statistical methods in medical research*, 13(5):363–394, 2004.

P. Miettinen. Sparse Boolean matrix factorizations. In *IEEE international conference on data mining (ICDM)*, pages 935–940. IEEE, 2010.

P. Miettinen. On finding joint subspace Boolean matrix factorizations. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 954–965, 2012.

P. Miettinen. Generalized matrix factorizations as a unifying framework for pattern set mining: complexity beyond blocks. In *Joint European conference on machine learning and knowledge discovery in databases (ECML PKDD)*, pages 36–52. Springer, 2015.

P. Miettinen and J. Vreeken. MDL4BMF: minimum description length for Boolean matrix factorization. *ACM transactions on knowledge discovery from data (TKDD)*, 8(4):18, 2014.

P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *IEEE transactions on knowledge and data engineering (TKDE)*, 20(10):1348–1362, 2008.

B. Mirkin, P. Arabie, and L. J. Hubert. Additive two-mode clustering: the error-variance approach revisited. *Journal of classification*, 12(2):243–263, 1995.

B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann. The Laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991.

K. Morik, J.-U. Kietz, W. Emde, and S. Wrobel. *Knowledge acquisition and machine learning – theory, methods, and applications*. Academic Press, 1993.

S. Muggleton. *Inductive logic programming*. Number 38 in APIC series. Academic Press, 1992.

L. Mukherjee, S. N. Ravi, V. K. Ithapu, T. Holmes, and V. Singh. An NMF perspective on binary hashing. In *IEEE International conference on computer vision (ICCV)*, pages 4184–4192, 2015.

F. Nie, X. Wang, C. Deng, and H. Huang. Learning a structured optimal bipartite graph for co-clustering. In *Advances in neural information processing systems (NIPS)*, pages 4129–4138, 2017.

S. Nikitidis, A. Tefas, and I. Pitas. Projected gradients for subclass discriminant nonnegative subspace learning. *IEEE transactions on cybernetics*, 44(12):2806–2819, 2014.

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.

P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

M. P. Pacifico, C. Genovese, I. Verdinelli, and L. Wasserman. Scan clustering: a false discovery approach. *Journal of multivariate analysis*, 98(7):1441–1469, 2007.

N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and trends in optimization*, 1(3):127–239, 2014.

L. Pfahler, K. Morik, F. Elwert, S. Tabti, and V. Krech. Learning low-rank document embeddings with weighted nuclear norm regularization. In *IEEE international conference on data science and advanced analytics (DSAA)*, pages 21–29. IEEE, 2017.

N. Piatkowski. *Exponential families on resource-constrained systems.* PhD thesis, Technical University of Dortmund, Germany, 2018.

T. Pock and S. Sabach. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems. *SIAM journal on imaging sciences*, 9(4):1756–1787, 2016.

F. Pompili, N. Gillis, P.-A. Absil, and F. Glineur. Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. *Neurocomputing*, 141:15–25, 2014.

I. Ramírez. Binary matrix factorization via dictionary learning. *IEEE journal of selected topics in signal processing*, 12(6):1253–1262, 2018.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

J. v. Rosmalen, P. J. Groenen, J. Trejos, and W. Castillo. Optimization strategies for two-mode partitioning. *Journal of classification*, 26(2):155–181, 2009.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

A. Schramm, J. Köster, Y. Assenov, K. Althoff, M. Peifer, E. Mahlow, A. Odersky, D. Beisser, C. Ernst, A. G. Henssen, et al. Mutational dynamics between primary and relapse neuroblastomas. *Nature genetics*, 47(8):872, 2015.

E. Schubert, S. Hess, and K. Morik. The relationship of DBSCAN to matrix factorization and spectral clustering. In *Proceedings of the conference "Lernen, Wissen, Daten, Analysen" (LWDA)*, pages 330–334, 2018.

B.-H. Shen, S. Ji, and J. Ye. Mining discrete patterns via binary matrix factorization. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 757–766. ACM, 2009.

F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao. A fast optimization method for general binary code learning. *IEEE transactions on image processing*, 25(12):5610–5621, 2016.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

A. Siebes and R. Kersten. A structure function for transaction data. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 558–569. SIAM, 2011.

A. Siebes, J. Vreeken, and M. v. Leeuwen. Item sets that compress. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 395–406. SIAM, 2006.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

M. Slawski, M. Hein, and P. Lutsik. Matrix factorization with binary components. In *Advances in neural information processing systems (NIPS)*, pages 3210–3218, 2013.

K. Smets and J. Vreeken. Slim: directly mining descriptive patterns. In *Proceedings of the SIAM international conference on data mining (SDM)*, pages 236–247. SIAM, 2012.

N. Tatti and J. Vreeken. Comparing apples and oranges: measuring differences between exploratory data mining results. *Data mining and knowledge discovery (DAMI)*, 25(2):173–207, 2012.

C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage. Descriptive matrix factorization for sustainability adopting the principle of opposites. *Data mining and knowledge discovery (DAMI)*, 24(2):325–354, 2012.

R. C. Tryon. *Cluster analysis: correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality*. Edwards brother, Incorporated, lithoprinters and publishers, 1939.

H. Turner, T. Bailey, and W. Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational statistics & data analysis*, 48(2):235–254, 2005.

M. Udell, C. Horn, R. Zadeh, S. Boyd, et al. Generalized low rank models. *Foundations and trends in machine learning*, 9(1):1–118, 2016.

L. van den Dries. *Tame topology and o-minimal structures*, volume 248 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1998.

S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM journal on optimization*, 20(3):1364–1377, 2009.

J. Vreeken, M. v. Leeuwen, and A. Siebes. Characterising the difference. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pages 765–774. ACM, 2007.

J. Vreeken, M. v. Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data mining and knowledge discovery (DAMI)*, 23(1):169–214, 2011.

H. Wang, F. Nie, H. Huang, and F. Makedon. Fast nonnegative matrix tri-factorization for large-scale data co-clustering. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)*, page 1553, 2011.

Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: a comprehensive review. *IEEE transactions on knowledge and data engineering (TKDE)*, 25(6): 1336–1353, 2013.

G. I. Webb. Discovering significant patterns. *Machine learning*, 68(1):1–33, 2007.

J. J. Whang and I. S. Dhillon. Non-exhaustive, overlapping co-clustering. In *Proceedings of the ACM conference on information and knowledge management (CIKM)*, pages 2367–2370. ACM, 2017.

J. J. Whang, Y. Hou, D. Gleich, and I. S. Dhillon. Non-exhaustive, overlapping clustering. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Summarizing transactional databases with overlapped hyperrectangles. *Data mining and knowledge discovery (DAMI)*, 23(2):215–251, 2011.

J. Yoo and S. Choi. Orthogonal nonnegative matrix tri-factorization for co-clustering: multiplicative updates on Stiefel manifolds. *Information processing & management*, 46(5):559–570, 2010.

H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the international conference on information and knowledge management*, pages 25–32. ACM, 2001.

H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems (NIPS)*, pages 1057–1064, 2002.

Z. Zhang, T. Li, C. Ding, and X. Zhang. Binary matrix factorization with applications. In *IEEE international conference on data mining (ICDM)*, pages 391–400. IEEE, 2007.

Z.-Y. Zhang, T. Li, C. Ding, X.-W. Ren, and X.-S. Zhang. Binary matrix factorization for analyzing gene expression data. *Data mining and knowledge discovery (DAMI)*, 20(1):28, 2010.

Z.-Y. Zhang, Y. Wang, and Y.-Y. Ahn. Overlapping community detection in complex networks using symmetric binary matrix factorization. *Physical review E*, 87(6):062803, 2013.

A. Zimek and J. Vreeken. The blind men and the elephant: on meeting the problem of multiple truths in data from clustering and pattern mining perspectives. *Machine learning*, 98(1-2):121–155, 2015.