Universität des Saarlandes
Max-Planck-Institut für Informatik
AG5

Matrix factorization over max-times algebra for data mining

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

Sanjar Karaev

angefertigt unter der Leitung von / supervised by

Dr. Pauli Miettinen

begutachtet von / reviewers

Dr. Pauli Miettinen

Prof. Gerhard Weikum

November 2013

Hilfsmittelerklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Non-plagiarism Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, den 07. November 2013,

(Sanjar Karaev)

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlichtwird.

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department at Saarland University.

Saarbrücken, den 07. November 2013,

(Sanjar Karaev)

To my family

- Sanjar

Abstract

Decomposing a given matrix into two factor matrices is a frequently used technique in data mining for uncovering underlying latent patterns in the data. Unlike in pure mathematics, the emphasis is put on obtaining results that are interpretable, rather than necessarily having a small reconstruction error. One approach to increase interpretability is to pose constraints on the factors. For example they might be restricted to the same type as the original matrix.

Among many different ways one can define matrix multiplication, the standard and Boolean cases are the most thoroughly studied. In this work we introduce matrix multiplication over max-times algebra, which is the set of nonnegative real numbers endowed with standard multiplication, but with addition being replaced by the maximization operation. The main objective of the thesis is to develop efficient factorization algorithms for this newly defined matrix multiplication.

We propose several methods for solving this problem. The choice of a particular algorithm depends on the nature of the data, in particular its density. The sparser the matrices, the closer their max-multiplication is to the standard matrix multiplication, and for extremely sparse data even an algorithm for nonnegative matrix factorization can be quite good. However, for other cases algorithms that are specifically tailored for max-times data are required.

The max matrix factorization problem is hard, and solving the problem precisely for large input sizes is infeasible, which means that approximate methods should be sought. It turned out in the experiments that the most promising approach is to relax the objective in such way that it becomes differentiable, after which convex optimization algorithms (e.g. gradient descent) can be used. This produces relatively good results when the max-times structure is present in the data.

Acknowledgements

I would like to express my deep gratitude to Dr. Pauli Miettinen, who paid very close attention to my work and frequently gave valuable advice. He provided strong guidance from the very start and paid attention to every detail in his comments. He also gave me sufficient freedom to try my own ideas and to discover where they lead. Thanks to him, I have learned a lot, not only on the topic of my thesis, but also on how to conduct scientific research in general, which is probably even more important.

This thesis was written during my studies at Saarland University in Saarbrücken, Germany. I received funding from the Saarbrücken Graduate School of Computer Science, and later from Max-Planck Institute of Informatics. I cordially thank both organizations for their financial support, which allowed me to concentrate on my studies.

Writing a thesis is a long road, which takes a lot of energy as well as emotional resources. It is vital that along this road one can rely on someone whom they trust and who can understand them well. I am pleased to thank my friends who were supporting me throughout my work.

Finally, and most importantly, I would like to thank my family, especially my parents. Without their love and support it would be impossible for me to accomplish even a tiny part of what I have achieved so far. They taught me how to live my life in integrity and how to achieve my goals no matter the circumstance.

Contents

Chapter 1

Introduction

Data mining can be viewed as a collection of techniques for extracting interesting patterns from large bodies of data [HKP06]. In the last couple decades the amount of data collected in almost all areas of human endeavour has been increasing rapidly, making it virtually impossible to analyze it manually. This has led to the development of a wide range of methods that facilitate the process of discovering new knowledge from data repositories. The ultimate goal of data mining is to present users with extracted interesting patterns in a form they will be able to understand.

Matrices are a natural and very convenient way of representing data, which is the reason why they are ubiquitous in data mining. The data mining view of a matrix is different from that of linear algebra, that considers matrices as linear operators from one linear space to another [GVL12].

Matrix decomposition is a problem of finding two or more factor matrices whose product is equal to the original matrix. There are some variations to this definition, for example, rather than seeking a precise decomposition, the problem might be to find the best low-rank approximation with respect to some norm. Some classic examples of matrix decompositions are SVD and QR decomposition [GVL12]. In data mining matrix decompositions play a crucial role and are valuable tools allowing better interpretation of data. What differs matrix decompositions in data mining from linear algebra is that domain knowledge is used to obtain results that are more useful to the user, and interpretability is valued higher than obtaining a precise factorization. There are many ways to increase interpretability [Mie09], among which are making factor matrices considerably smaller than the original matrix or restricting the entries in factors to the same domain as the original one (if, say, all values in the matrix being decomposed are nonnegative real numbers, then so are the entries in the factors). A simple example of matrix factorization in data mining comes from information retrieval.

Interpretability of the results is of primary importance in data mining: a method effective in uncovering patterns is practically useless if its output cannot be understood by the end user. It is worth mentioning that filtering interesting patterns and data presentation are not usually considered parts of data mining, but rather are separate steps in the Knowledge Discovery in Databases (KDD) process [HKP06]. However, when designing a data mining algorithm, the issue of interpretability still plays a very important role. Quite often we are even willing to sacrifice some characteristics of a method (e.g. precision or speed) for the sake of making the results easier to understand. For example, in the context of matrix factorization it is usually not imperative to obtain a precise decomposition for two reasons: first, patterns can often be extracted without necessarily having a zero recovery error, and second, the data is often noisy, which means that by driving the error to zero we are effectively fitting the noise.

The topic of this thesis is matrix decompositions over max-times algebra, which is defined as an algebra over the set of nonnegative real numbers with standard multiplication, but addition being replaced by the maximum operation. That is for any two numbers $a,\ b \geq 0$ we have $a \boxplus b = \max\{a, b\}$. Informally, one can think of it as an extension of logical operations from $\{0, 1\}$, where 1 stands for `TRUE` and 0 for `FALSE`, to the set of all nonnegative numbers. Indeed, the logical addition is defined as `OR`, that is $1 + 1 = 1,\ 0 + 1 = 1$ and $1 + 0 = 1$, which is the same as taking the maximum. Hence, maximization operation in max-times algebra can be viewed as a generalization of the logical addition.

There exist many domains where the max-times algebraic data occurs naturally. The common characteristic of such data is that one feature dominates all the rest and determines the outcome. Think of an auction where the amount of money paid for an item is determined by a highest bid.

A more sophisticated example of data with max-times algebraic structure comes from the entertainment area. Consider a customer review web page where users give ratings to movies they have watched. All movies have features (genre, actors playing main characters, director, special effects, etc.). Assume that all users have to give each film a single score. Though different people have different voting patterns, it is quite unlikely that the majority would just give a movie the average of the scores of its features. It appears much more plausible that (apart from movie critics) they would rate a movie based on its best feature (say, somebody might give The Hobbit an outstanding score just because he happens to be a J. R. R. Tolkien fan). More precisely, let B be a matrix with rows denoting users and columns denoting features. It represents how important is each feature to a particular user. Let another matrix W represent how much a movie scores for each feature. Then, if our assumption about the highest scoring feature is

correct, the aggregate rating will be given by the max-times product of matrices B and W.

The contribution of this thesis is the development of algorithms for matrix factorization over max-time algebra and their application to real-world data. Also, the complexity analysis of the factorization problem is provided. The algorithms created for the thesis can be broadly divided into two groups. The first group consists of methods that attempt to solve the factorization problem as it is, and are called direct. However, the max-times factorization problem is hard, which motivates looking for approximate relaxed methods. Algorithms comprising the second group are using the latter approach, and for this reason they are called relaxed.

As has been mentioned above, the notion of a "good" factorization in the realm of data mining is not the same as in pure linear algebra. Whereas in linear algebra we are interested in obtaining an exact decomposition or in minimizing the approximation error, an effective data mining method aims to find underlying patterns in the data and does not necessarily produce a very small error. For example, while experimenting with the Yale "eigenfaces" database, which contains pictures of people under different lightning conditions (see e.g. [BHK97]), we were able to extract meaningful features of human faces even when the reconstruction error was relatively high.

Data mining is supposed to deal with high quantities of data, where it becomes incomprehensible for humans, and hence, scalability is of primary importance for most data mining methods. We tested the presented algorithms on large amounts of data, and (to be written when we know the final results).

Structure of the thesis. Chapter 2 introduces the necessary background and notation for the thesis. Here we also formulate the max matrix factorization problem. Then all the related work is discussed in Chapter 3. Chapter 4 contains theoretical results on the MMF problem, which include rank, sparsity, computational complexity and a relation to the max-times algebra. All the new algorithms are introduced in Chapters 5 and 6. Chapter 5 is dedicated to algorithms `Greedy`, `SDDMMF` and `SDDUNDERFIT` that solve the original problem directly. Chapter 6 discusses methods that are based on the exponential relaxation technique. Experimental evaluation of the algorithms is performed in Chapter 7, and Chapter 8 is the conclusion.

Chapter 2

Notation and Definitions

This chapter introduces the notation used in the thesis and provides the background necessary for further reading. Here we also give a formal definition of the matrix factorization problem over max-times algebra.

2.1 Background

In this section we introduce the minimal knowledge from linear algebra, theory of matrices and convex analysis that is fundamental for understanding the material covered in the thesis.

2.1.1 Linear Analysis

Definition 2.1.1. Given linear spaces L and U over a field $\mathbb{F}$, where $\mathbb{F}$ is either real or complex numbers, a map $A\colon L \to U$ is a *linear operator* if it satisfies the following axioms for all $c \in \mathbb{F}$ and $v, w \in L$:

1. $A(cv) = cAv$, homogeneity,
2. $A(v + w) = Av + Aw$, additivity.

In this work we extensively use different matrix norms for measuring the errors of factorization algorithms. One can think of a norm as a length of a vector in a linear space. The formal definition is as follows:

Definition 2.1.2. Given a linear space L over a field $\mathbb{F}$, where $\mathbb{F}$ is either real or complex numbers, a *norm* is a map $\|\cdot\|\colon L \to \mathbb{R}_+$ that satisfies the following axioms for all $c \in \mathbb{F}$ and $v, w \in L$:

1. $\|cv\| = |c| \, \|v\|$, homogeneity of the norm,
2. $\|v + w\| \leq \|v\| + \|w\|$, triangle inequality,
3. $\|v\| = 0$ if and only if $v = 0$, separating points.

Definition 2.1.3 ([RS81])**.** Let L be a linear space over $\mathbb{R}$ or $\mathbb{C}$. Two norms $\|\cdot\|_1$ and $\|\cdot\|_2$ are said to be *equivalent* if there exist constants $C_1, C_2 > 0$ such that

$$C_1\|x\|_1 \leq \|x\|_2 \leq C_2\|x\|_1 \tag{2.1}$$

for all $x \in L$.

The following result is useful for matrix analysis since all finite dimensional matrices can be viewed as finite dimensional vectors.

Lemma 2.1.1 ([RS81])**.** All norms on $\mathbb{R}^n$ are equivalent.

For more information on linear spaces, linear operators and norms see e.g. a monograph by Dunford and Schwartz [DS58].

There are many different vector norms, among which some of the most common are l_1 and l_2 or Euclidean, norms, which are defined for a vector $x \in \mathbb{F}^n$ as $\|x\|_1 = \sum\limits_{1\leq i\leq n} |x|_i$ and $\|x\|_2 = \sqrt{\sum\limits_{1\leq i\leq n} |x|_i^2}$, respectively. Though the Definition 2.1.2 is given for vectors, it can readily be extended to matrices: we only need to observe that a matrix of size n-by-m can be viewed as a vector of size nm. Perhaps the most well known and ubiquitous is the Frobenius matrix norm (see [GVL12]).

Definition 2.1.4. *Frobenius norm* of an n-by-m matrix A is defined as

$$\|A\|_F^2 = \sum_{\substack{1\leq i\leq n \\ 1\leq j\leq m}} A_{ij}^2 \, . \tag{2.2}$$

The Frobenius norm of a matrix is nothing else than the standard Euclidean norm of a vector composed of its elements. Another important norm, which, together with the Frobenius norm we use for measuring the error is the sum of absolute values of a matrix. We denote it by $\|\cdot\|_1$, analogous to the $\|\cdot\|_1$ vector norm [HJ12].

Definition 2.1.5. $\|\cdot\|_1$ norm of an n-by-m matrix A is defined as

$$\|A\|_1 = \sum_{\substack{1\leq i\leq n \\ 1\leq j\leq m}} |A_{ij}| \, . \tag{2.3}$$

2.1.2 Matrix Decompositions

Informally speaking, the matrix decomposition (or factorization) problem is, given a matrix A, find a collection of matrices $B_1, \ldots, B_N$, called factors, whose product provides a good approximation of A with respect to some cost function $cost(A, BW)$. Certain constraints can be imposed on the matrix being decomposed, as well as on the factors. In order to illustrate this we will give formal definitions of three very common matrix decomposition problems: Singular Value Decomposition, Nonnegative Matrix Factorization and Boolean Matrix Factorization.

For the definition of Singular Value Decomposition we need the notion of an orthogonal matrix (see e.g. [GVL12] for details).

Definition 2.1.6. A matrix $U \in \mathbb{R}^{n \times n}$ is called *orthogonal* if the following holds

$$UU^T = U^T U = I_n \,, \tag{2.4}$$

where I_n is an n-by-n identity matrix.

Theorem 2.1.1 ([GVL12])**.** If A is a real n-by-m matrix, then there exist orthogonal matrices $U \in \mathbb{R}^{n \times n}$ and $U \in \mathbb{R}^{m \times m}$ and a nonnegative rectangular diagonal matrix $\Sigma \in \mathbb{R}^{n \times m}$, such that

$$A = U \Sigma V^T \,. \tag{2.5}$$

The decomposition (2.5) is called the *Singular Value Decomposition* or (SVD).

When the original matrix and the factors are restricted to nonnegative real matrices, we obtain the Nonnegative Matrix Factorization problem (for more details see e.g.[LS99]).

Definition 2.1.7. Given a matrix $A \in \mathbb{R}_+^{n \times m}$ and a positive integer rank k, the *Nonnegative Matrix Factorization* (*NMF*)problem is to find two factors $B \in \mathbb{R}_+^{n \times k}$ and $W \in \mathbb{R}_+^{k \times m}$ such that the approximation error

$$cost(A, BW) \tag{2.6}$$

is minimized. The most common choice for the cost function is the Frobenius norm – $cost(A, BW) = \|A - BW\|_F$.

Boolean matrix factorization (see [Mie09]) considers binary matrices with arithmetic operations defined over so called Boolean algebra, which differs from the conventional arithmetics. The following definition of Boolean algebra is due to Whitesitt [Whi95].

Definition 2.1.8. A class of elements B together with two binary operations $(+)$ and $(\cdot)$ (where $a \cdot b$ will be written ab) is a *Boolean algebra* if and only if the following postulates hold:

1. The operations $(+)$ and $(\cdot)$ are commutative.
2. There exist in B distinct identity elements 0 and 1 relative to the operation $(+)$ and $(\cdot)$, respectively.
3. Each operation is distributive over the other.
4. For every $a \in B$ there exists an element $a' \in B$ such that

$$a + a' = 1 \quad \text{and} \quad aa' = 0.$$

In the following we will always assume that $B = \{0, 1\}$, in which case the binary operations $(+)$ and $(\cdot)$ are logical `OR` and `AND`, respectively.

Definition 2.1.9 ([Mie09])**.** Given a binary matrix $A \in \{0, 1\}^{n \times m}$ and a positive integer k, the *Boolean Matrix Factorization* (*BMF*) problem is to find binary matrices $B \in \{0, 1\}^{n \times k}$ and $W \in \{0, 1\}^{k \times m}$ such that

$$cost(A, B \square W) = \|A - B \square W\|_1 \tag{2.7}$$

is minimized.

2.1.3 Convex Analysis

We now introduce some basic notions from the theory of convex sets and functions that will be used in optimization routines described in Chapter 6. More details on the theory of convex sets and functions in general can be found, for example, in a classical book by Kelley [Kel75].

Definition 2.1.10. Let L be a linear space over $\mathbb{R}$. A set $X \subset L$ is called *convex* if for any two elements $x, y \in X$ and any number $0 \leq \alpha \leq 1$ we have $\alpha x + (1 - \alpha)y \in X$.

Definition 2.1.11. Let L be a linear space over $\mathbb{R}$ and $X \subset L$ be convex. A function $f \colon X \to \mathbb{R}$, is called *convex* if for any $x, y \in X$ and for any number $0 \leq \alpha \leq 1$ the following condition holds

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)y \,. \tag{2.8}$$

Convexity is a very useful property when it comes to optimization as for convex functions every local minimum is global [BV04].

A closely related notion is that of a biconvex set and a biconvex function (see e.g. [GPK07] and [AH86]). Informally speaking, a function of two arguments is biconvex if it is convex on each of the arguments when the other one is fixed. As we consider functions of two arguments it is useful to define the Cartesian product of sets (see [Hal60] for more).

Definition 2.1.12. The *Cartesian product* of two sets X and Y is the set of pairs (x, y) defined as follows

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\} . \tag{2.9}$$

When the objective is biconvex it is still possible to use convex optimization methods if we fix one of the variables, as then we obtain a convex function. In order to give a proper definition of a biconvex set let us first introduce some useful notation. Let U and V be linear spaces over $\mathbb{R}$ and sets $X \subset U$ and $Y \subset V$ be convex. For a set $B \subset U \times V$ and for any $x \in X$ and $y \in Y$ we define x-section and y-section of B as $B_x = \{y \in Y \mid (x, y) \in B\}$ and $B_y = \{x \in X \mid (x, y) \in B\}$ respectively.

Definition 2.1.13. Let U and V be linear spaces over $\mathbb{R}$, and let sets $X \subset U$ and $Y \subset V$ be convex. A set $B \subset X \times Y$ is called *biconvex* if for any $x \in X$ and $y \in Y$ the sets B_x and B_y are convex.

For any element $x \in X$ we denote by f_x a mapping from B_x to $\mathbb{R}$ such that $f_x(y) = f(x, y)$ for all $y \in B_x$, and the same notation applies to $f_y \colon B_y \to \mathbb{R}$.

Definition 2.1.14. Let U and V be linear spaces over $\mathbb{R}$, sets $X \subset U$ and $Y \subset V$ convex, and a set $B \in X \times Y$ biconvex. A function $f \colon B \to \mathbb{R}$ is called *biconvex* if for any $x \in X$ and $y \in Y$ the functions f_x and f_y are convex.

2.2 Notation

Most of the notation in the thesis is standard, except for the one used for the max-times algebra. This section gives a quick recap of the known notation, as well as introduces a new one for the max-times algebra.

2.2.1 Vectors and Matrices

Throughout this work matrices are denoted by capital letters and vectors are set with lowercase letters. Unless otherwise specified, BW denotes the normal product of matrices B and W, and Ax stands for the result of applying the linear operator defined by matrix A to vector v. We use standard notation for indexing vectors and matrices: for a vector v its i-th component is denoted by v_i and for a matrix A an element on the intersection of in its i-th row and j-th column is denoted by A_{ij}. However, we do introduce some more elaborated notation for complex indexing. Quite often we need to consider a contiguous range of elements in a vector. We use an expression $v(i\!:\!j)$ to denote the range in a vector v from position i to position j. Given a matrix X, the symbol $X(i)$ denotes the i-th column of X and the symbol $X(j,:)$ denotes its j-th row. We also use a vector of indices to extract elements from another vector. Namely, given a vector v and an index vector w of size n, the expression v_w stands for $v_{w_1}, v_{w_2}, \ldots, v_{w_n}$.

Sometimes, apart from standard matrix multiplication, it is convenient to consider their elementwise product. More formally, given n-by-m matrices B and W their elementwise or *Hadamard* product (see e.g. [GVL12]) is denoted by $B \circ W$ and is defined as

$$(B \circ W)_{ij} = B_{ij} W_{ij} \,.$$

2.2.2 Max-Times Algebra

Max-times algebra is defined over the set of nonnegative real numbers and has two operations, which we call addition and multiplication. Multiplication is the same as in the conventional algebra of real numbers, but addition is defined as taking the maximum instead. More formally

Definition 2.2.1. *Max-times algebra* is a triple $(\mathbb{R}_+, \boxplus, \boxtimes)$, where $\mathbb{R}_+ = \{x \in \mathbb{R} \mid\ x \geq 0\}$ and for any $x, y \geq 0$, $x \boxplus y = \max\{x, y\}$ and $x \boxtimes y = xy$.

Since taking a product of matrices requires only operations of addition and multiplication we can now define max-times matrix multiplication.

Definition 2.2.2. Given matrices $B \in \mathbb{R}_+^{n \times k}$ and $W \in \mathbb{R}_+^{k \times m}$, their *max-times product* is defined as follows

$$(B \boxtimes W)_{ij} = \max_{1 \leq s \leq k} B_{is} W_{sj} \,. \tag{2.10}$$

The analogy between the conventional algebra of real numbers and the max-times algebra can be further extended by formulating the problem of max-times matrix factorization.

Given a matrix $A \in \mathbb{R}_+^{n\times m}$ we want to find factors B and W whose product gives the best approximation of A with respect to some cost function. In this work we mostly use Frobenius norm, and the corresponding decomposition problem is as follows.

Definition 2.2.3. Given a matrix $A \in \mathbb{R}_+^{n\times m}$ and an integer $k > 0$, the *Max Matrix Factorization* problem (*MMF*) is to find matrices $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ that minimize the error

$$cost(A, B \boxtimes W) = \|A - B \boxtimes W\|_F \,. \tag{2.11}$$

For the sake of brevity in the future we will call this problem simply *max-times matrix factorization.*

2.2.3 Max-Plus Algebra

Very closely related to the max-times algebra is another unconventional algebra with of real numbers, where addition defined as taking the maximum and multiplication replaced by addition and. Prior to giving a formal definition we will introduce a concept of the extended real number line.

Definition 2.2.4. The extended real number line, which we denote by $\overline{\mathbb{R}}$ is defined as $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$. In addition the following rules apply to the infinity element:

$$\exp\{-\infty\} = 0 \tag{2.12}$$

and

$$\log(0) = -\infty \,. \tag{2.13}$$

Definition 2.2.5. Max-plus algebra is a triple $(\overline{\mathbb{R}}, \oplus, \otimes)$, where for any $x, y \geq 0$, $x \oplus y = \max\{x, y\}$ and $x \otimes y = x + y$.

It is well known that the max-times and max-plus algebras are isomorphic [BGT00]. This can readily be checked by considering the exponent function, which maps $\overline{\mathbb{R}}$ to $\mathbb{R}_+$ and preserves the operations. This isomorphism can be extended to matrices by simply mapping matrix elements to their exponents:

Definition 2.2.6. For any positive integers n and m define a map $\pi : \overline{\mathbb{R}}^{n\times k} \to \mathbb{R}_+^{n\times k}$ such that for any $A \in \overline{\mathbb{R}}^{n\times k}$, $\pi(A)_{ij} = \exp A_{ij}$.

This indicates that we can obtain a max-times decomposition by finding the corresponding solution over max-plus algebra. This link will be exploited in the algorithmic part of the thesis. We will also prove that there is a linear relation between approximation errors in max-plus and max-times matrix algebras.

Chapter 3

Related Work

In this chapter we review previous work in areas related to the topic of the thesis. Whereas matrix factorization can be viewed as the reverse of matrix multiplication, the latter can be defined in many different ways. Beside the standard matrix multiplication over real numbers, there is also Boolean matrix multiplication, which defines the product of two binary matrices using Boolean algebra, max-times matrix multiplication, which is the topic of this thesis, and many other.

3.1 Real-Valued Matrix Decompositions

A classic example of a real-valued matrix decomposition is the singular value decomposition (SVD) [GVL12], which is very well known and finds extensive applications in many disciplines. The SVD of a real n-by-m matrix A is a factorization of the form $A = U\Sigma V^T$, where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{n\times m}$ is a rectangular diagonal matrix with nonnegative entries. An important property of SVD is that it provides the best low-rank approximation of a given matrix with respect to the Frobenius norm [GVL12]. More formally, given an n-by-m matrix A and an integer $k > 0$, the matrix obtained by truncating the SVD of A to k greatest values of Σ minimizes the error $\|A - B\|_F$ among all rank-k n-by-m matrices B.

The above property is often used in practice to separate the more important parts of data from the noise. In [JY11] the truncated SVD was used to remove the noise from sensor data in electronic nose systems. Another prominent usage of the truncated SVD is in dimensionality reduction (see for example [SKKR00, DDF$^+$90]).

Despite SVD being so ubiquitous there are some restrictions to its usage in data mining due to possible presence of negative elements in the factors. In many applications

negative values are hard to interpret, and thus other methods have to be used [Mie09]. Nonnegative matrix factorization (NMF) is a way to tackle this problem. For a given nonnegative real matrix A the NMF problem is to find a decomposition of A into two matrices $A \approx BW$ such that B and W are also nonnegative. The NMF problem differs from MMF in that normal matrix multiplication is used, compared to max-times multiplication in MMF. However, when factors are sparse, the two can be quite close. This becomes apparent if we write the definition of max-times matrix multiplication (see Section 2.2.2) $(B \boxtimes W)_{ij} = \max_{1 \le s \le k} B_{is} W_{sj}$ and observe that the sparser the factors are, the more likely it is to have only one nonzero element inside the maximization, in which case $(B \boxtimes W)_{ij} = (BW)_{ij}$. Applications of NMF include text mining [PSBP04], document clustering [XLG03], pattern discovery [BTGM04] and many other. This area drew considerable attention after a publication by Lee and Seung [LS99], where they provided an efficient algorithm for solving NMF problem. It is worth mentioning that even though Lee and Seung's paper, which was published in 1999, is the most famous in NMF literature, it was not the first one to consider the problem. Earlier work on NMF problem include a paper by Paatero and Tapper published in 1994 [PT94] (see also [Paa97] and [Paa99]) and a 1993 paper by Cohen and Rothblum [CR93]. An overview of NMF algorithms and applications can be found in [BBL+07].

Though both NMF and SVD perform approximations of a fixed rank, there are also other ways to enforce compact representation of data. For example in maximum-margin matrix factorization constraints are posed on the norms of factors. This approach was exploited in Srebro et. al. [SRJ04], where they show that it is a good method for predicting unobserved values in a matrix. The authors also indicate that posing constraints on the factor norms, rather than on the rank yields a convex optimization problem, which is easier to solve.

Semi-discrete decomposition (SDD) was introduced by O'Leary and Peleg in their 1983 paper [OP83], where it was used for image compression. It decomposes a real matrix into a sum of rank-1 matrices $A \approx \sum_{k=1}^{K} d_k x_k y_k^T$, where d_k are real numbers and the vectors x_k and y_k can only take values from the set $\{-1, 0, 1\}$. This decomposition can be represented in a form similar to SVD: $A \approx U \Sigma V$, with Σ being diagonal and matrices U and V having values restricted to $\{-1, 0, 1\}$. SDD has subsequently found applications in other areas, for example it was used for latent semantic indexing in information retrieval [KO00]. In its original form SDD is not very similar to MMF since it allows negative values in the factors and also uses standard matrix multiplication. However, if in the definition we change the summation to maximization, that is seek for a decomposition of the form $A \approx \max_k d_k x_k y_k^T$, and restrict values of x_k and y_k to

$\{0, 1\}$ and d_k to $\mathbb{R}_+$, then we obtain an MMF version of it. This latter approach is used for two algorithms that we propose in Section 5.2.

3.2 Binary Matrix Decompositions

Binary matrix factorization occurs when the matrix being decomposed has only binary values, but its type can vary depending on the constraints we pose on the factors. A special case is Boolean matrix factorization (BMF) that deals with matrices over Boolean algebra (for the definition see Section 2.1). There is a strong link between max-times matrix factorization and BMF. One can observe that Boolean matrix multiplication is nothing else than max-times multiplication on binary matrices. Furthermore, for every precise max-times decomposition of a matrix, there is a corresponding Boolean decomposition of its pattern (see Lemma 4.1.1). In many applications data is naturally represented as a binary matrix (e.g. transaction databases), which makes it reasonable to seek decompositions that preserve the type of the data. The conceptual and algorithmic analysis of the problem was done in [Mie09], which focuses mainly on data mining perspective of the problem and highlights the interpretability of the obtained decompositions, rather than necessarily having a small reconstruction error. For a linear algebra perspective see [Kim82], where the emphasis is put on the existence of exact decompositions.

It is not always the case that the factors are required to be binary as well. For example in logistic PCA [SSU03] real matrices are used to model probability distributions of binary matrices. More formally, a real matrix $\Theta \in \mathbb{R}^{n \times m}$ defines a probability distribution on binary data as follows $P(X|\Theta) = \prod_{ij} \sigma(\Theta_{ij})^{X_{ij}}$, where $\sigma(\Theta_{ij}) = 1/(1 + e^{-\Theta_{ij}})$ is the logistic function. Like in standard PCA, the idea is to find a more compact representation of the matrix Θ, which is done by finding two matrices B and W of lower inner dimension, such that $\Theta = U\,V$. In a sense logistic PCA is somewhere in between BMF and MMF, as it deals with binary data, but the decomposition is performed in the domain of real numbers.

Another variation of binary matrix factorization is the problem of finding two binary factors such that their normal matrix product approximates the original matrix, which is also binary [ZDLZ07]. As MMF this can be considered a mixture of BMF and NMF: it uses the matrix type of BMF and the product of NMF, whereas MMF does the opposite. This type of binary matrix factorization finds applications in for example document clustering and gene analysis [ZDLZ07].

A closely related problem to BMF is tiling transaction databases [GGM04]. A tile is a submatrix consisting only of ones. Given a binary matrix (transaction database), the

goal is to cover as many ones as possible with tiles, without covering any zeros. The problem can be viewed as Boolean matrix factorization where covering 0 s with 1 s is not allowed [Mie09]. This in turn implies that it is also a restricted variation of the MMF problem, where all matrices are binary and overcovering the data is not allowed.

The co-clustering problem is, given a matrix, split the indices of both dimensions into groups, such that elements within each group have some similarity. Co-clustering was introduced in 1972 by Hardigan [Har72]. For newer research in the area see for example [BDG+04] and [Dhi01]. Boolean matrix factorization is essentially a co-clustering problem, where the clusters are allowed to overlap [Mie09]. As with tiling, the co-clustering problem is related to MMF through the Boolean factorization. Namely, if MMF is restricted to binary matrices and no overlap is required, then we obtain the co-clustering problem.

Boolean algebra is, in fact, a subalgebra of the max-times algebra. Hence, one can view the max-times matrix decomposition problem as a mixture of NMF and BMF. As with NMF we allow only nonnegative elements in factors, but the product is defined in a way that essentially copies Boolean product. Moreover, if we consider max-times product on binary matrices we will obtain exactly Boolean matrix multiplication.

3.3 Max-Plus Algebra

Max-plus algebra [Hog06] is known to be isomorphic to max-times algebra [BGT00]. It differs from the conventional algebra of real numbers in that addition is defined as maximization and multiplication as addition. Despite the theory of max-plus algebra being relatively young it has been thoroughly studied in recent years. The reason for this is an explosion of interest in so called discrete event systems (DES) [CL99], where max-plus algebra has become ubiquitously used for modeling (see e.g. [BCOQ92] and [CGQ99]). In order to get some insight on what a discrete event system is and how max-plus algebra helps in modeling it, let us have a look at the following example [BCOQ92]. Consider a set of machines repeatedly running a bunch of jobs, where each job is assigned to a specific machine. Assume that in order to start running its task each machine needs outputs from some of the others. We can model the whole process as a directed weighted graph $G = (V, E, W)$, where nodes represent machines and each weight w_{ij} stands for the time that should elapse after node i has started to process its job before node j can start to work. If machine i does not depend on the output of machine j, then we set $w_{ij} = -\infty$. For each machine i we would like to know when it can activate for the k-th time at the earliest. We denote by x^k a vector that represents the earliest activation times for the machines at iteration k, with x^0 standing for the initial activation times. Then we can describe the system by the following recurrence

relation

$$x_i^{k+1} = \max_j \{w_{ij} + x_j^k\} , \tag{3.1}$$

or in vector notation

$$x_i^{k+1} = A \otimes x^k , \tag{3.2}$$

which is a max-plus linear equation. Solving linear systems of equations is an important routine used in many matrix factorization algorithms. It was shown in [But03] that for max-plus algebra this problem is equivalent to the set cover problem, which is known to be NP-hard. This result directly affects the max-times algebra through the above mentioned isomorphism [BGT00] and makes the problem of precisely solving max-times linear systems infeasible for high dimensions. However, we avoid this pitfall as algorithms considered in this work usually do not require this problem to be solved precisely. Instead, we are aiming for a good approximation of the best solution.

To the best of our knowledge there is no published work on matrix decompositions in the settings considered in this thesis. However, there are proven results for a slightly different algebra which extends the max-plus algebra. Namely, Schutter and Moor [SM02] demonstrated that if the max-times algebra is extended in such a way that there is an additive inverse for each element, then it is possible to solve many of the standard matrix decomposition problems. In particular, they obtained max-plus analogues of QR and SVD. Authors also claim that techniques they propose can readily be extended to other types of classic factorizations (e.g. Hessenberg and LU decomposition).

Chapter 4

Matrix Factorization over Max-Times Algebra

Here we prove some theoretical results concerning the MMF problem. These include a connection between max-times and Boolean ranks, as well as computational complexity and relationship to max-plus algebra.

4.1 Rank

Definition 4.1.1. Define a map $\phi\colon \mathbb{R}_+^{n\times m} \to \{0,1\}^{n\times m}$ as follows: $\phi(A)_{ij} = 1$ if $A_{ij} > 0$ and 0 else. For every matrix A its image $\phi(A)$ is called *pattern* of A.

Lemma 4.1.1. The map ϕ preserves multiplication, that is for any $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ the following holds

$$\phi(B)\,\square\,\phi(W) = \phi(B \boxtimes W)\ . \tag{4.1}$$

Proof. Denote by A the max-product of B and W. Let $B' = \phi(B)$, $W' = \phi(W)$ and $A' = \phi(A)$. We need to show that for all $1 \le i \le n,\ 1 \le j \le m,\ A'_{ij} = (B' \square W')_{ij}$. Consider an element A_{ij} for some $1 \le i \le n,\ 1 \le j \le m$. If $A_{ij} > 0$ then $B_{is}W_{sj} > 0$ for at least one $1 \le s \le k$, which entails $(B' \square W')_{ij} = \max\limits_k B'_{ik}W'_{kj} \ge B'_{is}W'_{sj} = 1$. Similarly, from $A_{ij} = 0$ it follows that $B_{is}W_{sj} = 0$ for all $1 \le s \le k$, and hence $(B' \square W')_{ij} = \max\limits_k B'_{is}W'_{sj} = 0$. By (4.1) $A'_{ij} = 1$ if and only if $A_{ij} > 0$, which completes the proof. □

It immediately follows from Lemma 4.1.1 that the max-times rank of a matrix is not smaller than the Boolean rank of its pattern. The converse however does not hold as can easily be verified by considering a matrix that has no zeros. A pattern of such a matrix has the Boolean rank 1 (it is the Boolean product of two vectors of all ones), but the max-times rank can obviously be higher than 1.

4.2 Sparsity of Factors

When doing matrix factorization for data mining purposes we usually desire for sparse factors, which make the patterns more prominent and easier to find. We would like to know how the factor density depends on the density of the original matrix. Two important measures of sparsity are the number of nonzero elements in a matrix (see e.g. [Mie10]) and the fraction of zero elements in a matrix (see for example a work by by Gillis and Glineur [GG10]). In this section we will show that even though the above paper by Gillis and Glineur only considers NMF, the results can be readily transferred to MMF. The fraction of zeros of an n-by-m matrix A is defined as follows:

$$s(A) = \frac{nm - \eta(A)}{nm} , \tag{4.2}$$

where for convenience we denoted by $\eta(A)$ the number of nonzero elements in a matrix. In their article Gillis and Glineur used the notion of one matrix being dominated by another, which they called "underapproximation".

Definition 4.2.1. An n-by-m matrix A is said to be *dominated* by an n-by-m matrix B if $A_{ij} \leq B_{ij}$ for all $1 \leq i \leq n$, and $1 \leq j \leq m$.

They proved that for any two matrices $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$, if their product is dominated by an n-by-m matrix A, then we have $s(B) + s(W) \geq s(A)$. In fact, the same result holds for dominated max-times products, which we will show by adjusting the proofs of Theorem 1 and Corollary 1 of [GG10].

Theorem 4.2.1. Let matrices $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ be such that their max-times product is dominated by an n-by-m matrix A. Then the following estimate holds

$$s(B) + s(W) \geq s(A) . \tag{4.3}$$

Proof. We first prove (4.3) for $k = 1$. Let $b \in \mathbb{R}_+^n$ and $w \in \mathbb{R}_+^m$ be such that $b_i w_j^T \leq A_{ij}$ for all $1 \leq i \leq n$, $1 \leq j \leq m$. Since $(bw^T)_{ij} > 0$ if and only if $b_i > 0$ and $w_j > 0$ we

have

$$\eta(bw^T) = \eta(b)\,\eta(w)\ . \tag{4.4}$$

By (4.2) we have $\eta(bw^T) = nm(1-s(bw^T))$, $\eta(b) = n(1-s(b))$ and $\eta(w) = m(1-s(w))$. Plugging these expressions into (4.4) we obtain $(1 - s(bw^T)) = (1 - s(b))(1 - s(w))$. Hence, the number of zeros in a rank-1 dominated approximation of A is

$$s(b) + s(w) \geq s(bw^T)\ . \tag{4.5}$$

From (4.5) and the fact that the number of nonzero elements in bw^T is no greater than in A, it follows that

$$s(b) + s(w) \geq s(A)\ . \tag{4.6}$$

Now let $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ be such that $B \boxtimes W$ is dominated by A. Then $B_{il}W_{lj} \leq A_{ij}$ for all $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq l \leq k$, which means that for each $1 \leq l \leq k$, $B(l)W(l,:)$ is dominated by A. To complete the proof observe that $s(B) = 1/k \sum_{1\leq l\leq k} B(l)$ and $s(W) = 1/k \sum_{1\leq l\leq k} W(l)$, and that for each l the estimate (4.6) holds. □

4.3 Complexity

In this section we prove that MMF is computationally hard, and thus the use of approximate methods is justified. Before formulating the main results we introduce some basic concepts from complexity theory. We start by defining the notion of a decision problem. Informally, a decision problem is a computational problem that can only have "YES" or "NO" answers.

Definition 4.3.1 ([GJ79])**.** A *decision problem* Π consists of a set of instances I_Π and a subset $Y_\Pi \subseteq I_\Pi$ of YES-instances.

The following definition of the complexity class NP is informal, but it suffices for our analysis.

Definition 4.3.2 ([Sch07])**.** *NP* is a class of all problems for which polynomial-time verification algorithms exist.

Definition 4.3.3 ([CLRS03])**.** A problem is called *NP-hard* if it is at least as hard as the hardest problem in NP.

Definition 4.3.4 ([Sch07])**.** An NP-hard problem Π is called *NP-complete* if it is also in NP.

In order to prove the complexity results for MMF, we first need to turn it into a decision problem. Recall from Section 2.2.2 that in MMF problem we perform optimization with respect to some cost function $cost(A, B \boxtimes W)$.

Definition 4.3.5. Given a matrix $A \in \mathbb{R}_+^{n\times m}$, a positive integer k and a real number $t \geq 0$, the *d-MMF* problem is to answer whether there exist factor matrices $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ such that

$$cost(A, B \boxtimes W) \leq t\ . \tag{4.7}$$

The d-MMF problem is NP-hard, which is quite expected since max-times algebra is an extension of Boolean algebra, and d-BMF is NP-hard [Mie09].

Theorem 4.3.1. d-MMF problem is NP-complete.

Proof. Assume for the sake of contradiction that there exists an algorithm Ω that can solve the decision version of the MMF problem in polynomial time. We will prove the statement of the theorem by reduction from the BMF problem, which is known to be in NP [Mie09]. Consider an instance (A, k) of the BMF problem that has a precise decomposition. Since A is binary, any of its exact BMF decompositions is also an exact MMF decomposition. Hence, there exists an exact MMF factorization $A = B \boxtimes W$. Let us define binarizations B' and W' of the factors B and W with $B'_{ij} = 1$ iff $B_{ij} > 0$ and $W'_{ij} = 1$ iff $W_{ij} > 0$. It is now straightforward to see that $B' \square W' = A$, since for any $1 \leq i \leq n$, $1 \leq j \leq m$, from $\max_k B_{ik}W_{kj} = 0$ it follows that $B'_{is}W'_{sj} = 0$ for all $1 \leq s \leq k$, and from $\max_k B_{ik}W_{kj} > 0$ it follows that $B'_{is}W'_{sj} = 1$ for at least one $1 \leq s \leq k$. Thus, knowing an MMF decomposition of a binary matrix A, we can obtain its BMF decomposition in polynomial time. It follows that the d-MMF problem is at least as hard as the d-BMF problem. Checking the error of the obtained solution is clearly polynomial in the size of A and k, which finishes the proof. □

Theorem 4.3.2. The problem of approximating MMF to within a polynomially computable factor is NP-hard.

Proof. This theorem is analogous to Theorem 4.2 in [Mie09] and can be proved a similar way. Let us assume that the opposite holds, that is there exists an algorithm Ω and a polynomial function $f\colon \mathbb{R}_+ \to \mathbb{R}_+$ such that for any instance I of the MMF problem we have $cost(I, \Omega(I)) \,/\, cost^*(I) \leq f(|I|)$, where $cost^*(I)$ is the optimal cost for instance I. Now consider an instance $I = (A, k)$ admitting an exact decomposition, which means that $cost^*(I) = 0$. The polynomial bound above can only hold if $cost(I, \Omega(I)) = 0$, which produces a contradiction since the exact MMF problem is NP-hard by Theorem 4.3.1. □

4.4 Relation to Max-Plus Algebra

Max-plus and max-times algebras are isomorphic, which means that solving the decomposition problem with zero error in one of them leads to zero error in another. However, in the context of data mining we are interested in minimizing an error, rather than obtaining an exact decomposition. This is due to the fact that precise decomposition is computationally hard and does not always exist, and we resort to finding approximate solutions. Moreover, real-world data is usually noisy, which makes the precise factorization problem practically meaningless as we will be effectively modeling the noise. What we want is to be able to bound an error over max-times algebra given the corresponding error over max-plus algebra. The following theorem states that this dependence is in fact linear with respect to the max-plus error.

At the time of writing this thesis we do not have algorithms that exploit the link between max-times and max-plus algebras, however we plan to tackle it in future research. The following theorem, which shows that the max-times approximation error is linearly bounded by the corresponding error for the max-plus algebra, provides the basis for bridging the two factorization problems.

Theorem 4.4.1. Let $A \in \overline{\mathbb{R}}^{n\times m}$, $B \in \overline{\mathbb{R}}^{n\times k}$ and $W \in \overline{\mathbb{R}}^{k\times m}$. Denote

$$M = \exp\left\{\max_{\substack{1\le i\le n\\ 1\le j\le m}}\left\{\max\left\{A_{ij}, \max_{1\le d\le k}\{B_{id}+W_{dj}\}\right\}\right\}\right\}.$$

If an error can be bounded in max-plus algebra

$$\|A - B\otimes W\|_F^2 \le \lambda\,, \tag{4.8}$$

then the following estimate holds with respect to the max-times algebra

$$\|\pi(A) - \pi(B)\boxtimes\pi(W)\|_F^2 \le M^2\lambda\,. \tag{4.9}$$

Proof. Denote $\alpha_{ij} = \max\limits_{1\le d\le k}\{B_{id}+W_{dj}\}$. From (4.8) it follows that there exists a set of numbers $\{\lambda_{ij}\ge 0 \mid\ 1\le i\le n,\ 1\le j\le m\}$, s.t. for any i,j we have $(A_{ij}-\alpha_{ij})^2 \le \lambda_{ij}$ and $\sum\limits_{ij}\lambda_{ij} = \lambda$. By mean-value theorem for every $i,\ j$ we obtain

$$|\exp A_{ij} - \exp\alpha_{ij}| = |A_{ij}-\alpha_{ij}|\,\exp\alpha_{ij}^* \le \sqrt{\lambda_{ij}}\,\exp\alpha_{ij}^*\,,$$

for some $\min\{A_{ij}, \alpha_{ij}\} \leq \alpha^*_{ij} \leq \max\{A_{ij}, \alpha_{ij}\}$. Hence,

$$(\exp A_{ij} - \exp \alpha_{ij})^2 \leq \lambda_{ij} (\exp \max\{A_{ij}, \alpha_{ij}\})^2 \ .$$

The estimate for the max-times error now follows from the monotonicity of the exponent

$$\|\pi(A) - \pi(B) \boxtimes \pi(W)\|_F^2 \leq \sum_{ij} \left(\exp \alpha^*_{ij}\right)^2 \lambda_{ij} \leq \sum_{ij} \left(\exp \max\{A_{ij}, \alpha_{ij}\}\right)^2 \lambda_{ij} \leq M^2 \lambda \ .$$

□

Even though the constant hidden in this estimate is exponential this problem can be easily overcome by scaling.

Chapter 5

Direct Methods

In this chapter we propose methods that try to solve the original max-times matrix decomposition problem, as opposed to a relaxation approach described in Chapter 6. We start by describing a greedy method, which incorporates all the existing constraints and attempts to find the best solution that does not violate any of them. Next, a group of algorithms based on semidiscrete decomposition (SDD) [KO00, OP83] is presented.

5.1 Greedy solver

`Greedy` solver uses the alternating updates heuristics to reduce the matrix factorization problem to solving a system of linear equations. Assume as usual that we are given a matrix $A \in \mathbb{R}_+^{n\times m}$ and a rank k, and that the problem is to find factors $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$ such that they minimize $cost(A, B \boxtimes W)$. The idea of alternating updates is to initialize one factor, say B, then fix it and try to optimize $cost(A, B \boxtimes W)$ with respect to W. Then the roles of B and W are changed, that is W is fixed and $cost(A, B \boxtimes W)$ is optimized with respect to B. This approach is well known and is frequently used with problems having biconvex or approximately biconvex structure (an example of its application to nonnegative matrix factorization can be found in [PT94]).

After applying the above heuristics what is left is to solve the inner problem, which takes the form of a system of max-linear equation. In order to see that let us assume that the objective is to minimize the error with respect to Frobenius norm, and that we have fixed factor B. The optimization problem is then

$$\min_{W\in\mathbb{R}_+^{k\times m}} \|A - B \boxtimes W\|_F = \min_{W\in\mathbb{R}_+^{k\times m}} \sum_{ij} (A_{ij} - \max_s B_{is}W_{sj}) \,. \tag{5.1}$$

Algorithm 5.1 `Greedy Solver`

Input: matrix $A \in \mathbb{R}_+^{n\times m}$, integer $k \leq \min\{n, m\}$
Output: Matrices $B \in \mathbb{R}_+^{n\times k}$, $W \in \mathbb{R}_+^{m\times k}$
Initialize: $0 < \epsilon \ll 1$, $B \in \mathbb{R}_+^{n\times k}$ random
$B_{ij} \leftarrow B_{ij} + \epsilon$, $1 \leq i \leq n$, $1 \leq j \leq k$ ▷ To allow division by B_{ij}
repeat
  for $j = 1$ *to* m **do**
    for $s = 1$ *to* k **do**
      $W_{sj} \leftarrow \min_i A_{ij}/B_{is}$ ▷ See (5.4)
    end for
  end for
  for $i = 1$ *to* n **do**
    for $s = 1$ *to* k **do**
      $B_{is} \leftarrow \min_j A_{ij}/W_{sj}$
    end for
  end for
until convergence conditions are satisfied
return B, W

We can split (5.1) into m independent optimization problems.

$$\min_{w\in\mathbb{R}_+^k} \|A(j) - B \boxtimes w\|_2 = \min_{w\in\mathbb{R}_+^k} \sum_{i=1}^{n} (A_{ij} - \max_s B_{is} w_s), \quad j = 1, \ldots, m \,, \tag{5.2}$$

where the symbol $A(j)$ stands for j-th column of A (see Section 2.2.1).

Each problem in (5.2) is of the form $B \boxtimes w \approx A(j)$, which is a max-linear equation. The `Greedy` solver tries to solve it by finding a vector w such that $B \boxtimes w \leq A(j)$. It does so by explicitly forcing all the constraints imposed on w, thus the name. For convenience let us denote the j-th column of A by a. For all $1 \leq s \leq k$ such that $B_{is} \neq 0$, a feasible w must satisfy

$$w_s \leq a_i/B_{is} \,. \tag{5.3}$$

By adding a small number to all elements of B we can write (5.3) for all s. The `Greedy` solver tries to approximate the solution by choosing an elementwise maximal w that satisfies all the conditions in (5.3), which is done by setting

$$w_s = \min_i a_i/B_{is} \,, \quad s = 1, \ldots, k \,. \tag{5.4}$$

Combining (5.4) with alternating updates we obtain the `Greedy` algorithm (Algorithm 5.1).

5.2 SDD Based Algorithms

The standard SDD algorithm decomposes a real-valued matrix $A \in \mathbb{R}^{n\times m}$ into a sum

$$A \approx \sum_{k=1}^{K} d_k x_k y_k^T, \tag{5.5}$$

where d_k are real numbers, and $x_k \in \{-1,\, 0,\, 1\}^n$ and $y_k \in \{-1,\, 0,\, 1\}^m$, [KO00, OP83]. This sum can be viewed as a matrix decomposition – in order to see this introduce factor matrices B and W with k-th column of B being equal to x_k and k-th row of W being equal to $d_k\, y_k^T$. The main idea behind the algorithm is that on each iteration a rank-1 approximation of a matrix of the form $d_k x_k y_k^T$ is found. Due to the constraints imposed on x_k and y_k, each summand of (5.2) is an n-by-m matrix with a flat bump determined by nonzero entries of x_k and y_k. Because of their shape these bumps are called *blocks*. We then subtract it from the original matrix, obtaining what is called a residual matrix, and then reiterate the process, this time applying the same procedure to the residual. After K iterations we have a rank-K approximation of the original matrix in the form (5.5).

In this section we present two algorithms based on the above approach. The first one, called SDDMMF (SDD from the name of the parent algorithm, and MMF refers to max-matrix factorization), is an alteration of the classic SDD algorithm – it uses the same objective on each iteration, but updates the residual differently. The second algorithm, which we call SDDUNDERFIT due to its behavior, differs from SDDMMF in that it never overcovers the data and uses $\|\cdot\|_1$ instead of Frobenius matrix norm in its objective.

5.2.1 SDDMMF

What inspires using SDD approach for max-times matrices is the fact that, despite having defined matrix multiplication differently, the expression $x_k\, y_k^T$, where x_k and y_k are column vectors, produces the same matrix for both standard and max-times multiplication operations. Thus, as in original SDD, the aim is to minimize the deviation of the recovered rank-1 matrix $d_k\, x_k\, y_k^T$, from the residual R_k, or to solve the problem

$$\min_{d\geq 0,\ x\in\{0,1\}^n,\ y\in\{0,1\}^m} J(x, y, d, R)\ ,$$

where

$$J(x, y, d, R) = \sum_{i=1}^{n}\sum_{j=1}^{m} (R_{ij} - d\, x_i\, y_j)^2\ . \tag{5.6}$$

However, that there are some important characteristics of max-times matrix multiplication that affect this optimization problem. The most obvious difference is that all entries of both factors and the original matrix must be nonnegative, which is why we do not allow negative elements in x and y, and also restrict d to nonnegative numbers.

Another peculiarity of max-times algebra is that there is no additive inverse, i.e. for $a \neq 0$ there is no element b, such that $\max\{a, b\} = 0$. This poses some serious difficulties as it is unclear how to define the residual matrix. We address this issue by maintaining a binary matrix χ that has ones only where the corresponding elements of the original matrix have been covered. We say that an element A_{ij} of the original matrix is covered if the corresponding element in the current recovered matrix is greater or equal to it, that is there exists k, $1 \leq k \leq K$, such that $d_k(x_k)_i(y_k)_j \geq A_{ij}$. The residual R is now defined as follows

$$R_{ij} = \begin{cases} A_{ij}, & \chi_{ij} = 0 \\ 0, & \text{otherwise} , \end{cases}$$

which means that all elements covered on the previous iterations turn to 0.

The main difficulty of max-times matrix decomposition, which is not present in case of standard decompositions, is that there is no "going back", in the sense that once we have some value for an element of the recovered matrix, it can never be decreased. This happens due to addition in normal matrix multiplication being substituted by maximization. To see this observe that after K iterations we have the following approximation of element A_{ij} : $A_{ij} \approx \max\limits_{k=1,\ldots,K} d_k(x_k)_i(y_k)_j = \nu$. Then, after performing one more step it becomes $A_{ij} \approx \max\{\nu, d_{K+1}(x_{K+1})_i(y_{K+1})_j\}$, and the nondecreasing property follows by induction on K. This means that once the algorithm has made a mistake by choosing a bigger value than in the original matrix, it can never correct it.

We propose a number of heuristics to battle this problem. The first one is to change blocks found on previous iterations, once a new block is found that partially covers them. We update each block in the as follows: if new blocks have covered all elements in the original matrix corresponding to some row (column) of the block, this row (column) is removed from it. This way blocks can shrink if some of their parts become redundant. Heights are then recomputed using the same algorithm as in the original SDD algorithm (see the references above). Block updates are done in the routine named UPDATEBLOCKS on each iteration of SDDMMF algorithm. It is possible that at some point a block might lose all its rows (columns), and thus become redundant. The second heuristics is that we drop such blocks entirely and decrease by one the number of blocks found so far, so that one additional block can be found.

In Algorithm 5.2 the procedure used to solve the optimization problems on lines 8 and 10, as well as the initialization of y on line 6 is the same as in [KO00] and [OP83] .

Algorithm 5.2 SDDMMF

Input: matrix $A \in \mathbb{R}_+^{n\times m}$, integer $K \leq \min\{n, m\}$, $\epsilon > 0$
Output: Matrices $X \in \mathbb{R}_+^{n\times k}$, $Y \in \mathbb{R}_+^{m\times k}$ and vector $d \in \mathbb{R}_+^k$
Initialize: $k \leftarrow 1$, $NITER > 1$, $R \leftarrow A$, $\chi \leftarrow$ zero $K \times n \times m$ matrix
while $k \leq K$ **do** ▷ Solves optimization problem 5.6
  for $count = 1$ **to** $NITER$ **do**
    initialize y
    if $count \mod 2 = 1$ **then**
      $[x, d] \leftarrow \underset{d\geq 0,\ x\in\{0,1\}^n}{\arg\min} J(x, y, d, R)$
    else
      $[y, d] \leftarrow \underset{d\geq 0,\ y\in\{0,1\}^m}{\arg\min} J(x, y, d, R)$
    end if
  end for
  k-th column of $X \leftarrow x$, k-th column of $Y \leftarrow y$, $d_k \leftarrow d$ ▷ Save results
  $[X, Y] \leftarrow UPDATEBLOCKS(A, X, Y, \chi)$
  $S \leftarrow \max_{i=1,\ldots,k} d_i X(i) Y(i)^T$
  $\kappa \leftarrow$ binary n-by-m matrix, s.t $\kappa_{ij} = 1$ iff $S_{ij} > A_{ij} - \epsilon$
  $\chi_k \leftarrow \kappa$ ▷ Save κ from k-th iteration
  $R_{ij} \leftarrow A_{ij}$ if $\kappa_{ij} = 0$, else $R_{ij} \leftarrow 0$ ▷ Update residual
  $count \leftarrow 0$
  for $l = k - 1$ **downto** 1 **do**
    if $d_l\, X(l)\, Y(l)^T = 0$ **then**
      remove l-th block
      $count \leftarrow count + 1$
    end if
  end for
  $k \leftarrow k - count + 1$ ▷ If blocks have been removed decrease k accordingly
end while
return X, Y, d

5.2.2 SDDUNDERFIT

The above presented heuristics give considerable improvement for some sorts of data, but the problem of overcovering still persists. The next algorithm of the SDD family, which we called SDDUNDERFIT, avoids this issue altogether by choosing new blocks in such a way that the recovered matrix is always elementwise smaller than the original. Assume we want to find a rank-1 approximation of the matrix $A \in \mathbb{R}_+^{n\times m}$ without covering any of its elements with a greater value. Since all norms on $\mathbb{R}^n$ are equivalent (see Lemma 2.1.1), we can replace the Frobenius norm, which was originally used in SDD, with $\|\cdot\|_1$. The optimization problem is then to find such $x \in \{0, 1\}^n$, $y \in \{0, 1\}^m$ and $d \in \mathbb{R}_+$ that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $dx_i y_j \leq A_{ij}$ and the error with respect to $\|\cdot\|_1$ is minimized. More formally, we need to solve the following mixed integer-linear problem

Algorithm 5.3 UPDATEBLOCKS

Input: matrices $A \in \mathbb{R}_+^{n\times m}$, $X \in \mathbb{R}_+^{n\times k}$, $Y \in \mathbb{R}_+^{m\times k}$, $\chi \in \{0,1\}^{K\times n\times m}$
Output: updated matrices $X \in \mathbb{R}_+^{n\times k}$ and $Y \in \mathbb{R}_+^{m\times k}$
$k \leftarrow$ number of columns of X
for $i = 1$ **to** $k-1$ **do**
  if $d_i < d_k$ **then**
    $X(i) \leftarrow X(i) \circ (\mathbf{1} - X(k))$
    $Y(i) \leftarrow Y(i) \circ (\mathbf{1} - Y(k))$ ▷ Remove entries covered by the last block
    $Z \leftarrow$ binary matrix with $Z_{rs} = 1$
      iff A_{rs} is not covered by $d_k X(k)_r Y(k)_s^T$ and $(\chi_i)_{rs} = 0$
    $d_i = \sum_{Z_{rs}=1} X(i)_r\, Y(i)_s\, A_{rs} \Big/ \sum_{Z_{rs}=1} 1$ ▷ Update heights since some of elements previously covered by i-th block are now covered by k-th block, causing redundancy
  end if
end for
return X, Y

$$\min_{\substack{dx_iy_j\le A_{ij}\\ d\ge 0,\ x\in\{0,1\}^n,\ y\in\{0,1\}^m}} r(x,y,d,A) = \min_{\substack{dx_iy_j\le A_{ij}\\ d\ge 0,\ x\in\{0,1\}^n,\ y\in\{0,1\}^m}} \sum_{i=1}^{n}\sum_{j=1}^{m} |A_{ij} - dx_iy_j| \,. \quad (5.7)$$

Let us fix y and solve the problem with respect to x and d. Since we are not allowed to overcover, problem (5.7) is equivalent to the following.

$$\max_{\substack{dx_iy_j\le A_{ij}\\ d\ge 0,\ x\in\{0,1\}^n}} \sum_{i=1}^{n}\sum_{j=1}^{m} dx_iy_j \sim \max_{\substack{dx_iy_j\le A_{ij}\\ d\ge 0,\ x\in\{0,1\}^n}} d\sum_{i=1}^{n} x_i \,.$$

The last equivalence is due to the fact that y is fixed. We now proceed in the same way as in the standard SDD - fix the number of nonzero elements in x and then find the optimal x and d. Assuming $\sum_{1\le i\le n} x_i = J$, we obtain

$$\max_{\substack{dx_i\le f_i\\ d\ge 0,\ \sum_i x_i = J}} d \,, \quad (5.8)$$

where $x \in \{0,1\}^n$ and $f_i = \min_{y_j\neq 0} A_{ij}/y_j$. We now have to go through the n possible values of J to find the one for which problem (5.8) reaches its optimum. Finally, the solution to (5.8) is given by $d = \min_{x_i\neq 0} f_i$ and $x \in \{0,1\}^n$ is chosen such that ones correspond to indices of J largest elements in f.

Algorithm 5.4 `SDDUNDERFIT`

Input: matrix $A \in \mathbb{R}_+^{n\times m}$, integer $K \leq \min\{n, m\}$
Output: Matrices $X \in \mathbb{R}_+^{n\times k}$, $Y \in \mathbb{R}_+^{m\times k}$ and vector $d \in \mathbb{R}_+^k$
Initialize: $R \leftarrow A$, $\chi \leftarrow$ zero $n \times m$ matrix, $\epsilon > 0$
for $k = 1$ **to** K **do**
    for $count = 1$ **to** $NITER$ **do**
        initialize y
        if $count \mod 2 = 1$ **then**
            $[x, d] \leftarrow UNDERFITSOLVE(y, A)$
        else
            $[y, d] \leftarrow UNDERFITSOLVE(x, A)$
        end if
    end for
    k-th column of $X \leftarrow x$, k-th column of $Y \leftarrow y$, $d_k \leftarrow d$ ▷ Save results
    $S \leftarrow \max_{i=1,\ldots,k} d_i X(i) Y(i)^T$
    $\chi \leftarrow$ binary matrix, s. t $\chi_{ij} = 1$ iff $S_{ij} > A_{ij} - \epsilon$
    $R_{ij} \leftarrow A_{ij}$ if $\chi_{ij} = 0$, else $R_{ij} \leftarrow 0$ ▷ Update residual
end for
return X, Y, d

Algorithm 5.5 `UNDERFITSOLVE`

Input: *vector* $y \in \{0, 1\}^m$, *matrix* $A \in \mathbb{R}^{n\times m}$
Output: vector $x \in \{0, 1\}^n$, $d \geq 0$
$f \leftarrow$ *vector from* $\mathbb{R}^n$ *with* $f_i = \min_{y_j \neq 0} A_{ij}/y_j$
$order \leftarrow$ *increasing order of elements in* f
$best \leftarrow f_{order(1)}$
$xbest \leftarrow$ *zero vector* $\in \mathbb{R}^n$
$xbest_{order(1)} \leftarrow 1$
$dbest \leftarrow \max_i f_i$
for $J = 2$ *to* n **do**
    $d \leftarrow \min f_{order(1:J)}$
    $result \leftarrow dJ$
    if $result > best$ **then**
        $best \leftarrow result$
        $xbest \leftarrow$ *zero vector* $\in \mathbb{R}^n$
        $xbest_{order(1:J)} \leftarrow 1$
        $dbest \leftarrow d$
    end if
end for
return $xbest$, $dbest$

Chapter 6

Relaxation Methods

One of the main difficulties of max-times matrix factorization problem is the nondifferentiability of the maximization function, as it prevents us from using effective methods of continuous optimization, e.g. gradient descent. In this chapter we present two methods, `MERA` and `BMERA`, that exploit the exponential relaxation technique, which allows to obtain a smooth objective.

6.1 MERA

The Max Exponential Relaxation Algorithm (`MERA`) is based on two heuristics: alternating updates of the factors (see Section 5.1), which reduces the MMF problem to finding only one factor, and the exponential relaxation technique, that makes the problem smooth. Recall from Section 5.1 that the inner problem when using alternating updates is

$$\min_{w \in \mathbb{R}_+^k} \|A(j) - B \boxtimes w\|_2 = \min_{w \in \mathbb{R}_+^k} \sum_{i=1}^{n} (A_{ij} - \max_s B_{is} w_s), \quad j = 1, \ldots, m \,. \tag{6.1}$$

Here the symbol $A(j)$ denotes j-th column of matrix A (see Section 2.2.1). In order to solve each of the problems in (6.1), we would like to use a convex optimization algorithm, e.g. gradient descent or Newton's method [BV04]. However, that would require us to compute the gradient (or in in case of the Newton's method Hessian) of the objective, and the max function is not differentiable. The exponential relaxation is a technique that allows to relax the maximization function in such way that it becomes infinitely differentiable. For that let us first introduce a function

$$p_\sigma(x_1, \ldots, x_k;\ x_s) = \frac{e^{\sigma x_s}}{\sum_{i=1}^{k} e^{\sigma x_i}} \,. \tag{6.2}$$

We will see in the proof of Lemma 6.1.1 that it satisfies

$$p_\sigma(x_1, \ldots, x_k;\ x_s) \to \begin{cases} l, & x_s = \max\limits_{1 \le q \le k} x_q \\ 0, & \text{otherwise} \end{cases}$$

as $\sigma \to \infty$, where l is the number of maximal elements in vector x. This property will be used to mimic the behavior of the max function.

Definition 6.1.1. A σ-max product of matrices $B \in \mathbb{R}_+^{n \times k}$ and $W \in \mathbb{R}_+^{k \times m}$ is defined as follows

$$(B \boxtimes_\sigma W)_{ij} = \sum_{s=1}^{k} p_\sigma(B_{i,:} \circ W_{:,j};\ B_{is}W_{sj}) B_{is}W_{sj} \ ,$$

where $B_{i,:} \circ W_{:,j}$ stands for the elementwise product of the i-th row of B and the j-th column of W.

Lemma 6.1.1. We have that $\boxtimes_\sigma \to \boxtimes$ as $\sigma \to \infty$ in the sense that for any $B \in \mathbb{R}_+^{n \times k}$ and $W \in \mathbb{R}_+^{k \times m}$, $\|B \boxtimes W - B \boxtimes_\sigma W\|_F \to 0$.

Proof. We need to show that for any $B \in \mathbb{R}_+^{n \times k}$, $W \in \mathbb{R}_+^{k \times m}$, $1 \le i \le n$ and $1 \le j \le m$

$$(B \boxtimes W)_{ij} - (B \boxtimes_\sigma W)_{ij} \to 0$$

as $\sigma \to \infty$. Assume that there are l maximal elements in the vector $B_{i,:} \circ W_{:,j}$. Then

$$1/p_\sigma(B_{i1}W_{1j}, \ldots, B_{ik}W_{kj};\ B_{is}W_{sj}) = \sum_{t=1}^{k} e^{\sigma B_{it}W_{tj}} / e^{\sigma B_{is}W_{sj}}$$

$$= \sum_{i=1}^{k} e^{\sigma(B_{it}W_{tj} - B_{is}W_{sj})} \xrightarrow{\sigma \to \infty} \begin{cases} l, & B_{is}W_{sj} = \max\limits_{1 \le q \le k} B_{iq}W_{qj} \\ \infty, & \text{otherwise} \ . \end{cases}$$

Thus,

$$\begin{aligned} (B \boxtimes_\sigma W)_{ij} &= \sum_{s=1}^{k} p_\sigma(B_{i,:} \circ W_{:,j};\ B_{is}W_{sj}) B_{is}W_{sj} \xrightarrow{\sigma \to \infty} \sum_{s=1}^{l} 1/l \max_{1 \le q \le k} B_{iq}W_{qj} \\ &= \max_{1 \le q \le k} B_{iq}W_{qj} \ . \end{aligned}$$

□

Lemma 6.1.1 guarantees that we can approximate the max-product with any precision as long as σ is sufficiently large.

Algorithm 6.1 MERA

1: **Input:** matrix $A \in \mathbb{R}_+^{n\times m}$, integer $k \leq \min\{n, m\}$
2: **Output:** Factors $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$
3: **function** MERA(k, A)
4: **Initialize:** $B \in \mathbb{R}_+^{n\times k}$, $W \in \mathbb{R}_+^{k\times m}$ random, s.t. $\|B\|_\infty \|W\|_\infty = \|A\|_\infty$
5: $MAXITER > 1$, $\sigma > 0$, $\tau > 1$, $error \leftarrow \|A - B \boxtimes W\|_F$, $\epsilon > 0$, $\delta > 0$, $count \leftarrow 1$
6: **while** $error > \epsilon$ && $count \leq MAXITER$ **do**
7: $inner\_error \leftarrow \|A - B \boxtimes_\sigma W\|_F$
8: **repeat**
9: **if** $count \mod 2 = 1$ **then**
10: $W \leftarrow \arg\min_{U\in\mathbb{R}_+^{k\times m}} \|A - B \boxtimes_\sigma U\|_F$
11: **else**
12: $B \leftarrow \arg\min_{U\in\mathbb{R}_+^{n\times k}} \|A - U \boxtimes_\sigma W\|_F$
13: **end if**
14: $inner\_error\_prev \leftarrow inner\_error$
15: $inner\_error \leftarrow \|A - B \boxtimes_\sigma W\|_F$
16: **until** $inner\_error\_prev - inner\_error < \delta$
17: $error \leftarrow \|A - B \boxtimes W\|_F$, $count \leftarrow count + 1$, $\sigma \leftarrow \tau * \sigma$
18: **end while**
19: **return** B, W
20: **end function**

The MERA algorithm combines the relaxation of the objective described above with the alternating updates heuristics. The controlling parameter σ is increased after each iteration, which ensures that the relaxed product converges to the original max-times matrix product.

The optimization problem on line 10 of Algorithm 6.1 can be divided into m independent parts, each consisting of finding one column u of U. For each $1 \leq j \leq m$ we need to optimize the objective

$$f_j(u) = \sum_{i=1}^{n} \left(a_i - \sum_{s=1}^{k} p_\sigma(B_{i,:} \circ u;\ B_{is}u_s) B_{is}\, u_s \right)^2 \tag{6.3}$$

$$\sim \sum_{i=1}^{n} \left(\left(\sum_{s=1}^{k} p_\sigma(B_{i,:} \circ u;\ B_{is}u_s) B_{is}\, u_s \right)^2 - 2a_i \sum_{s=1}^{k} p_\sigma(B_{i,:} \circ u;\ B_{is}u_s) B_{is}\, u_s \right) ,$$

where a denotes the j-th column of A. The two objectives in (6.3) yield the same optimization problem since they only differ by a constant term. In our implementation of the MERA algorithm the gradient descent was used to optimize the objective (6.3). The algorithm performs relatively well on dense matrices, but we still find only a local minimum since the the objective function is not convex [FP96]. A way to address the problem of nonconvexity is to represent the objective as a difference of convex functions

[Har59]. Currently we have a preliminary implementation of this approach, but it has not yet improved on the original `MERA` algorithm, and thus we do not describe it here. It is worth mentioning that if a function is representable as a difference of convex functions, then there are infinitely many such representations, and choosing the right one is far from trivial. We now obtain explicit form of the gradient of the objective.

Denote $\phi = B_{i,:} \circ u,\ v = \exp\{\sigma\phi\}$. We can represent the objective as a sum of independent components

$$f_j(u) = \sum_{i=1}^{n} \psi_i(v),$$

$$\psi_i(u) = \frac{\sum\limits_{s=1}^{k}\sum\limits_{d=1}^{k} v_s v_d \phi_s \phi_d}{\sum\limits_{\alpha=1}^{k}\sum\limits_{\beta=1}^{k} v_\alpha v_\beta} - 2a_i \frac{\sum\limits_{s=1}^{k} v_s \phi_s}{\sum\limits_{s=1}^{k} v_s} = g/h - 2p/q\ .$$

We further have

$$\frac{\partial g}{\partial v_s} = 2\phi_s\langle v, \phi\rangle, \quad \frac{\partial g}{\partial \phi_s} = 2v_s\langle v, \phi\rangle, \quad \frac{\partial h}{\partial v_s} = 2\langle \mathtt{1}, v\rangle, \quad \frac{d\phi_j}{du_j} = B_{ij}, \quad \frac{dv_j}{du_j} = \sigma B_{ij} v_j\ ,$$

and for the derivatives of p and q :

$$\frac{\partial p}{\partial v_s} = \phi_s, \quad \frac{\partial p}{\partial \phi_s} = v_s, \quad \frac{\partial q}{\partial v_s} = 1\ .$$

Plugging in these expressions we obtain

$$\begin{aligned}
\frac{\partial g}{\partial u_s} &= \frac{\partial g}{\partial v_s}\frac{\partial v_s}{\partial u_s} + \frac{\partial g}{\partial \phi_s}\frac{\partial \phi}{\partial u_s} = 2B_{is}\langle v, \phi\rangle v_s(\sigma\phi_s + 1)\ , \\
\frac{\partial h}{\partial u_s} &= \frac{\partial h}{\partial v_s}\frac{dv_s}{du_s} = 2B_{is}\sigma\langle v, \mathtt{1}\rangle v_s\ , \\
\frac{\partial p}{\partial u_s} &= \frac{\partial p}{\partial v_s}\frac{dv_s}{du_s} + \frac{\partial p}{\partial \phi_s}\frac{d\phi}{du_s} = B_{is}v_s(\sigma\phi_s + 1)\ , \\
\frac{\partial q}{\partial u_s} &= \sigma B_{is} v_s\ .
\end{aligned}$$

Hence

$$\frac{\partial\psi(u)}{\partial u_s} = \frac{2B_{is}\langle v,\phi\rangle v_s(\sigma\phi_s+1)}{\left(\sum_{s=1}^{k} v_s\right)^2} - \frac{2B_{is}\langle v,\mathbf{1}\rangle\sigma v_s\left(\sum_{s=1}^{k} v_s\phi_s\right)^2}{\left(\sum_{s=1}^{k} v_s\right)^4} - 2a_i\frac{B_{is}\langle v,\mathbf{1}\rangle v_s(\sigma\phi_s+1) - B_{is}\langle v,\phi\rangle\sigma v_s}{\left(\sum_{s=1}^{k} v_s\right)^2}.$$

The computational complexity of the `MERA` algorithm is $O(nmk)$. In order to see this observe that the main work is done when evaluating the objective of the inner problem and finding its gradient, both of which require $O(nk)$ time. The precision of the gradient descent used to solve the inner problem is not critical and it converges in $O(1)$ iterations. Thus, the complexity of the inner problem is $O(nk)$. It is invoked m times for each iteration of the algorithm (we separately find all columns of W). We only need $O(1)$ iterations of the algorithm, which means that the overall time complexity is $O(nmk)$.

While effective for dense data, this method fails on sparse matrices since it tends to fill zero entries with positive values. A better way of handling sparse data is presented in the next section.

6.2 BMERA

As was noted above, the `MERA` algorithm is in general not suitable for sparse matrices. Here we consider its modification, which we call `BMERA`, that combines the exponential relaxation approach with Boolean matrix factorization. The idea is to construct a binary matrix from the original one by using thresholding and then run a BMF algorithm (in our implementation we use `iterX` algorithms initialized by `Asso`, both of which were proposed in [Mie09]). This yields the basis for the `BMERA` method – we first preprocess the data with BMF, and allow an element in a factor to be nonzero only if the corresponding binary entry was 1. This allows to turn a sparse matrix into a smaller more dense matrix, which is more suitable for the exponential relaxation procedure. First on line 7 of Algorithm 6.2 we run the above mentioned BMF algorithms in order to determine which elements in the factors can be nonxzero. The remaining part of the algorithm is very similar to `MERA` except for the way the inner problem is solved. As in case of `MERA` each column is optimized independently (line 5), and for each column j we distinguish three cases: all elements in j-th column of $mask$ are zero, there is only one nonzero

Algorithm 6.2 BMERA

1: **Input:** matrix $A \in \mathbb{R}_+^{n\times m}$, integer $k \leq \min\{n, m\}$
2: **Output:** Factors $B \in \mathbb{R}_+^{n\times k}$ and $W \in \mathbb{R}_+^{k\times m}$
3: **function** `BMERA`(k, A)
4: **Initialize:** $B \in \mathbb{R}_+^{n\times k}$, $W \in \mathbb{R}_+^{k\times m}$ random, s.t. $\|B\|_\infty\|W\|_\infty = \|A\|_\infty$
5: $MAXITER > 1$, $\kappa > 0$, $error \leftarrow \|A - B \boxtimes W\|_F$, $\epsilon > 0$, $count \leftarrow 1$
6: $D \leftarrow$ binary matrix with $D_{ij} = \begin{cases} 1, & A_{ij} > \kappa \\ 0, & \text{otherwise} \end{cases}$
7: $[mask1, mask2] = BMF(D, k)$
8: $error \leftarrow \|A - B \boxtimes W\|_F$
9: **repeat**
10: **if** $count \mod 2 = 1$ **then**
11: $W \leftarrow InnerSolve(A, B, mask2)$
12: **else**
13: $B \leftarrow InnerSolve(A, W, mask1)$
14: **end if**
15: $error \leftarrow \|A - B \boxtimes W\|_F$
16: $count \leftarrow count + 1$
17: $error\_prev \leftarrow error$
18: **until** $error\_prev - error < \epsilon \;||\; count \geq MAXITER$
19: **return** B, W
20: **end function**

element and there is more than one nonzero element. If all elements in $mask(j)$ are zero, then we simply set $W(j)$ to be a zero column. If there is a single nonzero element, which say has index i, then the inner problem can be solved optimally for column j. Indeed, we only need to find one element of the column, the rest of it will be filled with zeros. Assume that the index of this element is d. Since V in this case is a vector, the optimization problem becomes

$$W_{dj} = \arg\min_{y\in\mathbb{R}_+} \|A(j) - yV\|_2^2 = \arg\min_{y\in\mathbb{R}_+} J(y) \,. \tag{6.4}$$

We have $\partial J(y)/\partial y = 2y\langle V, V\rangle - 2\langle A(j), V\rangle$. Since at the minimum point we should have $\partial J(y)/\partial y = 0$, which means that $y = \langle A(j), V(i)\rangle/\langle V(i), V(i)\rangle$. The problem (6.4) is obviously convex, and hence y is its unique minimum. Also since $A(j)$, $V \in \mathbb{R}_+^n$ we have $y \geq 0$, which means that the vector $W(j)$ we have found is feasible. In case there is more than one nonzero element in $mask(j)$ we use the exponential relaxation in a way similar to Algorithm 6.1, but without alternating updates (lines 15-21) of `InnerSolve`.

The asymptotic computational complexity of `BMERA` $O(nmk)$, same as that of `MERA`, which can be checked analogously. Except for extreme cases, when the data is very dense or has a highly pronounced block structure (see Chapter 5), `BMERA` is an algorithm of choice for max-times matrix factorization.

Algorithm 6.3 InnerSolve

Input: matrices $A \in \mathbb{R}_+^{n \times m}$, $B \in \mathbb{R}_+^{n \times k}$ and $mask \in \{0, 1\}^{k \times m}$
Output: Matrix $W \in \mathbb{R}_+^{k \times m}$
function InnerSolve(A, B, $mask$)
    Initialize: $\tau > 0$
    for $j = 1$ **to** m **do**
        $idx \leftarrow \{s \mid mask(s, j) = 1\}$
        $V \leftarrow B(idx)$ ▷ Select columns of B whose indices are in idx.
        if $|idx| = 0$ **then**
            $W(j) \leftarrow zeros^{k \times 1}$
        end if
        if $|idx| = 1$ **then**
            $W_{ij} \leftarrow \begin{cases} \langle A(j), V(i) \rangle / \langle V(i), V(i) \rangle, & idx_i = 1 \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \ldots, n$
        end if
        if $|idx| > 1$ **then**
            Initialize: $\sigma > 0$, $count \leftarrow 1$, $w \in \mathbb{R}_+^k$ random
            $error \leftarrow \|A(j) - V \boxtimes w\|_2$
            while $error \geq \epsilon$ && $count < MAXITER$ **do**
                $w \leftarrow \arg\min_{u \in \mathbb{R}_+^k} \|A(j) - V \boxtimes_\sigma u\|_2$ ▷ Solve this using the gradient descent initialized by w.
                $count \leftarrow count + 1$
                $error \leftarrow \|A(j) - V \boxtimes w\|_2$
                $\sigma \leftarrow \tau * \sigma$
            end while
        end if
    end for
    return W
end function

Chapter 7

Experiments

7.1 Synthetic Experiments

We ran the proposed methods and compared them to benchmark algorithms on a number of synthetic test setups. These include varying inner dimension test, where we observe how algorithms behave when the rank of the approximation changes, varying factor density test, that is designed to reveal the dependence between the density and the quality of the approximation, and varying noise test, which demonstrates how noise tolerant the algorithms are.

7.1.1 Algorithms Used

A number of algorithms have been introduced for max-times matrix factorization. They can be divided into two categories: those that solve the original problem directly, called direct methods, and those that relax the objective, called relaxation methods. The direct methods are `SDDMMF` (Section 5.2.1) and `SDDUNDERFIT` (Section 5.2.2), which are both based on the semi-discrete decomposition (`SDD`) algorithm [OP83]. We also proposed two relaxation methods: `MERA` (Section 6.1) and its extension - `BMERA` (Section 6.2), which use exponential relaxation of the objective. The difference is that `BMERA` uses a combination of BMF algorithms `Asso` and `iterX` (see [Mie09]) to determine which elements of the product to ignore. The `MERA` algorithm is not suitable for sparse data since it makes many small errors and tends to fill the matrix with small numbers where there should be zeros. In general, the denser the data, the better results can be obtained with the `MERA` algorithm, which is demonstrated by the tests with the varying factor density. Note that in the varying inner dimension and noise tests the factor density was set to 0.25, which is too low for the `MERA` algorithm and results in very high errors. To the best of our

knowledge there are no published algorithms for the MMF problem. However, for the tests described in this chapter we chose `NMF` [KP08] for this purpose, as it appears to be the closest fit for solving the MMF problem.

We did not include any experiments with the `Greedy` algorithm (Section 5.1) since its performance is very poor, unless matrices are extremely small. It is thus more of theoretical interest than an actual solving algorithm, as it demonstrates the most naive approach to the problem.

7.1.2 Comparing Different Methods

It is always a challenge to make a fair comparison of methods that solve different problems. The `NMF` algorithm [KP08] used here for the comparison with the proposed methods assumes that matrix multiplication is defined in a normal way, which makes it drastically different from all the MMF algorithms. So why to use it? The most obvious reason is that NMF uses the same matrix type as MMF, but there is also a more subtle similarity. Recall that the max-product of two matrices B and W is defined as follows: $(B \boxtimes W)_{ij} = \max_k B_{ik} W_{kj}$. Observe that if the factors are sparse, then many of the terms over which maximization is performed may be zero, and that when there is only one nonzero element, or no such elements at all, the above formula is equivalent to taking the sum over k, which is the usual matrix multiplication. Thus, the sparser the factors are, the closer their max-product is to the standard matrix product.

We conducted two sets of experiments with max-times and standard matrix products used for multiplying the factors produced by `NMF` [KP08]. Even though `NMF` is solving a problem with respect to the usual matrix multiplication, using the max-times product for reconstruction is justified from the point of view of comparing the effectiveness of `NMF` to that of the proposed algorithms in solving the MMF problem. From the data mining perspective, however, we are more interested in patterns contained in the data, thus posing the problem of what information we can extract using `NMF` from the MMF data. This motivates the second experimental setup where normal matrix multiplication was used for reconstruction with `NMF`.

Another important question one has to ask when performing experimental evaluation of algorithms is how to measure the error they make. It is very common in matrix factorization community to use the Frobenius or $\|\cdot\|_1$ norms for this purpose. The Frobenius norm is very natural for the continuous problems like NMF since it basically measures the Euclidean distance between the two matrices represented as vectors of their values. However, it has a drawback of treating small and big errors differently. Namely, due to squaring of the element-wise errors, it increases the impact of errors

more than 1 whereas it diminishes the contribution of errors below 1 [Mie09]. In our experiments we used both Frobenius and $\|\cdot\|_1$ norms. More precisely, we used relative errors with respect to these norms: $cost_F(A, BW) = \|A - B \boxtimes W\|_F / \|A\|_F$ and $cost_1(A, BW) = \|A - B \boxtimes W\|_1 / \|A\|_1$, which scale the error by the norm of the original matrix. Here AB denotes whichever matrix product is used. The Frobenius norm seems slightly more natural for both the MMF and NMF algorithms since they use it as an objective (all except `SDDUNDERFIT`), but $\|\cdot\|_1$ does not have the above mentioned problem with small errors.

7.1.3 Experimental Setup and Results

Experiments with synthetic data were conducted in order to evaluate the proposed algorithms and compare them to each other and the `NMF` algorithm [KP08]. Since NMF is actually solving a different problem we present its results when both standard and max-times matrix multiplications are used. For all tests with real data we plot the results of NMF with max-times multiplication on Figures `(a)` and `(b)` and with standard multiplication on Figures `(c)` and `(d)`. The error is measured with respect to Frobenius norm (Figures `(a)` and `(c)`) and $\|\cdot\|_1$-error (Figures `(b)` and `(d)`). In all tests factors with dimensions 400-by-10 and 10-by-200 were generated and then multiplied to obtain a 400-by-200 matrix. Factors were obtained in the following way: first matrices of the corresponding dimensions were drawn from a uniform distribution on the interval (0, 1), then some randomly chosen elements were turned to 0, so that the resulting matrices would have a required density. Depending on the type of the test, noise of some level might have been added to the product. The goal was to reconstruct the original matrix from the factors obtained by factorization algorithms, and the resulting errors were measured with respect to the Frobenius and $\|\cdot\|_1$ norms. The reported results are the means over 10 instances for each experimental setup, with error bars on the plots having length of twice the standard deviation.

Inner Dimension. This experimental setup is designed to reveal how well we can approximate the original matrix using a smaller inner dimension than the one used for data generation, and how fast the error decreases as it grows. The inner dimension (the second dimension of the first factor and the first dimension of the second factor) varied from 2 to 10 with steps of size 2. The factor density was fixed to 0.25, and the noise level was 10% in all experiments. It is visible on Figure 7.1 that when the inner dimension k of the factors is close to that used for data generation, `BMERA` provides the best approximation with respect to both Frobenius and $\|\cdot\|_1$ norms, even when the normal matrix multiplication is used for `NMF`. On the other hand `NMF` is comparable to or even better than `BMERA` with respect to the Frobenius error when k is small (Figures

(a) and (c)). This can be explained by the fact that for very small inner dimensions (as well as low factor densities) max-product becomes close to the standard matrix product. It is also worth mentioning that NMF performs considerably better when standard matrix multiplication is used, which is expected, but BMERA is still more effective when k is close to the actual inner dimension. When $\|\cdot\|_1$ is used BMERA is better than all the rest of the algorithms for all k. The MERA algorithm demonstrates poor results in these settings as the factor density of 0.25 used in the experiments is too low for it. Two SDD-based algorithms SDDMMF and SDDUNDERFIT do not show any meaningful results as they are only useful for block matrices.

Density of factors. The density of the factors used for data generation varied from 0.1 to 0.75. The inner dimension was always 10, which is the same as the one used for data generation, and the level of noise was fixed to 10%. Figure 7.2 demonstrates that the effectiveness of the MERA algorithm is highly dependent on the factor density. The correlation is positive, that is the results become better as the density increases. Both BMERA and NMF (with max-times matrix multiplication) produce worse results as the factor density grows, however NMF seems to be much more sensitive to it. On the other hand NMF combined with the normal matrix multiplication can outperform BMERA for very dense data. The SDDMMF algorithm, although unusable on general data with moderate densities, obtains somewhat better results when the density is very high.

Noise. The noise level varied from 0% to 40%. The inner dimension was again set to 10 and the factor density was 0.25. Both BMERA and NMF algorithms show a similar degree of noise tolerance (Figure 7.3). However, BMERA demonstrates better results for all noise levels and both error measures. The rest of the algorithms produce very high errors almost irrespective of the noise.

7.2 Real-World Data

7.2.1 The Extended Yale Face Database B

The facial images database[1] was used in this experiment [GBK01]. The original data consists of images of 28 human subjects under 9 poses and 64 lighting conditions, overall 16128 pictures. In our experimental setup we used the cropped images [LHK05], which were further resized to 32-by-32 format by Xiaofei He, et al. ([CHH+07b], [CHH07a], [CHHZ06], [HYH+05])[2]. We selected 127 images from the database and ran the MERA and SDDUNDERFIT algorithms on it. The results were compared to those of the NMF

[1] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
[2] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

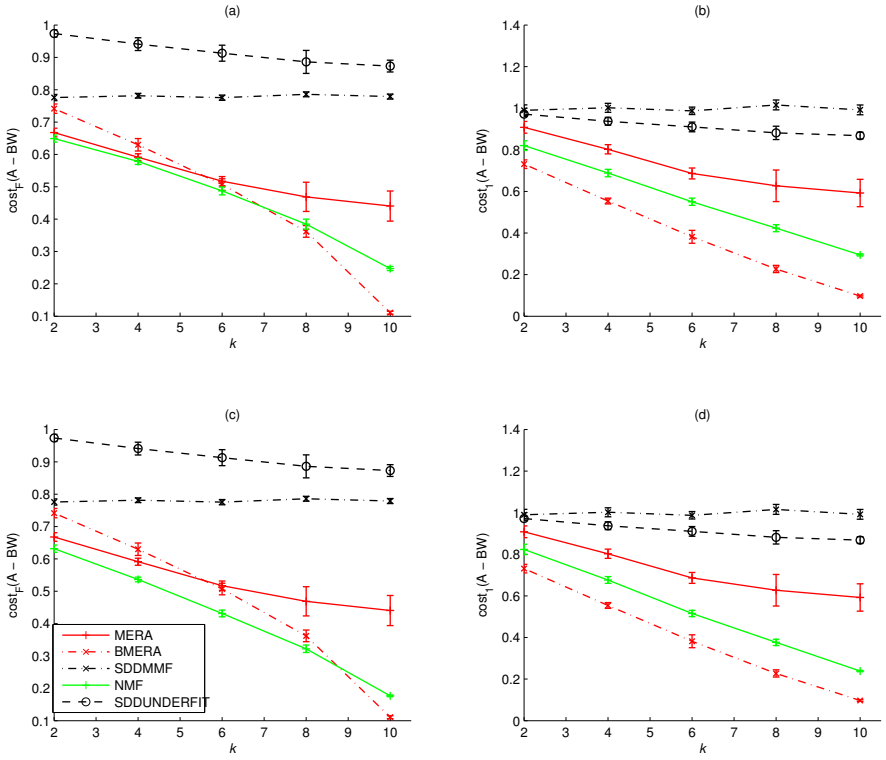

Figure 7.1: Reconstruction error with respect to the Frobenius error (left) and $\|\cdot\|_1$-error (right) when the inner dimension k varies from 2 to 10. Max-times multiplication is used for all max-times methods. For `NMF` multiplication is with respect to max-times algebra on plots `(a)` and `(b)` and standard on plots `(c)` and `(d)`. The level of noise is 10% and the factor density is fixed to 0.25. Markers are mean values over 10 instances, and the error bars have height of twice the standard deviation.

algorithm [KP08], which is known to be a good choice for face recognition and feature extraction from faces ([LS99], [GV02]). The data was represented as a 1024-by-127 matrix, where each column is a reshaped 32-by-32 image. This matrix was decomposed into two factors with inner dimension $k = 15$, and the resulting recovered images and features were studied. Since each image is represented as a column, the recovered faces are linear combinations of the columns in the first factor (features) for `NMF` and max-linear combinations for MERA and `SDDUNDERFIT`. The images obtained after multiplying the factors are shown on Figure 7.4, and the extracted features (columns of the first factor matrix) are demonstrated on Figure 7.5. `NMF` demonstrates better results, which appears to be due to the fact that it "adds" features, which gives it an upper hand over MMF-based methods that always take elementwise maximum. In order to get some intuition as to why this happens, consider the problem of finding the best approximation of an image with a linear combination of features, where both, the image and the features are represented as vectors of the same size. Assume that the original image has a big spike, and in order to obtain a good approximation we need to fit it. For normal matrix multiplication the approach would be to select those features that have a spike at the same place as the original data, and add them after multiplying with appropriate

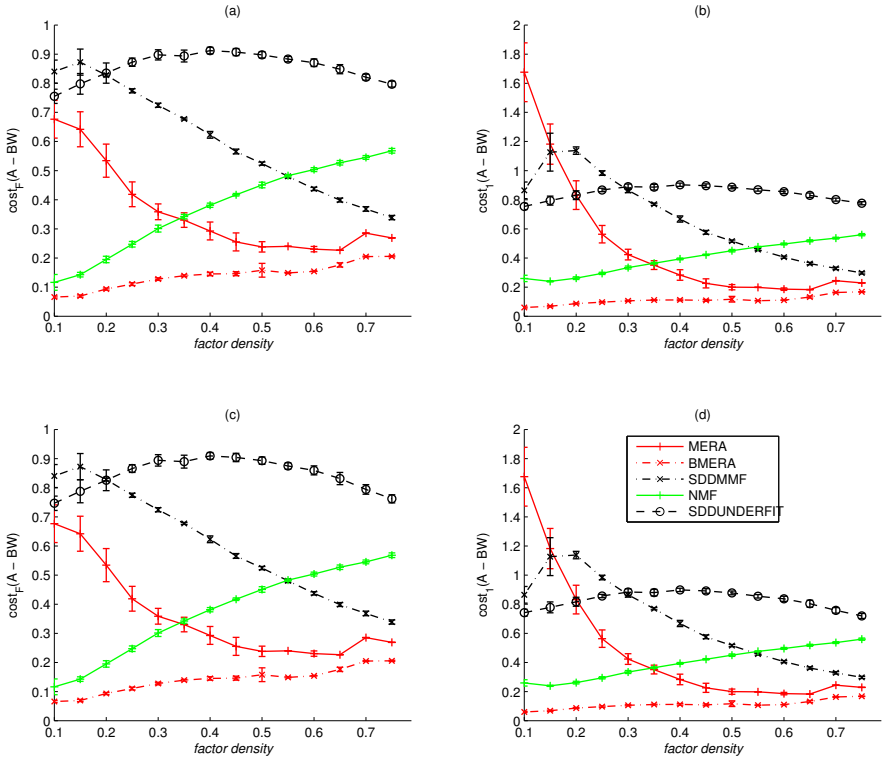

Figure 7.2: Reconstruction error with respect to the Frobenius error (left) and $\|\cdot\|_1$-error (right) when the factor density varies from 0.1 to 0.75. Max-times multiplication is used for all max-times methods. For `NMF` multiplication is with respect to max-times algebra on plots `(a)` and `(b)` and standard on plots `(c)` and `(d)`. Inner dimension is $k = 10$ and the level of noise is 10%. Markers are mean values over 10 instances, and the error bars have height of twice the standard deviation.

coefficients. Since the selected vectors and the original image have a spike at the same place, this spike becomes exaggerated compared to others, since they are unlikely to be present in all of the selected vectors. However, in case of max-linear combination of the features, it is harder to obtain a high spike without overcovering the data in other places. The reason for this is that the spike will be represented by one vector, rather than a sum of several vectors, and hence all the elements of this vector will be included with the same coefficient. Faces recovered by the `MERA` algorithm appear to be similar to those obtained by the `NMF` but with a very hilgh level of noise, which is visible as black dots. When comparing two MMF-based methods `MERA` and `SDDUNDERFIT`, it is remarkable how drastically different their results are. `MERA` is better in terms of the data recovery since the image data is not well approximated by blocks, which greatly restricts `SDDUNDERFIT`. However, the features produced by `MERA` can hardly be recognized as anything meaningful, whereas `SDDUNDERFIT` extracts some sketchy pictures, which represent the most prominent parts of faces. This can be explained by the fact that `SDDUNDERFIT` looks for the block structure in the data, that is tries to approximate the original matrix with big flat submatrices. As a result it captures the most prominent

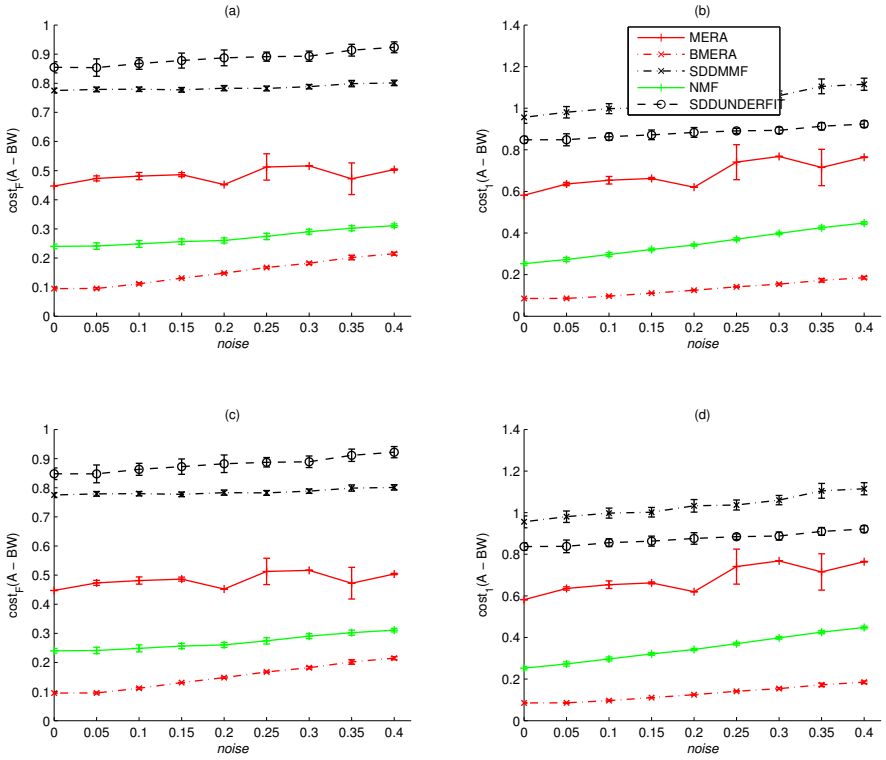

Figure 7.3: Reconstruction error with respect to the Frobenius error (left) and $\|\cdot\|_1$-error (right) when the level of noise varies from 0% to 40%. Max-times multiplication is used for all max-times methods. For `NMF` multiplication is with respect to max-times algebra on plots `(a)` and `(b)` and standard on plots `(c)` and `(d)`. Inner dimension is 10 and factor density is 0.25. Markers are mean values over 10 instances, and the error bars have height of twice the standard deviation.

features, and then further simplifies and generalizes them to be represented in a block form.

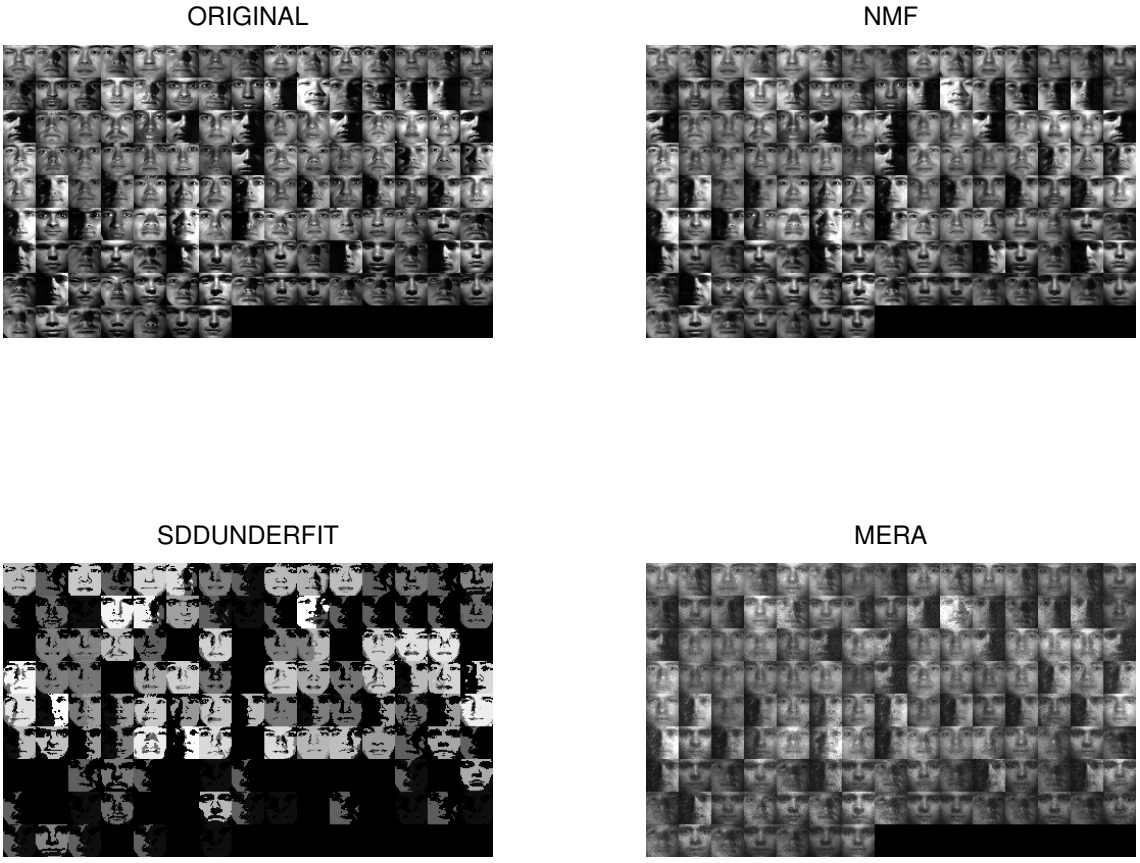

Figure 7.4: Faces recovered by `NMF`, `SDDUNDERFIT` and `MERA` algorithms compared to the original images from The Extended Yale Face Database B [GBK01, LHK05] resized to 32-by-32 format [CHH+07b]. The total of 127 images were used and the number of features is 15.

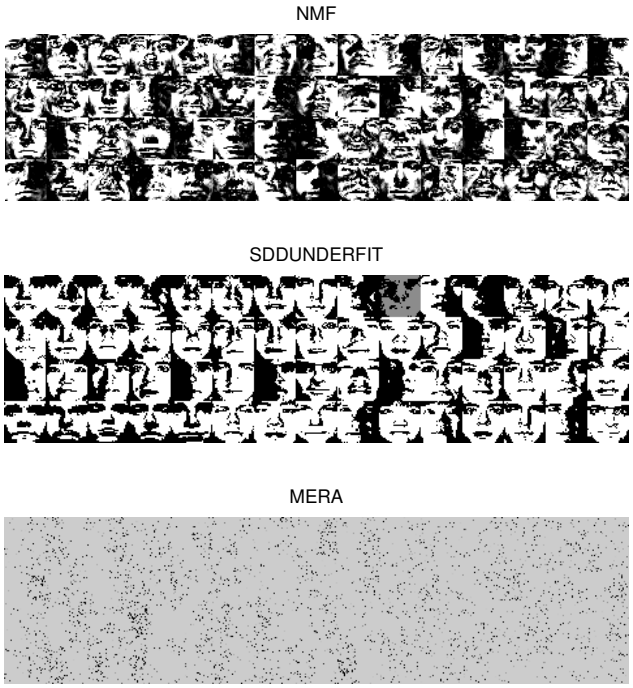

Figure 7.5: Features extracted by `NMF`, `SDDUNDERFIT` and `MERA` algorithms.

Chapter 8

Conclusions

We introduced a new type of matrix multiplication using maximization operation instead of addition, which is in some sense a hybrid of the standard and Boolean products. The thesis focused on finding factorizations with respect to thus defined matrix multiplication (the name of the problem is abbreviated as MMF, which stands for max matrix factorization).

The main objective of the thesis was to provide effective algorithms for MMF. To the best of our knowledge there is no published work on this exact problem. However, some authors worked on a closely related topic of max-times algebra. It is worth mentioning though, that they considered matrix decompositions over an extension of the max-times algebra, which is a different problem.

Unfortunately the MMF problem is computationally hard – at least as hard as BMF, which is known to be NP-hard. This makes finding best solutions in terms of the reconstruction error intractable for large data sizes and forces us to use heuristics and approximate algorithms.

We proposed several algorithms for the MMF problem, which can be broadly divided into two groups: those that solve the original problem – they are called direct, and those trying to substitute the objective with a function that is easier to handle, but that still exhibits the most important properties of the original objective – we call them relaxation methods. The former group includes `Greedy`, `SDDMMF` and `SDDUNDERFIT` algorithms. The relaxation algorithms are `MERA` and `BMERA`. They all show different characteristics and are suitable for different types of data. In particular the `SDD`-based algorithms are much better with block matrices and are almost unusable on general data. Overall `BMERA` algorithm is by far the best one in most cases, although `MERA` can be slightly superior for very dense data. The `Greedy` solver is not really an algorithm we propose for actual

solving. Its sole purpose is to illustrate the most naive approach, and obviously it does not perform well.

The motivation of the relaxation approach (`MERA` and `BMERA` algorithms) is that the maximization function, which is present in the objective, is not differentiable. It was shown that with exponential relaxation the objective can be approximated as closely as needed, while still maintaining differentiability.

Even after an appropriate relaxation the problem is still hard due to the objective function being nonconvex. This results in finding local minima and thus solving the problem suboptimally. One idea to address this issue is to represent the objective as a difference of convex functions. This approach was not covered in the thesis as currently it has not shown better results than other relaxation methods. However, it is certainly a promising direction for the future research.

Everywhere in this thesis we assume that the rank of the factorization is given. We do not address the problem of selecting the right rank, which is an interesting question on its own. It is trivial to see that the error never goes up if the rank is increased, but from the data mining perspective small error might not necessarily mean a good method since it could be hard to interpret the results.

It is obvious that the research conducted in this thesis is not exhaustive, rather we touched a small portion of a new area. Based on the results of the proposed algorithms, representing the objective as a difference of convex functions seems to be the most likely direction of the future work, as theoretically it should improve on `MERA` and `BMERA` algorithms. It is also quite possible that better algorithms will be found that use the direct approach. Only long and thorough research can answer these questions as the problem is new and requires more experience to make stronger statements.

List of Figures

Bibliography

[AH86] Robert J Aumann and Sergiu Hart. Bi-convexity and bi-martingales. *Israel Journal of Mathematics*, 54(2):159–180, 1986.

[BBL+07] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[BCOQ92] François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and linearity*, volume 3. Wiley New York, 1992.

[BDG+04] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514. ACM, 2004.

[BGT00] Vincent D Blondel, Stéphane Gaubert, and John N Tsitsiklis. Approximating the spectral radius of sets of matrices in the max-algebra is np-hard. *IEEE Transactions on Automatic Control*, 45(9):1762–1765, 2000.

[BHK97] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[BTGM04] Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101(12):4164–4169, 2004.

[But03] Peter Butkovič. Max-algebra: the linear algebra of combinatorics? *Linear Algebra and its applications*, 367:313–335, 2003.

[BV04] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CGQ99] Guy Cohen, Stéphane Gaubert, and Jean-Pierre Quadrat. Max-plus algebra and system theory: where we are and where to go now. *Annual Reviews in Control*, 23:207–219, 1999.

[CHH07a] Deng Cai, Xiaofei He, and Jiawei Han. Spectral regression for efficient regularized subspace learning. In *IEEE International Conference on Computer Vision (ICCV'07)*, 2007.

[CHH+07b] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conference Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*, 2007.

[CHHZ06] Deng Cai, Xiaofei He, Jiawei Han, and Hong-Jiang Zhang. Orthogonal laplacianfaces for face recognition. *IEEE Transactions on Image Processing*, 15(11):3608–3614, 2006.

[CL99] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems (The International Series on Discrete Event Dynamic Systems)*. Springer, 1 edition, September 1999.

[CLRS03] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill Science / Engineering / Math, 2nd edition, December 2003.

[CR93] Joel E Cohen and Uriel G Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by Latent Semantic Analysis. In *Journal of the American Society for Information Science*, volume 41, pages 391–407, 1990.

[Dhi01] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[DS58] Nelson Dunford and Jacob T Schwartz. *Linear Operators. Part 1: General Theory*. Interscience publishers New York, 1958.

[FP96] CA Floudas and PM Pardalos. *State of the Art in Global Optimization: Computational Methods and Applications. Nonconvex Optimization and Its Applications*. Kluwer Academic, Dordrecht, 1996.

[GBK01] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 23(6):643–660, 2001.

[GG10] Nicolas Gillis and François Glineur. Using underapproximations for sparse nonnegative matrix factorization. *Pattern Recognition*, 43(4):1676–1687, 2010.

[GGM04] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *Discovery science*, pages 278–289. Springer, 2004.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, January 1979.

[GPK07] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

[GV02] David Guillamet and Jordi Vitrià. Non-negative matrix factorization for face recognition. In *Topics in Artificial Intelligence*, pages 336–344. Springer, 2002.

[GVL12] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[Hal60] Paul Richard Halmos. *Naive set theory*. Springer, 1960.

[Har59] Philip Hartman. On functions representable as a difference of convex functions. *Pacific J. Math*, 9(3):707–713, 1959.

[Har72] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[HJ12] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[Hog06] Leslie Hogben. *Handbook of linear algebra*. CRC Press, 2006.

[HYH+05] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 27(3):328–340, 2005.

[JY11] Sunil K Jha and RDS Yadava. Denoising by singular value decomposition and its application to electronic nose data processing. *Sensors Journal, IEEE*, 11(1):35–44, 2011.

[Kel75] J.L. Kelley. *General Topology*. Graduate Texts in Mathematics. Springer, 1975.

[Kim82] Ki Hang Kim. *Boolean matrix theory and applications*, volume 70. Dekker New York, 1982.

[KO00] Tamara G Kolda and Dianne P O'Leary. Algorithm 805: computation and uses of the semidiscrete matrix decomposition. *ACM Transactions on Mathematical Software (TOMS)*, 26(3):415–435, 2000.

[KP08] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 353–362. IEEE, 2008.

[LHK05] K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 27(5):684–698, 2005.

[LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[Mie09] Pauli Miettinen. *Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms*. PhD thesis, Department of Computer Science, University of Helsinki, 2009.

[Mie10] Pauli Miettinen. Sparse boolean matrix factorizations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 935–940. IEEE, 2010.

[OP83] D O'Leary and Shmuel Peleg. Digital image compression by outer product expansion. *Communications, IEEE Transactions on*, 31(3):441–444, 1983.

[Paa97] Pentti Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and intelligent laboratory systems*, 37(1):23–35, 1997.

[Paa99] Pentti Paatero. The multilinear engineâĂŤa table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *Journal of Computational and Graphical Statistics*, 8(4):854–888, 1999.

[PSBP04] V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons. Text mining using nonnegative matrix factorizations. In *Proceedings of the fourth SIAM international conference on data mining*, pages 22–24. Society for Industrial Mathematics, 2004.

[PT94] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[RS81] Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics I: Functional Analysis*. Academic Press, New York, 1 edition, January 1981.

[Sch07] Satu E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[SM02] B. De Schutter and B. De Moor. The QR decomposition and the singular value decomposition in the symmetrized max-plus algebra revisited. *SIAM Review*, 44(3):417-454, 2002.

[SRJ04] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[SSU03] A. Schein, L. Saul, and L. Ungar. A generalized linear model for principal component analysis of binary data. In *In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[Whi95] J Eldon Whitesitt. *Boolean Algebra and its applications*. Courier Dover Publications, 1995.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[ZDLZ07] Zhongyuan Zhang, Chris Ding, Tao Li, and Xiangsun Zhang. Binary matrix factorization with applications. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 391–400. IEEE, 2007.