UNIVERSITÉ CATHOLIQUE DE LOUVAIN
ÉCOLE POLYTECHNIQUE DE LOUVAIN
DÉPARTEMENT D'INGÉNIERIE MATHÉMATIQUE

NONNEGATIVE MATRIX FACTORIZATION ALGORITHMS AND APPLICATIONS

NGOC-DIEP HO

Thesis submitted in partial fulfillment
of the requirements for the degree of
Docteur en Sciences de l'Ingénieur

Dissertation committee:
Prof. Vincent Wertz (President)
Prof. Paul Van Dooren (Promoter)
Prof. Vincent Blondel (Promoter)
Prof. François Glineur
Prof. Yurii Nesterov
Prof. Bob Plemmons
Prof. Johan Suykens

June 2008

ACKNOWLEDGMENTS

First, I would like to thank Prof. Paul Van Dooren and Prof. Vincent Blondel for the five long years as my advisors. Your support and advice have guided me through the most difficult moments of my research. Not only me, everyone will be really proud to have advisors like you. Paul and Vincent, thanks for your precious presence.

I am thankful to Prof. François Glineur, Prof. Yurii Nesterov, Prof. Bob Plemmons, Prof. Johan Suykens and Prof. Vincent Wertz for being on my dissertation committee.

Many thanks to the Centre for Systems Engineering and Applied Mechanics for providing me an excellent research environment, especially to Isabelle, Lydia and Michou and Etienne for their administrative and technical assistance and to all the members of the research group on Large Graphs and Networks for all activities both at work and off work.

I am also grateful to the small Vietnamese community and their friends in Louvain-la-Neuve for their help, to Alain and Frédérique who are simply very special.

Above all, I send my thanks to all my family in Vietnam for their daily support and I want to dedicate this thesis to my wife Hang, my daughter Mai and my son Nam who are always by my side.

This research has been supported by the Belgian Programme on Inter-university Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. It has been also supported by the ARC (Concerted Research Action) Large Graphs and Networks, of the French Community of Belgium. I was also a FRIA fellow (Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture).

TABLE OF CONTENTS

NOTATION GLOSSARY

$\mathbb{R}$	field of real numbers
$\mathbb{R}_+$	set of nonnegative real numbers
$\mathbb{R}_+^n$	set of nonnegative real vectors of size n
$\mathbb{R}_+^{m\times n}$	set of $m \times n$ nonnegative real matrices
$\Longleftrightarrow$	if and only if
$:=$	equal by definition to
$\dim X$	dimension of X
$\langle \cdot, \cdot \rangle$	generic inner product
$\| \cdot \|_p$	p-norm ($1 \leq p \leq +\infty$)
$\| \cdot \|_2$	Euclidean norm (vectors) / spectral norm (matrices)
$D(A\|B)$	generalized Kullback-Leibler divergence
e_i	unit vector $e_i = (0\,0\,\ldots\,\underbrace{1}_{i-th\ position}\,\ldots\,0)^T$
$\mathbf{1}_{m\times n}$	vector or matrix of all ones
I_k	$k \times k$ identity matrix
X^T	transpose of matrix X
X_{ij}	element located at the i^{th} row and the j^{th} column of X
$X_{i:}$	i^{th} row of the matrix X
$X_{:j}$	j^{th} column of the matrix X
$vec(X)$	vector formed by stacking the columns of X into one vector
$rank(X)$	rank of matrix X
$rank_{UV^T}(X)$	nonnegative rank of matrix X
$rank_{VV^T}(X)$	completely positive rank of matrix X
$\det X$	determinant of square matrix X
$trace(X)$	trace of square matrix X

$\lambda_k(X)$	k-th eigenvalue of matrix X
$\sigma(X)$	set of eigenvalues of the matrix X
$\rho(X)$	$\max_i \lvert\lambda_i(X)\rvert$
$\sigma_{\max}(X)$	maximal singular value of matrix X
$\sigma_{\min}(X)$	minimal singular value of matrix X
$A \otimes B$	Kronecker product between matrices A and B
$A \circ B$	Hadamard product between matrices A and B
$\frac{[A]}{[B]}$	Hadamard division between matrices A and B
$[A]_+$	projection of A onto the nonnegative orthant
$D(v)$	diagonal matrix with v on the main diagonal

Abbreviations and acronyms

NMF	Nonnegative Matrix Factorization
SNMF	Symmetric Nonnegative Matrix Factorization
SSNMF	Semi-Symmetric Nonnegative Matrix Factorization
WNMF	Weighted Nonnegative Matrix Factorization
SVD	Singular Value Decomposition

INTRODUCTION

In every single second in this modern era, tons of data are being generated. Think of the number of online people writing their blogs, designing their homepages and sharing their experiences through many other digital supports: videos, photos, etc. Think also of the data generated when decoding genes of living creatures and the data acquired from the outer space or even from our own planet, etc.

Data only become useful when having been processed. In front of this fast-growing amount of data, there are several approaches for data processing: applying classical methods, designing more powerful computing structures such as distributed computing, multicore processors, supercomputers, etc. But the growing amount and complexity of accumulated data seems to outweigh the growth of computing power which is, at the present time, roughly doubling every year (cfr. Moore's Law [90]). One very popular approach is called model reduction which tries to reduce the complexity while keeping the essentials of the problem (or data).

Besides, different types of data require different models to capture the insight of the data. Using the right model saves a lot of time. Of course, a model believed to be right will stand until a better model is found. An ideal model may not exist. For instance, using dominant subspaces with the Singular Value Decomposition (SVD) [50] has been proposed as the best model to reduce the complexity of data and complicated systems. It offers the least error (with respect to some measures) with the same reduced complexity, compared to other models. But it is not the only one since the conic representation or conic coding [82] is also extensively used. Its properties favour the *additive model* of some types of data while SVD-related techniques do not. In this thesis,

we focus on finding the best reduced conic representation of nonnegative data through Nonnegative Matrix Factorization (NMF). We will go through several issues that are considered as the building blocks for the nonnegative matrix factorization (NMF).

For nonnegative data, we will see that this additive model offers a closer physical representation to the reality than other techniques such as the SVDs. But this is not for free. On the one hand, SVD decomposition is known to have *polynomial-time* complexity. In fact, it can be done in a polynomial number, i.e. $nm\min(m,n)$, of basic operations for a full decomposition, where n and m are the dimensions of the matrix data [50]. When only a partial SVD decomposition is needed, iterative methods can be applied with the computational load of mnr basic operations per iteration, where r, $1 \geq r \geq \min(m,n)$, is the reduced dimension of the decomposition. Their convergence speed, represented by the number of iterations needed, has been being improved drastically, which allows us to process massive data sets. On the other hand, NMF factorization has been recently proved to have a *nondeterministic polynomial - NP* computational complexity [121] for which the existence of a polynomial-time optimal algorithm is unknown. However, iterative methods with low computational load iterations are still possible. There are iterative methods whose computational load per iteration is roughly equivalent to the one of SVD, i.e. mnr, such as [81], [86], etc. as well as the one described in this thesis. But only *acceptable* solutions are expected rather than the *optimal* one. And restarts maybe needed. The main aspects that differentiate these methods are then: *to which solution they tend to converge? how fast they converge? and how to drive them to converge to solutions that possess some desired properties?*

Part-based analysis

An ordinary object is usually a collection of simple parts connected by some relations between them. Building objects from basic parts is one of the simplest principle applicable to many human activities. Moreover, human vision is *designed* to be able to detect the presence or the absence of features (parts) of a physical object. Thanks to these parts, a human can recognize and distinguish most of the objects [15].

Without taking into account the possible relations between parts and assuming that we can establish a full list of all possible parts of all possible objects, then there is one unified formula for composing the objects:

$$\text{Object}_i = \text{Part}_1(b_{i1}) \quad \textit{with} \quad \text{Part}_2(b_{i2}) \quad \textit{with} \ldots,$$

where

$$b_{ij} = \begin{cases} \textit{present} & \text{if part } i \text{ is present in object } j \\ \textit{absent} & \text{if part } i \text{ is absent from object } j. \end{cases}$$

Then we can represent object i by a list $(b_{i1}, b_{i2}, \ldots)$ that can be simplified by replacing the status *present* and *absent* by 1 and 0. This model can be improved by taking into account the *quantity* of parts inside an object. The final recipe for making an object is then something like

$$\text{Object}_i = b_{i1} \times \text{Part}_1 \quad + \quad b_{i2} \times \text{Part}_2 \quad + \ldots$$

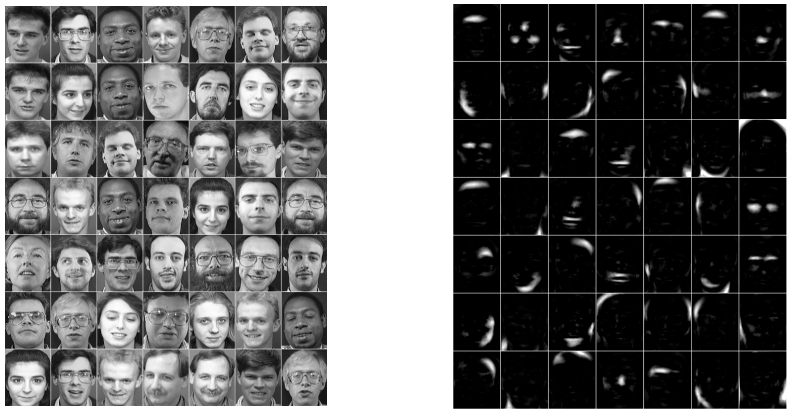
where $b_{ij} \geq 0$.

In reality, only some objects are available through observations. The task is then to detect parts from observed objects and to use the detected parts to reconstitute these objects. This simple idea appears in many applications and will be illustrated in the following examples.

Image learning

Digital image processing has been a hot topic in recent years. This includes face recognition [54], optical character recognition [70], content-based image retrieval [109], etc. Each monochrome digital image is a rectangular array of pixels. And each pixel is represented by its light intensity. Since the light intensity is measured by a nonnegative value, we can represent each image as a nonnegative matrix, where each element is a pixel. Color images can be coded in the same way but with several nonnegative matrices.

An example is the Cambridge ORL face database. It contains 400 monochrome images of a front view of the face of 40 persons (10 images per person). The size of each image is 112×92 with 256 gray levels per pixel. Some randomly selected images are shown on the left of the following figure.



We want to decompose those images as:

$$\text{Image}_i = b_{i1} \times \text{Feature}_1 + b_{i2} \times \text{Feature}_2 + \dots$$

where $b_{ij} \geq 0$ are the participation weight of feature j in image i. A procedure similar to that proposed in [80] is used to extract a list of pertinent features on which some sparsity constraints are imposed. These features are shown on the right of the above figure. Each of the images in the database is then reconstructed by a nonnegative mixture of those features.

The method that was used to construct these features guarantees not only a *good* reconstruction of images but also the nonnegativity of the features. Therefore, each feature can be considered again as an image. Together with the participation of each feature in an image, one can establish the composition of every image in a very comprehensible way.

Document classification

Textual data is an important source of information. The smallest meaningful units of texts are words. A sequence of words constitutes a document. Usually, each document is about a specific topic or category. In some cases, for instance news, school courses, etc, these categories are specified. But in others such as blogs, discussion groups, they may not

Topic 1	Topic 2	Topic 3	Topic 4
court	president	flowers	disease
government	served	leaves	behavior
council	governor	plant	glands
culture	secretary	perennial	contact
supreme	senate	flower	symptoms
constitutional	congress	plants	skin
rights	presidential	growing	pain
justice	elected	annual	infection

be listed. Moreover, a classification is hardly unique, several different classifications can be defined. For instance, news articles can be classified not only with topics such as: economics, cultures, sports, sciences, etc. but also according to the geographical regions (Asia, Europe, Africa, etc).

Without a grammar, a text can be seen as a set of words combined with their number of occurrences. Given a collection of texts, one wants to automatically discover the hidden classifications. The task is then to try to *explain* a text as:

$$\text{Text}_i = b_{i1} \times \text{Topic}_1 + b_{i2} \times \text{Topic}_2 + \dots$$

where b_{ij} can be considered as the similarity or the participation of topic j to text i. Topics are characterized by a list of keywords, which describe their semantics.

The above table shows some topics discovered by Nonnegative Matrix Factorization from 30991 articles from the Grolier encyclopedia and reported in Nature [80]. In each column of the table, a discovered topic is represented by a list of keywords taken from the 15276-word vocabulary. Both topics and their keywords are automatically retrieved.

In reality, the topic of a document is often not *pure*, in the sense that each document can belong to a number of topics. Carrying on with the above example, the "Constitution of the United States" entry is, in fact, semantically a mixture of different categories with some weights. Their experiment shows that it is strongly related to the first two topics and weakly (almost not) related to the last two. And this matches perfectly our intuition.

Having discovered a list of topics and their participation in each document, one can not only decide to which topics a document belongs, but also deal with the polysemy of words, detect new trends, reveal hidden categories, etc.

Why nonnegativity?

As we have seen, for the part-based analysis, the presence or absence of parts creates *recipes* for making an object. This existential status of parts is represented by nonnegative numbers, where 0 represents an absence and a positive number represents a presence with some degree. Furthermore, the objects are also represented by a set of nonnegative numbers, e.g., numbers of occurrences or light intensities. Because of that, nonnegativity is a crucial feature that one needs to maintain during the analysis of objects.

The part-based analysis is also referred to as the additive model because of the absence of subtractions in the model. This follows from the construction of an object:

$$\text{Object}_i = b_{i1} \times \text{Part}_1 \quad + \quad b_{i2} \times \text{Part}_2 \quad + \ldots$$

Allowing subtractions, i.e., $b_{ij} < 0$ for some i, j implies that any part can be interpreted as either a positive or a negative *quantity*. When each part belongs to a vector space, this is true since the orientation does not change the spanning space. But for other types of data such as: the concentrations of substances, absolute temperatures, light intensities, probabilities, sound spectra, etc. negative quantities do not arise. Decomposing nonnegative objects with general methods like the *Singular Value Decomposition* significantly alters the physical interpretation of the data. Other analysis tools like the *Principal Component Analysis* require some features which nonnegative objects by their nature never or hardly possess, such as *zero sum*, *orthogonality* etc.

While much evidence from existing applications shows the appeal of the part-based method, the lack of algorithmic understandings prohibits extensions to larger scale problems and to other types of data. This gave us enough motivations to focus on a better tool for nonnegative data through a thorough study of *nonnegative matrix factorization*. This technique allows us to approximate nonnegative objects, stored in columns

of a nonnegative matrix A, by the product of two other nonnegative matrices U and V:

$$A \approx UV^T.$$

This factorization captures all the key ideas from the above examples of the part-based analysis. Columns of U define the extracted *parts* (or image features or document topics, etc.). The matrix V describes the participations of those parts in the original objects (or images, document, etc.).

The idea of approximating a nonnegative matrix A by the product UV^T of two nonnegative matrices U and V is not new. In fact, it is a generalized method of the well-known K-Means method [88] from 1967, applied to nonnegative data. Suppose we have n nonnegative data vectors $a_1, \ldots, a_n$ and r initial centroids $u_1, \ldots, u_r$ representing r clusters $C_1, \ldots, C_r$, what the K-Means method does is to repeat the following two steps until convergence:

1. For each a_i, assign it to C_j if u_j is the nearest centroid to a_i, with respect to Euclidean distance.

2. For each u_j, replace it with the arithmetic mean of all a_i in C_j.

We can construct a matrix U by putting all the vectors u_j in the columns of U and create a matrix V such that

$$V_{ij} = \begin{cases} 1 & \text{if } a_i \in C_j \\ 0 & \text{otherwise,} \end{cases}$$

It turns out that the K-Means method tries to minimize the Euclidean distance between matrices A and UV^T. Moreover, because each column of U is a mean of some nonnegative vectors, both matrices U and V are nonnegative. Mathematically, we solve

$$\min_{U,V} \|A - UV^T\|_F^2$$

where A and U are nonnegative matrices, V is a binary matrix in which each row of V contains one and only one element equal to 1 and $\|A - UV^T\|_F^2$ denotes the Euclidean distance between A and UV^T. Two above iterative steps of the K-Means method are, in fact, the optimal solution of the following subproblems:

(P1) $\min_U \|A - UV^T\|_F^2$,
(P2) $\min_V \|A - UV^T\|_F^2$
with the special structure of V.

Nonnegative matrix factorization is different from the K-Means method only in the structure matrix V. Instead of binary matrix as above, in NMF, V taken to be a normal nonnegative matrix. This little difference offers more flexibility to NMF as well as more difficulty to optimally solve the two above subproblems. However, we will still see the same iterations (P1) and (P2) in a number of NMF algorithms in this thesis.

K-Means had been being applied successfully to many problems long before the introduction of NMF factorization in the nineties. Therefore, it is not surprising that the NMF, the generalized version of K-Means, has recently gained a lot of attention in many fields of application. We believe that preserving nonnegativity in the analysis of originally nonnegative data preserves essential properties of the data. The loss of some mathematical precision due to the nonnegativity constraint is compensated by a meaningful and comprehensible representation.

Thesis outline

The objective of the thesis is to provide a better understanding and to propose better algorithms for nonnegative matrix factorization. Chapter 2 is about various aspects of the nonnegative matrix factorization problem. Chapters 3 and 4 are about its algorithmic aspects. And the last three chapters are devoted to some extensions and applications. Here is the outline of each chapter:

- Chapter 1: **Preliminaries**. Some basic results and concepts used throughout the thesis are presented. Known results are shown without proof but references are given instead. This chapter is also a concise introduction to the main notations.
- Chapter 2: **Nonnegative matrix factorization**. This chapter is devoted to the introduction, the optimality conditions, the representations of the factorization, the solution for some easy cases, and the characterization of local minima of the nonnegative matrix factorization problem. The exact nonnegative factorization and nonnegative ranks are also discussed. Some interesting extensions of the factorization are also introduced such as: multilayer nonnegative matrix factorization and nonnegative tensor factorization.
- Chapter 3: **Existing algorithms**. In this chapter, investigations are carried out to clarify some algorithmic aspects of the existing algorithms such as: the *multiplicative updates*, *gradient based methods* and the *alternating least square*. Other algorithmic aspects like initializations and stopping conditions are also treated.
- Chapter 4: **Rank-one residue iteration**. This chapter is an extension of the report [62], where we proposed to decouple the problem based on rank-one approximations to create a new algorithm. A convergence analysis, numerical experiments and some extensions were also presented for this algorithm. Two other independent reports [31] and [49] have also proposed this algorithm. Numerical experiments are summarized at the end of the chapter to compare the performance of the newly proposed method to existing ones. It is seen that this method has good and fast convergence, and is suitable for large-scale problems. Moreover, it does not require

any parameter setting, which is an advantage over some other methods.

- Chapter 5: **Nonnegative matrix factorization with fixed row and column sums**. We introduce a new problem in nonnegative matrix factorizations where row and column sums of the original matrix are preserved in approximations. After some discussions of the problem, we prove that by using the generalized Kullback-Leibler divergence, one can produce such a factorization naturally. This also links the proposed method to Probabilistic Latent Semantic Analysis (pLSA) [65] and creates some applications such as: approximation of stochastic matrices, approximation that preserves the Perron vectors, etc.

- Chapter 6: **Weights in nonnegative matrix factorization**. This chapter incorporates weights into the nonnegative matrix factorization algorithms. We also extend the multiplicative rules to take weights into account. We also point out a link between the weighted Euclidean distance and the weighted generalized Kullback-Leibler divergence. A numerical experiment is carried out on the database of human facial images where weights are added to emphasize some image parts.

- Chapter 7: **Symmetry in nonnegative matrix factorization**. Some symmetric structures are imposed on the nonnegative matrix factorization. While solving the *exact* symmetric nonnegative matrix factorization is a hard problem, related to the class of completely positive matrices, approximating methods can nevertheless be designed. Several variants are treated. At the end, we mention two applications: graph clustering and nonnegative factorization of the correlation matrices.

Some conclusions drawn from our research end the thesis.

Related publications

2005. V.D. Blondel, N.-D. Ho and P. Van Dooren - *Nonnegative matrix factorization - Applications and Extensions*. Technical Report 005 – 35, Cesame. University catholique de Louvain. Belgium.

2005. N.-D. Ho and P. Van Dooren - *On the Pseudo-inverse of the Laplacian of a Bipartite Graph*. Applied Math. Letters, vol.18 p.917-922, 2005.

2006. A. Vandendorpe, N.-D. Ho, S. Vanduffel and P. Van Dooren - *On the parameterization of the CreditRisk+ model for estimating credit portfolio risk*. To appear in Insurance: Mathematics and Economics.

2007. V. D. Blondel, N.-D. Ho and P. Van Dooren - *Weighted Nonnegative Matrix Factorization and Face Feature Extraction*. Submitted to Image and Vision Computing.

2007. N.-D. Ho and P. Van Dooren - *Nonnegative Matrix Factorization with fixed row and column sums*. Linear Algebra and Its Applications (2007), doi:10.1016/j.laa.2007.02.026.

2007. N.-D. Ho, P. Van Dooren and V. D. Blondel - *Descent algorithms for Nonnegative Matrix Factorization*. Survey paper. To appear in Numerical Linear Algebra in Signals, Systems and Control.

1
PRELIMINARIES

This chapter introduces the basic results and concepts used throughout this thesis. Known results are only stated without proof.

1.1 Matrix theory and linear algebra

A $m \times n$ real matrix is a m-row and n-column table containing real scalars. We have a square matrix when the number of rows is equal to the number of columns. The set of $m \times n$ real matrices is denoted by $\mathbb{R}^{m \times n}$. In this thesis, all matrices are real. We use uppercase letters for matrices. The i^{th} row of the matrix A is denoted by $A_{i:}$. The j^{th} column of the matrix A is denoted by $A_{:j}$. The element at the intersection of the i^{th} row and the j^{th} column of the matrix A is denoted by A_{ij} or $[A]_{ij}$.

A column vector is a matrix of only one column. Likewise, a row vector is a matrix of only one row. Unless explicitly stated otherwise, a vector is always a column vector. The set of all size-n vectors is $\mathbb{R}^n$. Vectors are denoted by lowercase letters except when they are parts of a matrix as described in the preceding paragraph.

A $n \times n$ square matrix A is said to be symmetric if $A_{ij} = A_{ji}$, for all i, j. A diagonal matrix D is a square matrix having nonzero elements only on its main diagonal (i.e., $A_{ij} = 0$ for $i \neq j$). We use D_x to denote a diagonal matrix with the vector x on its main diagonal (i.e. $A_{ii} = x_i$, $i = 1, \ldots, n$).

Here are some special matrices:

- Matrices whose elements are all 1: $\mathbf{1}_{1\times n}$, $\mathbf{1}_{m\times 1}$, $\mathbf{1}_{m\times n}$.

$$\mathbf{1}_{1\times n} = (1,1,\ldots,1) \quad \mathbf{1}_{m\times 1} = (1,1,\ldots,1)^T \quad \mathbf{1}_{m\times n} = \mathbf{1}_{m\times 1}\mathbf{1}_{1\times n}.$$

- Unit vectors

$$e_i = (0,0,\ldots,\overbrace{1}^{i^{th}\text{ position}},\ldots,0)^T.$$

- Identity matrices I_n: diagonal matrices where diagonal elements are equal to 1.
- Permutation matrices: square matrices having on each row and each column only one nonzero element which is equal to 1.
- Selection matrices: any submatrices of permutation matrices.

1.1.1 Matrix manipulation

Here are some basic matrix operators

- Matrix transpose A^T: $\left[A^T\right]_{ij} := A_{ji}$. A is a symmetric matrix $\Leftrightarrow A^T = A$.
- Matrix addition $C = A + B$: $C_{ij} := A_{ij} + B_{ij}$.
- Matrix product $C = A.B$: $C_{ij} := \sum_k A_{ik}.B_{kj}$. The product dot is often omitted.
- Matrix vectorization of $A \in \mathbb{R}^{m\times n}$

$$vec(A) = \begin{pmatrix} A_{:1} \\ \vdots \\ A_{:n} \end{pmatrix} \in \mathbb{R}^{mn}.$$

- Kronecker product of matrix $A \in \mathbb{R}^{m\times n}$ and matrix B

$$A \otimes B = \begin{pmatrix} A_{11}B & \ldots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \ldots & A_{mn}B \end{pmatrix}.$$

An important relation between the matrix product and the Kronecker product is the following [118]:

$$vec(AXB^T) = (B \otimes A)vec(X).$$

We write $A < B$ if $A_{ij} < B_{ij}$ for all i, j and similarly for $A \leq B$, $A > B$ and $A \geq B$. We use $A < \alpha$, $A > \alpha$, $A \leq \alpha$ and $A \geq \alpha$, where $\alpha \in \mathbb{R}$, as abbreviations of $A < \alpha \mathbf{1}_{m\times n}$, $A > \alpha \mathbf{1}_{m\times n}$, $A \leq \alpha \mathbf{1}_{m\times n}$ and $A \geq \alpha \mathbf{1}_{m\times n}$. The absolute matrix $|A|$ is defined as: $[|A|]_{ij} = |A_{ij}|$ for all i, j.

We define the *inner product* of the two real vectors $x,\ y \in \mathbb{R}^n$ as a real functional:

$$\langle x, y \rangle = \sum_i x_i y_i = x^T y.$$

Nonzero vectors $x,\ y \in \mathbb{R}^n$ are said to be *orthogonal* if their inner product is zero:

$$\langle x, y \rangle = 0.$$

Considering a general matrix $A \in \mathbb{R}^{m\times n}$ as a vector: $vec(A) \in \mathbb{R}^{mn}$, we can also define the inner product of two real matrices of the same size:

$$\langle A, B \rangle = vec(A)^T vec(B) = \sum_{ij} A_{ij} B_{ij} = trace(A^T B),$$

where the trace of A ($trace(A)$) is the sum of all the diagonal elements of A. This implies the following useful relation:

$$\langle I, ABC \rangle = \left\langle A^T, BC \right\rangle = \left\langle B^T A^T, C \right\rangle = \left\langle C^T B^T A^T, I \right\rangle = trace(ABC).$$

A square matrix A is said to be *invertible* if there exists a matrix B such that

$$AB = BA = I,$$

where B is called the inverse of A and is denoted by $B = A^{-1}$. While not all matrices have an inverse, the *pseudo-inverse* (or Moore-Penrose pseudoinverse) is its generalization, even to rectangular matrices. The uniquely defined pseudo-inverse A^+ of the matrix A satisfies the following four conditions:

$$AA^+A = A,\ \ A^+AA^+ = A^+,\ \ (AA^+)^T = AA^+ \ \text{ and } \ (A^+A)^T = A^+A.$$

In particular, if A^TA is invertible, then $A^+ = (A^TA)^{-1}A^T$.

The matrix sum $C = A + B$ is defined as $C_{ij} = A_{ij} + B_{ij}$. This operator is said to be *elementwise* or *entrywise* since each entry of the result matrix C depends only on entries of A and B at the same position. This is contrary to the usual matrix product $C = AB$ where the relations are no longer local. A simpler matrix product that is elementwise is called the Hadamard Product or Schur Product $C = A \circ B$ where $C_{ij} = A_{ij}B_{ij}$ and A, B and C are $m \times n$ matrices. This helps considerably to simplify matrix formulas in many cases. Here are some properties of the Hadamard product [67]:

- $A \circ B = B \circ A$
- $A^T \circ B^T = (A \circ B)^T$
- $(a \circ b)(c \circ d)^T = (ac^T) \circ (bd^T) = (ad^T) \circ (bc^T)$

The following are some relations of the Hadamard product with other operators:

- $\mathbf{1}^T(A \circ B)\mathbf{1} = \langle A, B \rangle$
- $A \circ B = P^T(A \otimes B)Q$, where P and Q are selection matrices

  $$P = (e_1 \otimes e_1 \; e_2 \otimes e_2 \; \ldots \; e_m \otimes e_m)$$

  and

  $$Q = (e_1 \otimes e_1 \; e_2 \otimes e_2 \; \ldots \; e_n \otimes e_n).$$

  Roughly speaking, $A \circ B$ is a submatrix of $A \otimes B$.

From the definition of the Hadamard product, we can define other elementwise operators:

- Hadamard power: $[A^{\circ r}]_{ij} = A_{ij}^r$, $r \in \mathbb{R}$.
- Hadamard division: $C = \frac{[A]}{[B]} = A \circ B^{\circ -1}$.

1.1.2 Vector subspaces

A linear subspace E of $\mathbb{R}^n$ is the set of all linear combinations of a set of vectors $V = \{v_1, v_2, \ldots, v_k\}$ of $\mathbb{R}^n$:

$$E = \left\{ \sum_{i=1}^{k} \alpha_i v_i \mid \alpha_i \in \mathbb{R} \right\}.$$

E is also called the *span* of V and V is called a spanning set of E. Given a subspace E, there are many spanning sets. Among them, a set from which no vector can be removed without changing the span is said to be *linear independent* and a basis of E. The cardinality of a basis of E is fixed and is called the dimension of E.

For example:

$$E = span\left(\left\{(1,2,1)^T, (1,0,0)^T\right\}\right)$$

is a subspace of $\mathbb{R}^3$ and $dim(E) = 2$ since $\left\{(1,2,1)^T, (1,0,0)^T\right\}$ is linear independent. Following this, the *rank* of a $m \times n$ matrix A can also be defined as the dimension of the subspace spanned by the columns of A:

$$rank(A) = dim\left(span(A_{:1},\ A_{:2}, \ldots, A_{:n})\right) \leq min(m,n).$$

A linear subspace is closed under addition and scalar multiplication, i.e.,

$$u, v \in E \ \Rightarrow\ u + v \in E,$$
$$u \in E, \alpha \in \mathbb{R} \ \Rightarrow\ \alpha u \in E.$$

1.1.3 Eigenvalues and eigenvectors

Central concepts in matrix analysis are eigenvalues and eigenvectors of a square matrix. They provide essential information about the matrix. Related concepts for rectangular matrices are so-called singular values and vectors. They play a crucial role in low-rank approximations that retain dominating characteristics of the original matrix.

Definition 1.1. A scalar $\lambda \in \mathbb{C}$ is an eigenvalue of the matrix $A \in \mathbb{C}^{n \times n}$ if there exists a nonzero vector $x \in \mathbb{C}^n$ such that $Ax = \lambda x$. The vector x is called the associated eigenvector of the eigenvalue λ.

An $n \times n$ matrix has exactly n eigenvalues (multiplicity counted). The set of all the eigenvalues is denoted by $\sigma(A)$. The maximum modulus of $\sigma(A)$ is the spectral radius of A and is denoted by $\rho(A)$:

$$\rho(A) = \max\{|\lambda| \mid \lambda \in \sigma(A)\}.$$

In this thesis, only eigenvalues and eigenvectors of some symmetric matrices are investigated. For those matrices, the following well-known results can be established:

Theorem 1.2 (Spectral Theorem)**.** *Let A be a real symmetric matrix. All eigenvalues and eigenvectors of A are real.*

Moreover, for a real symmetric matrix A, if all the eigenvalues of A are nonnegative (respectively nonpositive), A is said to be *positive semidefinite* (respectively *negative semidefinite*). If all the eigenvalues are positive (respectively negative), A is said to be *positive definite* (respectively *negative definite*).

A very useful tool in matrix analysis is the *Singular Value Decomposition* defined in the following theorem:

Theorem 1.3. *For any matrix $A \in \mathbb{R}^{m\times n}$, there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that*

$$A = U\Sigma V^T; \tag{1.1}$$

$$\Sigma = \left(\begin{array}{ccc|c} \sigma_1 & & 0 & \\ & \ddots & & O_{r\times(n-r)} \\ 0 & & \sigma_r & \\ \hline & O_{(m-r)\times r} & & O_{(m-r)\times(n-r)} \end{array}\right), \tag{1.2}$$

where the singular values σ_i are real and non-increasing scalars :

$$\sigma_1 \geq \ldots \geq \sigma_r > 0. \tag{1.3}$$

Proof and algorithms can be found in [50]. Moreover, the columns of U and V are the eigenvectors of A^TA and AA^T, respectively.

1.1.4 Norms

A *norm* is used to measure the magnitude of a vector or a matrix. A norm on $\mathbb{R}^n$ (or $\mathbb{R}^{m\times n}$) is a real functional $\|.\|$ on $\mathbb{R}^n$ (or $\mathbb{R}^{m\times n}$) that satisfies the following four conditions:

$$\begin{aligned}
&\|x\| \geq 0, \forall x \in \mathbb{R}^n \text{ (or } \mathbb{R}^{m\times n});\\
&\|x\| = 0 \Longleftrightarrow x = 0;\\
&\|\alpha x\| = |\alpha|\|x\|, \forall x \in \mathbb{R}^n \text{ (or } \mathbb{R}^{m\times n}) \text{ and } \forall \alpha \in \mathbb{R};\\
&\|x + y\| \leq \|x\| + \|y\|, \forall x, y \in \mathbb{R}^n \text{ (or } \mathbb{R}^{m\times n}).
\end{aligned}$$

A most common norm is the Euclidean norm or the Frobenius norm derived from the inner product:

$$\|x\|_F = \sqrt{\langle x, x\rangle},$$

where x can be either a vector or a matrix. This norm plays the central role in least squares problems, where one tries to minimize an error measured by this norm.

Popular norms are instances of the Hölder norms (p-norm):

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \ p = 1, 2, \ldots$$

where the most commonly used are $p = 1$, $p = 2$ and $p = \infty$:

$$\begin{aligned}
&\text{1-norm: } \|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|\\
&\text{2-norm: } \|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2}\\
&\infty\text{-norm: } \|x\|_\infty = \max_i |x_i|.
\end{aligned}$$

For vectors, the 2-norm ($\|.\|_2$) is also the Frobenius norm ($\|.\|_F$). But this is no longer true for matrix p-norms, which are induced from vector p-norms :

$$\|A\|_p = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

It is proved that [67]

$$\begin{aligned}
&\text{1-norm: } \|A\|_1 = \max_j \textstyle\sum_i |A_{ij}|\\
&\text{2-norm: } \|A\|_2 = \left[\rho(A^T A)\right]^{1/2}\\
&\infty\text{-norm: } \|A\|_\infty = \max_i \textstyle\sum_j |A_{ij}|.
\end{aligned}$$

Since the main problem treated in this thesis is a constrained least squares problem, the Frobenius norm will be extensively used. Other norms will also be used to add more constraints on the main problem.

1.1.5 Convex cone and polyhedral cone

A set $C \subset \mathbb{R}^n$ is call a convex cone if it is closed under the addition and the *nonnegative* scalar multiplication, i.e.

$$\begin{aligned} &u, v \in C \;\Rightarrow\; u + v \in C, \\ &u \in C, \alpha \geq 0 \;\Rightarrow\; \alpha u \in C. \end{aligned}$$

A polyhedral cone is a convex cone nonnegatively generated by a finite set of vectors $V = \{v_1, v_2, \ldots, v_k\}$ of $\mathbb{R}^n$:

$$C = \left\{ \sum_{i=1}^{k} \alpha_i v_i \mid \alpha_i \in \mathbb{R}_+ \right\}.$$

In this relation, C is also called the *span* of V and V is call a spanning set of C. There exists a set $\bar{V} \subset V$ that nonnegatively generates C and from which no vector can be removed without changing the cone. $\bar{V}$ is called the frame of C and its cardinality is called the dimension of C.

1.1.6 Nonnegative matrices

Matrices whose elements are all nonnegative are called nonnegative matrices. We use $\mathbb{R}^n_+$ and $\mathbb{R}^{m\times n}_+$ to denote the set of n-dimensional nonnegative vectors and the set of $m \times n$ nonnegative matrices, respectively. These subsets are, indeed, polyhedral cones and usually called the nonnegative orthants.

A nonnegative matrix is called row-allowable if it has no zero row. Similarly, a nonnegative matrix is called column-allowable if it has no zero column. A nonnegative matrix is said to be column (row) stochastic if all the column (row) sums are equal to one. A nonnegative matrix is said to be doubly stochastic if it is column stochastic and row stochastic.

The most important result for nonnegative matrices is the following:

Theorem 1.4 (Perron-Frobenius, see [8])**.** *Let A be a square nonnegative matrix. There exist a largest modulus eigenvalue of A which is nonnegative and a nonnegative eigenvector corresponding to it.*

This vector is usually referred to as the Perron vector of the nonnegative matrix. For a rectangular nonnegative matrix, similar results can be established for the largest singular value and its corresponding singular vectors.

Given a subset $V \subset \mathbb{R}^{m\times n}$ and a matrix $A \in \mathbb{R}^{m\times n}$, the nearest element of V to A (with respect to a distance) is called the projection of A on V, denoted by $P_V(A)$. When the target subset V is the nonnegative orthant and the considered distance is the Euclidean distance, the projection of A is denoted by $[A]_+$ and defined as:

$$[[A]_+]_{ij} = \begin{cases} A_{ij} & \text{if } A_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} = max(0, A_{ij}).$$

1.2 Optimization

Before presenting some basic results about optimization, we review the concept of convex sets and convex functions.

1.2.1 Convex set and convex function

Definition 1.5 (Convex sets)**.** A set Ω is said to be convex if and only if for every $u, v \in \Omega$, we have

$$\alpha u + (1-\alpha)v \in \Omega, \quad \text{for all } \alpha \in [0,1].$$

Clearly, the convex cones and the polyhedral cone ares, by construction, convex sets, which implies that the set of $m \times n$ nonnegative matrices (the nonnegative orthant $\mathbb{R}_+^{m\times n}$) is also a convex set. The set $\mathbb{R}_+^{m\times n}$ is one of the main objects used in this thesis.

Definition 1.6 (Convex functions)**.** A function f defined on a convex set Ω is said to be convex if for every $u, v \in \Omega$ and every $\alpha \in [0,1]$, the following holds:

$$f\left(\alpha u + (1-\alpha)v\right) \leq \alpha f(u) + (1-\alpha) f(v).$$

If for every $\alpha \in (0,1)$ and $u \neq v$, the following holds:

$$f\left(\alpha u + (1-\alpha)v\right) < \alpha f(u) + (1-\alpha) f(v),$$

then the function is said to be strictly convex.

For more details about convex sets and convex functions, see [21].

1.2.2 Optimality conditions

Now, we summarize some basic results on the optimization problem

$$\min_{x\in\Omega} f(x),$$

where f is a real-valued function taken on the feasible set $\Omega \subset \mathbb{R}^n$.

We distinguish two types of minima.

Definition 1.7 (Local minimum)**.** A point $x^* \in \Omega$ is said to be a local minimum of f over Ω if there exists an open neighborhood $N(x^*)$ of x^* such that for all $x \in N(x^*) \cap \Omega$, $f(x) \geq f(x^*)$. It is considered a strict local minimum if for all $x \in N(x^*) \cap \Omega$ and $x \neq x*$, $f(x) > f(x^*)$.

Definition 1.8 (Global minimum)**.** A point $x^* \in \Omega$ is said to be a global minimum of f over Ω if for all $x \in \Omega$, $f(x) \geq f(x^*)$. A point $x^* \in \Omega$ is said to be a strict global minimum of f over Ω if for all $x \in \Omega$, $x \neq x^*$, $f(x) > f(x^*)$.

Usually, unless f has some convexity properties, finding the global minimum is a very difficult task that needs global knowledge of the function f. On the other hand, finding local minima requires only knowledge of the neighborhood. The necessary conditions for local minima can also be easily derived by differential calculus. This explains why in our minimization problem we will try to find a local minimum, instead of a global one.

In order to set up necessary conditions satisfied by local minima, the basic idea is to look around a point using the concept of feasible directions. From a point $x \in \Omega$, a vector d is a feasible direction if there is an $\bar{\alpha} > 0$ such that $x + \alpha d \in \Omega$ for all $\alpha \in [0, \bar{\alpha}]$. We have the following first-order necessary conditions:

Proposition 1.9 ([87])**.** *Let Ω be a subset of $\mathbb{R}^n$ and f be a continuously differentiable function on Ω. If x^* is a local minimum of f over Ω, then for every feasible direction d at x^*, we have*

$$(\nabla f(x^*))^T d \geq 0. \tag{1.4}$$

Conversely, every point that satisfies the condition (1.4) is called a *stationary point*. When x^* is an interior point of Ω, then every vector d is a feasible direction and (1.4) implies $\nabla f(x^*) = 0$.

If Ω is convex, to create all the feasible directions, one can use the vectors $d = x - x^*$, for every $x \in \Omega$. This will generate all the feasible directions at x^*, since from the convexity of Ω, we have

$$x^* + \alpha d = x^* + \alpha(x - x^*) = \alpha x + (1 - \alpha)x^* \in \Omega, \text{ for all } \alpha \in [0, 1].$$

Therefore, a point x^* is said to be a *stationary point* if it satisfies

$$(\nabla f(x^*))^T (x - x^*) \geq 0, \qquad \forall x \in \Omega.$$

For the special case where f and Ω are convex, every local minimum is also a global minimum. Furthermore, the set of all such minima is convex. For more results and implications, see [87].

1.2.3 Karush-Kuhn-Tucker conditions

Let us consider the following constrained optimization problem:

$$\min_{\substack{h_i(x)=0 \\ g_j(x)\leq 0}} f(x),$$

where $h_i(x) = 0$, $(i = 1, \ldots, k)$ are k equality constraints and $g_j(x) \leq 0$ $(j = 1, \ldots, m)$ are m inequality constraints. The following is known as Karush-Kuhn-Tucker necessary conditions (or *KKT conditions*):

Proposition 1.10 ([13]). *Let x^* be the local minimum of the above problem. Suppose that f, h_i and g_j are continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$ and $\nabla h_i(x^*)$ and $\nabla g_j(x^*)$ are linearly independent. Then there exist unique constants μ_i $(i = 1, \ldots, k)$ and λ_j $(j = 1, \ldots, m)$, such that:*

$$\begin{aligned} &\nabla f(x^*) + \textstyle\sum_{i=1}^{k} \mu_i \nabla h_i(x^*) + \sum_{j=1}^{m} \lambda_j \nabla g_j(x^*) = 0, \\ &\lambda_j \geq 0, \ j = 1, \ldots, m \\ &\lambda_j g_j(x^*) = 0, \ j = 1, \ldots, m. \end{aligned}$$

This constrained problem is often written in its associated Lagrange Function:

$$L(x, \mu_i, \ldots, \mu_k, \lambda_1, \ldots, \lambda_m) = f(x) + \sum_{i=1}^{k} \mu_i h_i(x) + \sum_{j=1}^{m} \lambda_j g_j(x)$$

where μ_i $(i = 1, \ldots, k)$ and λ_j $(j = 1, \ldots, m)$ are the same as those in the KKT conditions and are called *Lagrange multipliers*.

1.2.4 Coordinate descent algorithm on a convex set

We briefly describe a method for solving the following problem

$$\min_{x \in \Omega} f(x),$$

where $\Omega \subset \mathbb{R}^n$ is a Cartesian product of closed convex sets $\Omega_1, \Omega_2, \ldots, \Omega_m$, where $\Omega_i \subset \mathbb{R}^{n_i}$ (i = 1,...,m) and $\sum_i n_i = n$. The variable x is also partitioned accordingly as

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix},$$

where $x_i \in \Omega_i$. Algorithm 1 is called the coordinate descent algorithm. If we assume that Step 4 of Algorithm 1 can be solved exactly and

Algorithm 1 Coordinate descent

1: Initialize x_i
2: **repeat**
3: **for** $i = 1$ to m **do**
4: Solve $x_i = \text{argmin}_{\xi \in \Omega_i} f(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_m)$
5: **end for**
6: **until** Stopping condition

the minimum is uniquely attained, then we have the following result. Because this result is extensively used in this thesis, we include here its proof taken from Proposition 2.7.1 in [13].

Theorem 1.11 (Convergence of Coordinate Descent Method)**.** *Suppose that f is a continuously differentiable function over the set Ω described above. Furthermore, suppose that for each i and $x \in \Omega$, the solution of*

$$\min_{\xi \in \Omega_i} f(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_m)$$

is uniquely attained. Let $\{x^k\}$ be the sequence generated by Algorithm 1. Then every limit point is a stationary point.

Proof. Let

$$z_i^k = \left(x_1^{k+1}, \ldots, x_i^{k+1}, x_{i+1}^k, \ldots, x_m^k\right).$$

Step 4 of Algorithm 1 implies

$$f(x^k) \geq f(z_1^k) \geq f(z_2^k) \geq \cdots \geq f(z_{m-1}^k) \geq f(z_m^k), \qquad \forall k. \tag{1.5}$$

Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_m)$ be a limit point of the sequence $\{x^k\}$. Notice that $\bar{x} \in \Omega$ because Ω is closed. Equation (1.5) implies that the sequence $\{f(x^k)\}$ converges to $f(\bar{x})$. It now remains to show that $\bar{x}$ minimizes f over Ω.

Let $\{x^{k_j} \mid j = 0, 1, \ldots\}$ be a subsequence of $\{x^k\}$ that converges to $\bar{x}$. We first show that $\{x_1^{k_j+1} - x_1^{k_j}\}$ converges to zero as $j \to \infty$. Assume the contrary or, equivalently, that $\{z_1^{k_j} - x^{k_j}\}$ does not converges to zero. Let $\gamma^{k_j} = \|z_1^{k_j} - x^{k_j}\|$. By possibly restricting to a subsequence of $\{k_j\}$, we may assume that there exists some $\bar{\gamma} > 0$ such that $\gamma^{k_j} \geq \bar{\gamma}$ for all j. Let $s_1^{k_j} = (z_1^{k_j} - x^{k_j})/\gamma^{k_j}$. Thus, $z_1^{k_j} = x^{k_j} + \gamma^{k_j} s_1^{k_j}$, $\|s_1^{k_j}\| = 1$, and $s_1^{k_j}$ differs from zero only along the first block-component. Notice that $s_1^{k_j}$ belongs to a compact set and therefore has a limit point $\bar{s}_1$. By restricting to a further subsequence of $\{k_j\}$, we assume that $s_1^{k_j}$ converges to $\bar{s}_1$.

Let us fix some $\epsilon \in [0, 1]$. Notice that $0 \geq \epsilon\bar{\gamma} \geq \gamma^{k_j}$. Therefore, $x^{k_j} + \epsilon\bar{\gamma} s_1^{k_j}$ lies on the segment joining x^{k_j} and $x^{k_j} + \gamma^{k_j} s_1^{k_j} = z_1^{k_j}$, and belongs to Ω because Ω is convex. Using the fact that $z_1^{k_j}$ minimizes f over all x that differ from x^{k_j} along the first block-component, we obtain

$$f(z_1^{k_j}) = f(x^{k_j} + \gamma^{k_j} s_1^{k_j}) \leq f(x^{k_j} + \epsilon\bar{\gamma} s_1^{k_j}) \leq f(x^{k_j}).$$

Since $f(x^{k_j})$ converges to $f(\bar{x})$, Equation (1.5) shows that $f(z_1^{k_j})$ also converges to $f(\bar{x})$. We now take the limit as j tends to infinity, to obtain $f(\bar{x}) \leq f(\bar{x} + \epsilon\bar{\gamma}\bar{s}_1) \leq f(\bar{x})$. We conclude that $f(\bar{x}) = f(\bar{x} + \epsilon\bar{\gamma}\bar{s}_1)$, for every $\epsilon \in [0, 1]$. Since $\bar{\gamma}\bar{s}_1 \neq 0$, this contradicts the hypothesis that f is uniquely minimized when viewed as a function of the first block-component. This contradiction establishes that $x_1^{k_j+1} - x_1^{k_j}$ converges to zero. In particular, $z_1^{k_j}$ converges to $\bar{x}$.

From Step 4 of Algorithm 1, we have

$$f(z_1^{k_j}) \leq f(x_1, x_2^{k_j}, \ldots, x_m^{k_j}), \qquad \forall x_1 \in \Omega_1.$$

Taking the limit as j tends to infinity, we obtain

$$f(\bar{x}) \leq f(x_1, \bar{x}_2^{k_j}, \ldots, \bar{x}_m^{k_j}), \qquad \forall x_1 \in \Omega_1.$$

Using Proposition 1.9 over the convex set Ω_1, we conclude that

$$\nabla_1 f(\bar{x})^T (x_1 - \bar{x}_1) \geq 0, \qquad x_1 \in \Omega_1,$$

where $\nabla_i f$ denotes the gradient of f with respect to the component x_i.

Let us now consider the sequence $\{z_1^{k_j}\}$. We have already shown that $z_1^{k_j}$ converges to $\bar{x}$. A verbatim repetition of the preceding argument shows that $x_2^{k_j+1} - x_2^{k_j}$ converges to zero and $\nabla_1 f(\bar{x})^T (x_1 - \bar{x}_1) \geq 0$, for every $x_2 \in \Omega_2$. Continuing inductively, we obtain $\nabla_i f(\bar{x})^T (x_i - \bar{x}_i) \geq 0$, for every $x_i \in \Omega_i$ and for every i. Adding these inequalities, and using the Cartesian product structure of the set Ω, we conclude that $\nabla f(\bar{x})(x - \bar{x}) \geq 0$ for every $x \in \Omega$. □

1.3 Low-rank matrix approximation

Low-rank approximation is a special case of matrix nearness problem [58]. When only a rank constraint is imposed, the optimal approximation with respect to the Frobenius norm can be obtained from the Singular Value Decomposition.

We first investigate the problem without the nonnegativity constraint on the low-rank approximation. This is useful for understanding properties of the approximation when the nonnegativity constraints are imposed but inactive. We begin with the well-known Eckart-Young Theorem.

Theorem 1.12 (Eckart-Young). *Let $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) have the singular*

value decomposition

$$A = P\Sigma Q^T,\ \Sigma = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ *are the singular values of* A *and where* $P \in \mathbb{R}^{m\times m}$ *and* $Q \in \mathbb{R}^{n\times n}$ *are orthogonal matrices. Then for* $1 \leq r \leq n$*, the matrix*

$$A_r = P\Sigma_r Q^T,\ \Sigma_r = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_r & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}$$

is a global minimizer of the problem

$$\min_{B\in\mathbb{R}^{m\times n}\ rank(B)\leq r} \frac{1}{2}\|A - B\|_F^2 \tag{1.6}$$

and its error is

$$\frac{1}{2}\|A - B\|_F^2 = \frac{1}{2}\sum_{i=r+1}^{n} \sigma_i^2.$$

Moreover, if $\sigma_r > \sigma_{r+1}$ *then* A_r *is the unique global minimizer.*

The proof and other implications can be found for instance in [50]. The columns of P and Q are called singular vectors of A, in which vectors corresponding to the largest singular values are referred to as the dominant singular vectors.

Let us now look at the following modified problem

$$\min_{X\in\mathbb{R}^{m\times r}\ Y\in\mathbb{R}^{n\times r}} \frac{1}{2}\|A - XY^T\|_F^2, \tag{1.7}$$

where the rank constraint is implicit in the product XY^T since the dimensions of X and Y guarantee that $rank(XY^T) \leq r$. Conversely, every matrix of rank less than r can be trivially rewritten as a product XY^T, where $X \in \mathbb{R}^{m \times r}$ and $Y \in \mathbb{R}^{n \times r}$. Therefore Problems (1.6) and (1.7) are equivalent. But even when the product $A_r = XY^T$ is unique, the pairs (XR^T, YR^{-1}) with R invertible, yield the same product XY^T. In order to avoid this, we can always choose X and Y such that

$$X = PD^{\frac{1}{2}} \text{ and } Y = QD^{\frac{1}{2}}, \tag{1.8}$$

where $P^TP = I_{r \times r}$, $Q^TQ = I_{r \times r}$ and D is $r \times r$ nonnegative diagonal matrix. Doing this is equivalent to computing a compact SVD decomposition of the product $A_r = XY^T = PDQ^T$.

As usual for optimization problems, we calculate the gradient with respect to X and Y and set them equal to 0.

$$\nabla_X = XY^TY - AY = 0 \qquad \nabla_Y = YX^TX - A^TX = 0. \tag{1.9}$$

If we then premultiply A^T with ∇_X and A with ∇_Y, we obtain

$$(A^TA)Y = (A^TX)Y^TY \qquad (AA^T)X = (AY)X^TX. \tag{1.10}$$

Replacing $A^TX = YX^TX$ and $AY = XY^TY$ into (1.10) yields

$$(A^TA)Y = YX^TXY^TY \qquad (AA^T)X = XY^TYX^TX. \tag{1.11}$$

Replacing (1.8) into (1.11) yields

$$(A^TA)QD^{\frac{1}{2}} = QDP^TPDQ^TQD^{\frac{1}{2}} \text{ and } (AA^T)PD^{\frac{1}{2}} = PDQ^TQDP^TPD^{\frac{1}{2}}.$$

When D is invertible, this finally yields

$$(A^TA)Q = QD^2 \text{ and } (AA^T)P = PD^2.$$

This shows that the columns of P and Q are singular vectors and $D_{ii}{}'s$ are nonzero singular values of A. Notice that if D is singular, one can throw away the corresponding columns of P and Q and reduce it to a smaller-rank approximation with the same properties. Without loss of generality, we therefore can focus on approximations of Problem (1.7) which are of exact rank r. We can summarize the above reasoning in the following theorem.

Theorem 1.13. *Let $A \in \mathbb{R}^{m\times n}$ ($m > n$ and $rank(A) = t$). If A_r ($1 \leq r \leq t$) is a rank r stationary point of Problem 1.7, then there exists two orthogonal matrices $P \in \mathbb{R}^{m\times m}$ and $Q \in \mathbb{R}^{n\times n}$ such that:*

$$A = P\hat{\Sigma}Q^T \text{ and } A_r = P\hat{\Sigma}_r Q^T$$

where

$$\hat{\Sigma} = \begin{pmatrix} \hat{\sigma}_1 & 0 & \dots & 0 \\ 0 & \hat{\sigma}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \hat{\sigma}_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}, \quad \hat{\Sigma}_r = \begin{pmatrix} \hat{\sigma}_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \hat{\sigma}_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \hat{\sigma}_r & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}$$

and the $\hat{\sigma}_i's$ are unsorted singular values of A. Moreover, the approximation error is:

$$\frac{1}{2}\|A - A_r\|_F^2 = \frac{1}{2}\sum_{i=r+1}^{t} \hat{\sigma}_i^2.$$

This result shows that, if the singular values are all different, there are $\frac{n!}{r!(n-r)!}$ possible stationary points A_r. When there are multiple singular values, there will be infinitely many stationary points A_r since there are infinitely many singular subspaces. The next result will identify the minima among all stationary points. Other stationary points are saddle points whose every neighborhood contains both smaller and higher points.

Theorem 1.14. *The only minima of Problem 1.7 are given by Theorem 1.12 and are global minima. All other stationary points are saddle points.*

Proof. Let us assume that A_r is a stationary point given by Theorem 1.13 but not by Theorem 1.12. Then there always exists a permutation of the columns of P and Q, and of the diagonal elements of $\hat{\Sigma}$ and $\hat{\Sigma}_r$ such that $\hat{\sigma}_{r+1} > \hat{\sigma}_r$. We then construct two points in the ϵ-neighborhood of A_r that yield an increase and a decrease, respectively, of the distance

measure. They are obtained by taking:

$$\overline{\Sigma}_r(\epsilon) = \begin{pmatrix} \hat{\sigma}_1 + \epsilon & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots \\ 0 & \dots & \hat{\sigma}_r & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \qquad \overline{A}_r(\epsilon) = P\overline{\Sigma}_r(\epsilon)Q^T$$

and

$$\underline{\Sigma}_r(\epsilon) = \begin{pmatrix} \hat{\sigma}_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & \hat{\sigma}_r & \epsilon\sqrt{\hat{\sigma}_r} & \vdots & 0 \\ 0 & \dots & \epsilon\sqrt{\hat{\sigma}_r} & \epsilon^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \qquad \underline{A}_r(\epsilon) = P\underline{\Sigma}_r(\epsilon)Q^T.$$

Clearly $\overline{A}_r(\epsilon)$ and $\underline{A}_r(\epsilon)$ are of rank r. Evaluating the distance measure yields

$$\begin{aligned} \|A - \underline{A}_r(\epsilon)\|_F^2 &= 2\hat{\sigma}_r\epsilon^2 + (\hat{\sigma}_{r+1} - \epsilon^2)^2 + \sum_{i=r+2}^{t} \hat{\sigma}_i^2 \\ &= \epsilon^2[\epsilon^2 - 2(\hat{\sigma}_{r+1} - \hat{\sigma}_r)] + \sum_{i=r+1}^{t} \hat{\sigma}_i^2 \\ &< \sum_{i=r+1}^{t} \hat{\sigma}_i^2 = \|A - A_r\|_F^2 \end{aligned}$$

for all $\epsilon \in (0, \sqrt{2(\hat{\sigma}_{r+1} - \hat{\sigma}_r)})$ and

$$\|A - \overline{A}_r(\epsilon)\|_F^2 = \epsilon^2 + \sum_{i=r+1}^{t} \hat{\sigma}_i^2 > \sum_{i=r+1}^{t} \hat{\sigma}_i^2 = \|A - A_r\|_F^2$$

for all $\epsilon > 0$. Hence, for an arbitrarily small positive ϵ, we obtain

$$\|A - \underline{A}_r(\epsilon)\|_F^2 < \|A - A_r\|_F^2 < \|A - \overline{A}_r(\epsilon)\|_F^2$$

which shows that A_r is a saddle point of the distance measure. □

When we add a nonnegativity constraint in the next section, the results of this section will help to identify stationary points at which all the nonnegativity constraints are inactive.

2
NONNEGATIVE MATRIX FACTORIZATION

This chapter is a presentation of the Nonnegative Matrix Factorization problem. It consists of the formulation of the problem, the description of the solutions and some observations. It gives the basics for the rest of this thesis. Some observations are studied more carefully in other chapters.

One could argue that the name Nonnegative Matrix Factorization maybe misleading in some cases and that Nonnegative Matrix Approximation should be used instead. The term "Factorization" maybe understood as an exact decomposition such as Cholesky decomposition, LU decomposition, etc. where the input matrix is exactly factorized as a product of other matrices. However, "Nonnegative Matrix Factorization" has become so popular that it does stand for the problem of approximating a nonnegative matrix by a product of two nonnegative matrices. We continue to use this term and refer to Exact Nonnegative Matrix Factorization for the exact case.

2.1 Problem statement

Nonnegative Matrix Factorization was first introduced by Paatero and Tapper in [97]. But it has gained popularity by the works of Lee and Seung [80]. They argue that the nonnegativity is important in human perceptions and also give two simple algorithms for finding a nonnegative representation for nonnegative data. Given an $m \times n$ nonnegative

matrix A (i.e. $A_{ij} \geq 0$) and a reduced rank r ($r < \min(m, n)$), the nonnegative matrix factorization problem consists in finding two nonnegative matrices $U \in \mathbb{R}_+^{m \times r}$ and $V \in \mathbb{R}_+^{n \times r}$ that approximate A, i.e.

$$A \approx UV^T.$$

Looking at the columns of A, one sees that each of them is approximated by a conic combination of r nonnegative basis vectors that are the columns of U

$$A_{:i} \approx \sum_{j=1}^{r} V_{ij} U_{:j}.$$

We can consider the columns of U as the basis of the cone $\mathbf{U}$ completely contained inside the nonnegative orthant. And each column of A is approximated by an element of $\mathbf{U}$, typically the closest element to the column. We can also exchange the role of U and V to point out that each row of A is approximated by an element of $\mathbf{V}$, typically the closest element to the row, where $\mathbf{V}$ is the cone generated by the column of V.

There are several ways to quantify the difference between the data matrix A and the model matrix UV^T. But the most used measure is the Frobenius norm:

$$F(A, UV^T) = \frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{n}\left(A_{ij} - [UV^T]_{ij}\right)^2$$

which is also referred to as the Euclidean Distance.

Suppose that U is fixed, the function $F(U, V) = \frac{1}{2}\|A - UV^T\|_F^2$ can be seen as a composition of the Frobenius norm and a linear transformation of V. Therefore, F is convex in V. Likewise, if V is fixed, F is convex on U.

Throughout this thesis, the nonnegative matrix factorization problem will be studied with a bias to the Frobenius norm. The main problem is the following:

Problem 2.1 (Nonnegative matrix factorization - NMF)**.** Given a $m \times n$ nonnegative matrix A and an integer $r < \min(m, n)$, solve

$$\min_{U \in \mathbb{R}_+^{m \times r}\ V \in \mathbb{R}_+^{n \times r}} \frac{1}{2}\|A - UV^T\|_F^2.$$

Where r is called the reduced rank. From now on, m and n will be used to denote the size of the target matrix A and r is the reduced rank of a factorization.

We rewrite the nonnegative matrix factorization as a standard nonlinear optimization problem:

$$\min_{-U\leq 0\ -V\leq 0} \frac{1}{2}\|A - UV^T\|_F^2.$$

The associated Lagrangian function is

$$L(U, V, \mu, \nu) = \frac{1}{2}\|A - UV^T\|_F^2 - \mu \circ U - \nu \circ V,$$

where μ and ν are two *matrices* of the same size of U and V, respectively, containing the Lagrange multipliers associated with the nonnegativity constraints $U_{ij} \geq 0$ and $V_{ij} \geq 0$. Then the Karush-Kuhn-Tucker conditions for the nonnegative matrix factorization problem say that if (U, V) is a local minimum, then there exist $\mu_{ij} \geq 0$ and $\nu_{ij} \geq 0$ such that:

$$U \geq 0 \quad , \qquad V \geq 0, \tag{2.1}$$
$$\nabla L_U = 0 \quad , \qquad \nabla L_V = 0, \tag{2.2}$$
$$\mu \circ U = 0 \quad , \qquad \nu \circ V = 0. \tag{2.3}$$

Developing (2.2) we have:

$$AV - UV^TV - \mu = 0, \quad A^TU - VU^TU - \nu = 0$$

or

$$\mu = -(UV^TV - AV), \quad \nu = -(VU^TU - A^TU).$$

Combining this with $\mu_{ij} \geq 0$, $\nu_{ij} \geq 0$ and (2.3) gives the following conditions:

$$U \geq 0 \quad , \qquad V \geq 0, \tag{2.4}$$
$$\nabla F_U = UV^TV - AV \geq 0 \quad , \qquad \nabla F_V = VU^TU - A^TU \geq 0, \tag{2.5}$$
$$U \circ (UV^TV - AV) = 0 \quad , \qquad V \circ (VU^TU - A^TU) = 0, \tag{2.6}$$

where the corresponding Lagrange multipliers for U and V are also the gradient of F with respect to U and V.

Since the Euclidean distance is not convex with respect to both variables U and V at the same time, these conditions are only necessary. This is implied because of the existence of saddle points and maxima. We then call all the points that satisfy the above conditions, the stationary points.

Definition 2.2 (NMF stationary point)**.** We call (U, V) a stationary point of the NMF Problem if and only if U and V satisfy the KKT conditions (2.4), (2.5) and (2.6).

Alternatively, a stationary point (U, V) of the NMF problem can also be defined by using the condition in Proposition 1.9 on the convex sets $\mathbb{R}_+^{m\times r}$ and $\mathbb{R}_+^{n\times r}$, that is

$$\left\langle \begin{pmatrix} \nabla F_U \\ \nabla F_V \end{pmatrix}, \begin{pmatrix} X - U \\ Y - V \end{pmatrix} \right\rangle \geq 0, \qquad \forall\, X \in \mathbb{R}_+^{m\times r},\ Y \in \mathbb{R}_+^{n\times r}, \tag{2.7}$$

which can be shown to be equivalent to the KKT conditions (2.4), (2.5) and (2.6). Indeed, it is trivial that the KKT conditions imply (2.7). And by carefully choosing different values of X and Y from (2.7), one can easily prove that the KKT conditions hold.

Representing a rank-k matrix by the product UV^T is, in fact, rarely used due to the loss of the uniqueness of the presentation. Because a nonnegative factorization is, by definition, in this form, the rest of the section tries to fix the uniqueness problem and to establish some simple relations between the approximations.

Let us consider the simplest nonnegative factorization problem where the matrix A is just a scalar a. The problem (of rank one) is then to find two scalar x and y whose product approximate a. Problem 2.1 admits only exact approximations $(a - xy)^2 = 0$, and we have infinite number of solutions given by the graph $xy = a$ (Figure 2.1).

If we impose the unit norm condition on x (i.e. $\|x\| = 1$), then for this particular case, there will be only one solution ($x = 1$ and $y = a$).

To extend this scaling technique to higher dimensions, we continue to constrain the first factor U to have unit-norm columns. But this no longer guarantees the uniqueness of the approximations. Moreover, it is not easy to determine when and how the uniqueness is obtainable.

Two approximations (U_1, V_1) and (U_2, V_2) are said to be equivalent iff they yield the same product, i.e. $U_1V_1^T = U_2V_2^T$.

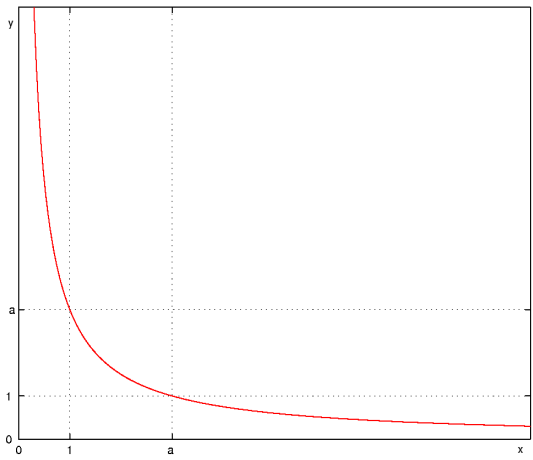


Figure 2.1: Graph $a = xy$

From a stationary point (U, V), if we can find an invertible matrix S such that $\hat{U} = US \geq 0$ and $\hat{V} = V(S^{-1})^T \geq 0$, have we constructed an equivalent stationary point $(\hat{U}, \hat{V})$? By plugging the matrices $\hat{U}$ and $\hat{V}$ into the KKT conditions, we can see that the answer is not always easy. Indeed, if $\hat{U}$ and $\hat{V}$ are made to be nonnegative, then according to the KKT conditions (2.5) and (2.6), we should also have:

$$\begin{aligned}
&(UV^TV - AV)(S^{-1})^T \geq 0,\\
&(VU^TU - A^TU)S \geq 0,\\
&(US) \circ \left[(UV^TV - AV)(S^{-1})^T\right] = 0,\\
&(V(S^{-1})^T) \circ \left[(VU^TU - A^TU)S\right] = 0.
\end{aligned}$$

In particular, for a permutation matrix S, these conditions are easily checked. In this case, all the columns of U and V are retained in $\hat{U}$ and $\hat{V}$, but in a permuted order, which generate essentially the same point. Note that S can not be a nonnegative *monomial matrix* (i.e. matrix created from a permutation matrix by replacing some elements equal to 1 by other positive numbers), since $\hat{U}$ and U are constrained to have unit-norm columns.

For general S, the study of the uniqueness of the stationary point is no longer easy and might be treated only on a case-by-case basis. For example, we remark that at some (stationary) points (U, V), S must

be a permutation matrix, otherwise, the nonnegativity of $(\hat{U}, \hat{V})$ will not be met. This implies that we can not generate other equivalent approximations. The following result helps to identify a class of them.

Lemma 2.3. *If U and V both contain a $r \times r$ monomial matrix, then S can only be the permutation matrices.*

Proof. The assumption implies that we can select r rows of U to form an $r \times r$ monomial matrix U_r and r rows of V to form an $r \times r$ monomial matrix V_r. Then the nonnegativity constraint implies

$$U_r S \geq 0 \text{ and } V_r (S^{-1})^T \geq 0.$$

Since, U_r and V_r are nonnegative monomial matrices, both S and S^{-1} must be nonnegative. This is only possible when S is a monomial matrix[1]. Moreover, U and US are constrained to have unit-norm columns, S can only be permutation matrices. □

Another way to consider the set of *equivalent stationary points* is to identify them by all the possible exact factorization of the matrix $\hat{A} = UV^T$ where (U, V) is one known entry of the set. But there is no easy method to construct this set.

A better representation of the stationary point is similar to the singular value decomposition. We can use a triplet (U, D, V) to represent a NMF stationary point. So, instead of solving Problem 2.1, we solve the following problem:

$$(u_i^*, d_i^*, v_i^*)_{i=1}^r = \underset{\substack{u_i \geq 0 \; u_i^T u_i = 1 \\ v_i \geq 0 \; v_i^T v_i = 1 \\ d_i \geq 0}}{\operatorname{argmin}} \|A - \sum_{i=1}^{r} d_i u_i v_i^T\|_2^2,$$

[1] We have $S \geq 0$ and $S^{-1} \geq 0$ and the off-diagonal elements of $SS^{-1} = I_r$ and $S^{-1}S = I_r$ are zero. As a consequence, if $S_{ij} > 0$, we can conclude that $S^{-1}_{jk} = 0$ and $S^{-1}_{li} = 0$ for $k \neq i$ and $l \neq j$. Because S^{-1} is invertible hence can not contain zero rows and columns, S^{-1}_{ji} is the only positive element on the j^{th} row and i^{th} column of S^{-1}. Reciprocally, $S^{-1}_{ji} > 0$ implies S_{ij} is the only positive element on the i^{th} row and j^{th} column of S. Since S can not contain zero rows and columns, repeating the above reasoning through all the nonzero elements of S yields the desired result.

or in matrix representation, U and V is nonnegative matrices with unit-norm columns and D is a nonnegative diagonal matrix. The matrix A is then approximated by UDV^T. With this, we can also sort the components in decreasing order of the value of D_{ii} (i.e. $D_{11} \geq D_{22} \geq \cdots \geq D_{rr}$). This helps to compare equivalent solutions.

In Chapter 4, we use this representation to design our iterative algorithm and point out its advantages.

2.2 Solution

There are two values of reduced rank r for which we can trivially identify the global solution which are $r = 1$ and $r = \min(m, n)$. For $r = 1$, a pair of dominant singular vectors are a global minimizer. And for $r = \min(m, n)$, $(U = A, V = I)$ is a global minimizer. Since most of existing methods for the nonnegative matrix factorization are descent algorithms, we should pay attention to all local minimizers. For the rank-one case, they can easily be characterized.

2.2.1 Rank one case

The rank-one NMF problem of a nonnegative matrix A can be rewritten as

$$\min_{u \in \mathbb{R}^m_+ \; v \in \mathbb{R}^n_+} \frac{1}{2} \|A - uv^T\|_F^2 \tag{2.8}$$

and a complete analysis can be carried out. It is well known that any pair of nonnegative Perron vectors of AA^T and A^TA yields a global minimizer of this problem, but we can also show that the *only* stationary points of (2.8) are given by such vectors. The following theorem excludes the case where $u = 0$ and/or $v = 0$.

Theorem 2.4. *The pair (u, v) is a local minimizer of (2.8) if and only if u and v are nonnegative eigenvectors of AA^T and A^TA respectively of the eigenvalue $\sigma = \|u\|_2^2 \|v\|_2^2$.*

Proof. The *if part* easily follows from Theorem 1.13. For the *only if part* we proceed as follows. Without loss of generality, we can permute the rows and columns of A such that the corresponding vectors u and v

are partitioned as $(u_+ \; 0)^T$ and $(v_+ \; 0)^T$ respectively, where $u_+, v_+ > 0$. Partition the corresponding matrix A conformably as follows

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

then from (2.5) we have

$$\begin{pmatrix} u_+ v_+^T & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v_+ \\ 0 \end{pmatrix} - \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} v_+ \\ 0 \end{pmatrix} \geq 0$$

and

$$\begin{pmatrix} v_+ u_+^T & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_+ \\ 0 \end{pmatrix} - \begin{pmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{pmatrix} \begin{pmatrix} u_+ \\ 0 \end{pmatrix} \geq 0$$

implying that $A_{21} v_+ \leq 0$ and $A_{12}^T u_+ \leq 0$. Since A_{21} , $A_{12} \geq 0$ and $u_+, v_+ > 0$, we can conclude that $A_{12} = 0$ and $A_{21} = 0$. Then from (2.6) we have:

$$u_+ \circ (\|v_+\|_2^2 u_+ - A_{11} v_+) = 0 \text{ and } v_+ \circ (\|u_+\|_2^2 v_+ - A_{11}^+ u_+) = 0.$$

Since $u_+, v_+ > 0$, we have:

$$\|v_+\|_2^2 u_+ = A_{11} v_+ \text{ and } \|u_+\|_2^2 v_+ = A_{11}^T u_+$$

or

$$\|u_+\|_2^2 \|v_+\|_2^2 u_+ = A_{11} A_{11}^T u_+ \text{ and } \|u_+\|_2^2 \|v_+\|_2^2 v_+ = A_{11}^T A_{11} v_+.$$

Setting $\sigma = \|u_+\|_2^2 \|v_+\|_2^2$ and using the block *diagonal* structure of A yields the desired result. □

Theorem 2.4 guarantees that all stationary points of the rank-one case are nonnegative singular vectors of a submatrix of A. These results imply that a global minimizer of the rank-one NMF can be calculated correctly based on the largest singular value and corresponding singular vectors of the matrix A.

For ranks other than 1 and $\min(m, n)$, there are no longer trivial stationary points. In the next section, we try to derive some simple characteristics of the local minima of the nonnegative matrix factorization.

2.2.2 Characteristics of local minima

The KKT conditions (2.6) help to characterize the stationary points of the NMF problem. Summing up all the elements of one of the conditions (2.6), we get:

$$\begin{aligned} 0 &= \sum_{ij} \Big(U \circ (UV^T V - AV) \Big)_{ij} \\ &= \Big\langle U, UV^T V - AV \Big\rangle \\ &= \Big\langle UV^T, UV^T - A \Big\rangle . \end{aligned} \tag{2.9}$$

From that, we have some simple characteristics of the NMF solutions:

Theorem 2.5. *Let (U, V) be a stationary point of the NMF problem, then $UV^T \in \mathcal{B}\left(\frac{A}{2}, \frac{1}{2}\|A\|_F\right)$, the ball centered at $\frac{A}{2}$ and with radius $= \frac{1}{2}\|A\|_F$.*

Proof. From (2.9) it immediately follows that

$$\left\langle \frac{A}{2} - UV^T, \frac{A}{2} - UV^T \right\rangle = \left\langle \frac{A}{2}, \frac{A}{2} \right\rangle$$

which implies

$$UV^T \in \mathcal{B}\left(\frac{A}{2}, \frac{1}{2}\|A\|_F \right).$$

□

Theorem 2.6. *Let (U, V) be a stationary of the NMF problem, then*

$$\frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}(\|A\|_F^2 - \|UV^T\|_F^2)$$

.

Proof. From (2.9), we have $\langle UV^T, A \rangle = \langle UV^T, UV^T \rangle$. Therefore,

$$\begin{aligned} \frac{1}{2}\left\langle A - UV^T, A - UV^T \right\rangle &= \frac{1}{2}(\|A\|_F^2 - 2\left\langle UV^T, A \right\rangle + \|UV^T\|_F^2) \\ &= \frac{1}{2}(\|A\|_F^2 - \|UV^T\|_F^2). \end{aligned}$$

□

Theorem 2.6 also suggests that at a stationary point (U, V) of the NMF problem, we should have $\|A\|_F^2 \geq \|UV^T\|_F^2$. This norm inequality can be also found in [25] for less general cases where we have $\nabla F_U = 0$ and $\nabla F_V = 0$ at a stationary point. For this particular class of NMF stationary point, all the nonnegativity constraints on U and V are inactive. And all such stationary points are also stationary points of the unconstrained problem, characterized by Theorem 1.13.

We have seen in Theorem 1.13 that, for the unconstrained least-square problem the only stable stationary points are in fact global minima. Therefore, if the stationary points of the constrained problem are inside the nonnegative orthant (i.e. all constraints are inactive), we can then probably reach the global minimum of the NMF problem. This can be expected because the constraints may no longer prohibit the descent of the update.

The equality of $\|A\|_F^2 \geq \|UV^T\|_F^2$ implied by Theorem 2.6 is only obtained when we have an exact factorization (i.e $A = UV^T$) and it will be the subject of the next section.

Let A_r be the optimal rank-r approximation of a nonnegative matrix A, which we obtain from the singular value decomposition, as indicated in Theorem 1.13. Then we can easily construct its nonnegative part $[A_r]_+$, which is obtained from A_r by just setting all its negative elements equal to zero. This is in fact the closest matrix in the cone of nonnegative matrices to the matrix A_r, in the Frobenius norm (in that sense, it is its projection on that cone). We now derive some bounds for the error $\|A - [A_r]_+\|_F$.

Theorem 2.7. *Let A_r be the best rank r approximation of a nonnegative matrix A, and let $[A_r]_+$ be its nonnegative part, then*

$$\|A - [A_r]_+\|_F \leq \|A - A_r\|_F.$$

Proof. This follows easily from the convexity of the cone of nonnegative matrices. Since both A and $[A_r]_+$ are nonnegative and since $[A_r]_+$ is the closest matrix in that cone to A_r we immediately obtain the inequality

$$\|A - A_r\|_F^2 \geq \|A - [A_r]_+\|_F^2 + \|A_r - [A_r]_+\|_F^2 \geq \|A - [A_r]_+\|_F^2$$

from which the result readily follows. □

If we now compare this bound with the nonnegative approximations then we obtain the following inequalities. Let $U_* V_*^T$ be an optimal nonnegative rank r approximation of A and let UV^T be any stationary point of the KKT conditions for a nonnegative rank r approximation, then we have :

$$\|A - [A_r]_+|_F^2 \leq \|A - A_r\|_F^2 = \sum_{i=r+1}^{n} \sigma_i^2 \leq \|A - U_* V_*^T\|_F^2 \leq \|A - UV^T\|_F^2.$$

2.3 Exact factorization and nonnegative rank

In this section, we will take a brief look at a stricter problem where we are interested only in the solutions where the objective function is zero. This means that the matrix A is exactly factorized by UV^T (i.e. $A = UV^T$) with the same nonnegativity constraints on the factors. The smallest value of r, the inner rank of the factorization UV^T, that factorizes correctly A is called the nonnegative rank of A, denoted by $rank^+_{UV^T}(A)$. In [33], a nice treatment of the problem is carried out.

The existence of an exact factorization of inner rank r is equivalent to determining $rank^+_{UV^T}(A)$. For any $r > rank^+_{UV^T}(A)$, we can trivially construct an exact nonnegative factorization from the factorization UV^T of inner rank r by adding zero columns to the factors U and V.

For the nonnegative rank, the following results are well known and can be found in [33]. First, an upper bound and a lower bound of this number are easily computed.

Lemma 2.8. *Let $A \in \mathbb{R}_+^{m \times n}$. Then*

$$rank(A) \leq rank^+_{UV^T}(A) \leq \min(m, n).$$

Proof. Since we can not construct the same matrix with lower rank then the first inequality holds. The second comes from one of the trivial factorizations $I_m A$ and $A I_n$. □

In certain cases, the first equality holds. For the rank-one nonnegative matrix A, we know that it can be represented by uv^T, where u and v are nonnegative. This implies that $rank^+_{UV^T}(A) = rank(A) = 1$. It is still true for a rank-two matrix, which is proved in [33] and [3].

Lemma 2.9. *Let $A \in \mathbb{R}_+^{m\times n}$ where $rank(A) = 2$. Then*

$$rank_{UV^T}^{+}(A) = 2.$$

Proof. Since $A \geq 0$, the cone spanned by the columns of A is a convex polyhedral cone contained in the nonnegative orthant. Moreover, $rank(A) = 2$ implies that the cone is contained in a two dimensional linear subspace. Therefore its spanning set, i.e. the columns of A, can be reduced to only two vectors called u_1 and u_2. Every column of A is then represented by

$$A_{:i} = V_{1i}u_1 + V_{2i}u_2, \quad \text{with } V_{1i}, V_{2i} \geq 0.$$

Creating $U = (u_1\ u_2)$ and $V = \{V_{ij}\}$ gives the desired rank-two nonnegative factorization UV^T. □

The two spanning vectors u_1 and u_2 in the proof of the preceding lemma are indeed a pair of columns of A that has the largest angle between them, i.e.

$$(u_1\ u_2) = \underset{A_{:i}\ A:j}{\operatorname{argmin}} \frac{A_{:i}^T A_{:j}}{\|A_{:i}\|\|A_{:j}\|}.$$

V is computed by solving a least square, which yields

$$V = A^T U(U^T U)^{-1} \geq 0.$$

So far, we have seen that when $rank(A)$ is 1, 2 or $min(m, n)$, we can construct an exact nonnegative matrix factorization with the same rank. For matrices with other ranks, determining the nonnegative rank is very difficult. Indeed, Vavasis in [121] has recently proved the *NP-hardness* of the nonnegative matrix factorization. Therefore, all algorithms for solving the exact problem are expected to have a non polynomial complexity. In [116], a method is proposed to create nonnegative matrix factorization via extremal polyhedral cones. Another possibility is using the quantifier elimination algorithms [113] to check for the feasibility of factoring a nonnegative matrix A by a nonnegative factorization of inner rank less than r. All these algorithms are finite. One such algorithm is given by Renegar [101] and was used in the nonnegative rank problem

in [33]. Recently, the same method has been applied to the completely positive rank [9] (cfr. Chapter 7). This method is quite generic and can be applied to other factorizations in this thesis. Here, we describe briefly how to derive the computational complexity bound for a nonnegative factorization.

Consider a first-order formula over the reals having the form

$$(Q_1 x^{[1]} \in \mathbb{R}^{n_1}) \dots (Q_\omega x^{[\omega]} \in \mathbb{R}^{n_\omega}) P(y, x^{[1]}, \dots, x^{[\omega]}), \tag{2.10}$$

where the quantifiers $Q_k \in \{\forall, \exists\}$, the vector y contains n_0 free variables (unquantified) and P is a boolean function constructed from M atom *true-false* expressions

$$g_i(y, x^{[1]}, \dots, x^{[\omega]}) \; \Delta_i \; 0, \quad i = 1, \dots, M$$

with g_i are polynomials of degree less than d and comparison operators $\Delta_i \in \{<, \leq, =, \neq, \geq, >\}$. Then the Renegar algorithm requires at most $(Md)^{2^{O(\omega)} \prod_k n_k}$ multiplications and additions and at most $(Md)^{O(\sum_k n_k)}$ evaluations of P.

These complexity bounds are derived from the known constants ω, n_0, n_1, M and d that can be easily computed for the standard nonnegative matrix factorization in the following lemma.

Lemma 2.10 ([101]). *The Renegar algorithm requires at most $(6mn)^{2^{O(1)} m^2 n^2}$ multiplications and additions and at most $(6mn)^{O(mn)}$ evaluations of P to determine the feasibility of factorizing A by a nonnegative factorization UV^T of inner rank r.*

Proof. We need to eliminate the quantifiers of the following formula:

$$\left((U^T, V^T) \in \mathbb{R}^{r \times (m+n)}\right) P(A, (U^T, V^T))$$

where $P(A, (U^T, V^T)) =$

$$\left(\bigwedge_{ij} \left(\sum_k U_{ik} V_{jk} = A_{ij}\right)\right) \wedge \left(\bigwedge_{ik} (U_{ik} \geq 0)\right) \wedge \left(\bigwedge_{jk} (V_{jk} \geq 0)\right).$$

This configuration gives: $\omega = 1$, $n_0 = mn$, $n_1 = r(m+n) \leq 2mn$, $M = mn + r(m+n) \leq 3mn$ and $d = 2$. And the bounds follow. □

The result implies that the problem of determining the nonnegative rank can be solved in finite time by looping through $r = 1, 2, \ldots,$ $\min(m, n)$, since the upper bound of the nonnegative rank is $\min(m, n)$.

The above lemma can be easily extended for other nonnegative matrix factorizations such as: multilayer nonnegative matrix factorization and nonnegative tensor matrix factorization in the next section, symmetric and semi-symmetric matrix factorization in Chapter 7. For each problem, $\omega = 1$, n_0 is equal to the number of elements of the target matrix (or tensor), n_1 is the total number of elements of all the factors, $M = n_0 + n_1$ and d is equal to the number of factors. Simple countings then yield upper complexity bounds for Renegar algorithm for each feasibility problem.

Similar to the nonnegative rank, the completely-positive rank $(rank_{UU^T}(A))$ and the semi-symmetric nonnegative rank $(rank_{USU^T}(A))$ (crf. Chapter 7) can be also computed in finite time using the Renegar algorithm. This is due to the existence of an upper bound on these ranks.

2.4 Extensions of nonnegative matrix factorization

The essence of the nonnegative matrix factorization is to represent nonnegative data by a nonnegative combination of nonnegative basis vectors, usually called parts. To enlarge the representing capability of the method, improvements are made on how these bases are combined and on the structure of the bases. More specifically, in the standard nonnegative matrix factorization, each data vector $A_{:j} \in \mathbb{R}^n_+$ is represented by

$$A_{:j} = \sum_i V_{ji} U_{:i}, \text{ with } V_{ji} \in \mathbb{R}_+ \text{ and } U_{:i} \in \mathbb{R}^n_+.$$

We can list here two constructions of $U_{:i}$ that may improve the performance of the nonnegative matrix factorization.

2.4.1 Multilayer nonnegative matrix factorization

We can assume that the $U_{:i}$ is approximated based on another set of bases $X_{:t}$. Again, each $U_{:i}$ is constructed by a nonnegative combination of $X_{:i}$'s. So we can write

$$U \approx XX_1$$

where X and X_1 are nonnegative. With the same reasoning, one can assume that $X_{:i}$'s are not the primitives and constructed by another set of bases $[X_1]_{:i}$'s, and so on. This gives the formulation of the multilayer nonnegative matrix factorization:

Problem 2.11 (Multilayer nonnegative matrix factorization)**.**

$$\min_{X_i \geq 0 \; V \geq 0} \frac{1}{2} \|A - X_0 X_1 \dots X_k V^T\|_F^2,$$

where A is the nonnegative data matrix and V and X_i's are nonnegative matrices of compatible sizes.

This problem was studied in a number of works, e.g. [36], [29]. Another related problem is the *Archetypal Analysis* [35], where the above problem is restricted to only three layers, wherein the first layer consists of the data themselves. Each data column is approximated by a convex combination of a set of archetypes that are, in turn, convex combinations of the data columns. The problem to be solved is the following:

$$\min_{\substack{X \geq 0 \; V \geq 0 \\ X^T \mathbf{1} = \mathbf{1} \; V \mathbf{1} = \mathbf{1}}} \frac{1}{2} \|A - (AX) V^T\|_F^2,$$

where each column of AX is an archetype.

For the multilayer nonnegative matrix factorization, one can use algorithms proposed in [36], [29] or the algorithm proposed in Section 4.6.1 to construct an approximation.

2.4.2 Nonnegative Tensor Factorization

Data is, by its nature, not only restricted to nonnegative vectors, i.e. one-dimensional data. Each data can also be points in higher dimensions, for example $m \times n$ nonnegative matrices. And the additional model can be adapted to handle such data

$$A_j \approx \sum_i V_{ji} U_i,$$

where $V_{ji} \geq 0$ and A_j, U_i's $\in \mathbb{R}_+^{m \times n}$.

If we further restricted U_i's to a nonnegative combination of some rank-one nonnegative matrices represented by $x_j y_j^T$ where $x_j \in \mathbb{R}_+^m$ and $y_j \in \mathbb{R}_+^n$. Then the problem of finding x_j, y_j and V_{ij} from A_i's is an example of the following Nonnegative Tensor Factorization problem:

Problem 2.12 (Nonnegative Tensor Factorization)**.**

$$\min_{u_{ij} \in \mathbb{R}_+^{n_i}} \frac{1}{2} \| A - \sum_{j=1}^{r} u_{1j} \star u_{2j} \star \ldots \star u_{dj} \|_F^2$$

where $A \in \mathbb{R}_+^{n_1 \times n_2 \times \ldots \times n_d}$ and $a \star b$ stands for the outer product between two vectors or tensors a and b.

A algorithm will be presented in Section 4.6.2. Other methods are in [124], [104] and [32].

3
EXISTING ALGORITHMS

In this chapter, we briefly describe a number of existing algorithms for the nonnegative matrix factorization problem and related issues such as: algorithm initializations, stopping conditions and convergence.

We choose typical algorithms in three main categories: the multiplicative updates, the alternating least squares methods and the gradient based methods. This list is established based on the popularity of the algorithms in practice. The earliest algorithm is the *alternating least squares* method proposed by Paatero [97] for the *positive matrix factorization*. But the the attention of this part-based analysis technique really took off after the introduction of the *multiplicative updates* of Lee and Seung [80]. The problem was then rebaptised to *nonnegative matrix factorization*. The simplicity of the multiplicative updates and the interpretability of the result helped to spread the influence of the nonnegative matrix factorizations to almost all research fields: image processing [59] [53] [83], text processing [128] [103], music transcription [108], video analysis [34], bioinformatics [46], chemistry [45], etc. It was solved using the standard *projected gradient* method only in [86], where some advantages in large-scale problems are reported. Recently, a revised version of the alternating least squares has been proposed in [12] offering a faster implementation by sacrifying the convergence property. Other attempts try to make a change of variable to eliminate the nonnegativity constraints. For example, in [26], $u \rightarrow x^2$ is used and two gradient algorithms are proposed. But they are, reportedly, not very efficient. Here we analyze a recently proposed method: the *rank-one residue iteration* algorithms that will be investigated in details in Chapter 4. Fast convergence property

without hidden parameters would make this method a good choice for the current and future applications. Its variants derived throughout the last four chapters of this thesis demonstrate its flexibility when additional constraints are imposed.

We classify algorithms into two categories according to the search space: *Full-space search* and *(Block-)Coordinate search*. Algorithms like standard gradient methods can belong to both categories.

Algorithms in the former category try to find updates for both U and V at the same time. This requires a search for a descent direction in the $(m+n)r$-dimensional space. Note also that the nonnegative matrix factorization problem in this full space is not convex but the convergence of algorithms using the full-space search might be easier to be proved.

Algorithms in the latter category, on the other hand, find updates for each (block) coordinate in order to guarantee the descent of the objective function. Usually, search subspaces are chosen to make the objective function convex so that efficient methods can be applied. Such a simplification might lead to the loss of some convergence properties. Most of the algorithms use the following column partitioning:

$$\frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}\|A_{:,i} - U(V_{i,:})^T\|_2^2, \tag{3.1}$$

which shows that one can minimize with respect to each of the rows of V independently. The problem thus decouples into smaller convex problems. This leads to the solution of quadratic problems of the form

$$\min_{v\geq 0}\frac{1}{2}\|a - Uv\|_2^2, \tag{3.2}$$

which is called Nonnegative Least Squares (NNLS).

Updates for the rows of V are then alternated with updates for the rows of U in a similar manner by transposing A and UV^T.

We begin the chapter with the description of the three categories of algorithms. More emphasis is put into the multiplicative rules, since they are very popular but still lack a good convergence property. We will try to explain why this method may fail to converge to a local minimum. We end the chapter with two short discussions of the stopping conditions and the initialization methods.

All the described algorithms are numerically compared at the end of Chapter 4, where a new algorithm is analyzed.

3.1 Lee and Seung algorithm

The most popular algorithm for the NMF problem are the multiplicative rules (Algorithm 2) suggested by Lee and Seung [80].

To formulate these rules, we choose to fix one of the factors (i.e. U or V) and try to minimize the cost function with respect to the other factor. The following development illustrates how to formulate the updating rule for V. We first assume that U and V are positive and will come back to this later.

Since the cost function can be decoupled as follows :

$$\frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}\|A_{:i} - U(V_{:i})^T\|_2^2,$$

one can minimize it with respect to each of the rows of V separately. This results in solving a sequence of quadratic problems as follows:

$$\min_{v\geq 0} F(v) \quad \text{where } F(v) = \frac{1}{2}\|a - Uv\|_2^2.$$

Consider a current approximation $\bar{v} > 0$ of the solution and formulate the following problem:

$$\min_{v\geq 0} \bar{F}(v) = \min_{v\geq 0} \frac{1}{2}\left[\|a - Uv\|_2^2 + (v-\bar{v})^T H_{\bar{v}}(v-\bar{v})\right] \tag{3.3}$$

where $H_{\bar{v}} = D_x - U^T U$ with $x = \frac{[U^T U\bar{v}]}{[\bar{v}]}$. Because we can prove the positive semidefiniteness of $H_{\bar{v}}$, we have $\bar{F}(v) \geq F(v)$ for all v and especially $\bar{F}(\bar{v}) = F(\bar{v})$. Furthermore, the function is also convex.

We set the derivative of $\hat{F}(v)$ to zero, i.e.

$$\nabla_v \bar{F} = U^T Uv - U^T a + H_{\bar{v}}(v - \bar{v}) = 0$$

in order to obtain the minimizer v^*:

$$(U^T U + H_{\bar{v}})v^* = U^T a - H_{\bar{v}}\bar{v}.$$

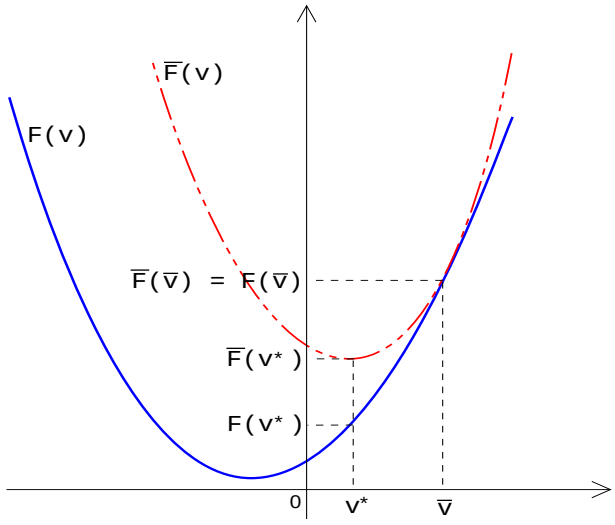


Figure 3.1: Non-increasing multiplicative update

Since $U^TU + H_{\bar{v}} = D_{U^TU\bar{v}}D_{\bar{v}}^{-1}$ and $H_{\bar{v}}\bar{v} = 0$, we conclude

$$v^* = \bar{v} \circ \frac{[U^Ta]}{[U^TU\bar{v}]}. \tag{3.4}$$

Since v^* is the global minimizer of $\bar{F}(v)$, we have $\bar{F}(v^*) \leq \bar{F}(\bar{v})$. Moreover, $\bar{F}(v)$ is constructed to satisfy $\bar{F}(v) \geq F(v)$ for all v. This implies that $F(v^*) \leq \bar{F}(v^*) \leq \bar{F}(\bar{v}) = F(\bar{v})$ or we have a descent on the cost function. This can be summarized in Figure 3.1. The function $\bar{F}$ is usually called the auxiliary function.

Solving for all rows of V results in the desired updating rule for V. The updating rule for U can be derived similarly. And these updates are the two alternating steps in Algorithm 2.

The additional term in (3.3) can also be seen as a penalty function to prevent the solution of the optimization problem from zero. Moreover, the matrix $D_x - U^TU$ with $x = \frac{[U^TU\bar{v}]}{[\bar{v}]}$ can be seen as an approximation of the Hessian matrix U^TU.

The above development can be summarized in the following Theorem:

Algorithm 2 Multiplicative Rules (Mult)

1: Initialize U^0, V^0 and $k = 0$
2: **repeat**
3: $\quad U^{k+1} = U^k \circ \frac{[AV^k]}{[U^k(V^k)^T V^k]}$
4: $\quad V^{k+1} = V^k \circ \frac{[A^T U^{k+1}]}{[V^k(U^{k+1})^T U^{k+1}]}$
5: $\quad k = k + 1$
6: **until** Stopping condition

Theorem 3.1. *The Euclidean distance* $\|A - UV^T\|_F^2$ *is non-increasing under the updating rules of Algorithm 2.*

Theorem 3.1 is a shortened version of the one in the original paper of Lee and Seung [80]. The original theorem has an additional part claiming that the Euclidean distance is unchanged under the multiplicative rules *only* when it is at a stationary point. This is, in fact, not necessarily true, since if it converges, only the conditions (2.4) and (2.6) are satisfied at the fixed point. No proof is provided to show that the conditions (2.5) can be met. There are two main obstacles in the investigation of the convergence of these multiplicative rules.

The first one is that these multiplicative rules fail to make a *sufficient* descent of the cost function. To see this, we can rewrite (3.4) as a variable metric method [13]:

$$\begin{aligned} v^* &= \bar{v} \circ \frac{[U^T a]}{[U^T U \bar{v}]} \\ &= \bar{v} \circ \frac{[U^T a + U^T U \bar{v} - U^T U \bar{v}]}{[U^T U \bar{v}]} \\ &= \bar{v} \circ \left(\mathbf{1} + \frac{[U^T a - U^T U \bar{v}]}{[U^T U \bar{v}]} \right) \\ &= \bar{v} - \frac{[\bar{v}]}{[U^T U \bar{v}]} \circ (U^T U \bar{v} - U^T a) \\ &= \bar{v} - D_{\bar{v}} \nabla_{\bar{v}F}, \end{aligned} \tag{3.5}$$

where $D_{\bar{v}}$ is a positive diagonal matrix with $D_{ii} = \frac{[\bar{v}]_i}{[U^T U \bar{v}]_i}$. With this

update, a necessary condition for a sufficient descent is that the eigenvalues of matrix $D_{\bar{v}}$ (i.e. D_{ii}) must be bounded above and away from zero [13]. But this is not true for $D_{\bar{v}}$ in general. Hence, the limit points of the algorithm may not be stationary.

The second obstacle is the possibility of zeros in U and V when considering the conditions (2.5). The situation that we want to avoid is

$$\nabla_{V_{ij}} F < 0 \text{ while } V_{ij} = 0 \tag{3.6}$$

for some (i, j) at a limit point. With the assumption that the initial U and V are positive, this problem might not happen because if, for some k:

$$\nabla_{V_{ij}^k} F < 0 \quad \text{or} \quad [V^k (U^k)^T U^k]_{ij} - [AU^k]_{ij} < 0,$$

we have from the update:

$$V_{ij}^{k+1} = V_{ij}^k \frac{[AU^k]_{ij}}{[V^k (U^k)^T U^k]_{ij}} > V_{ij}^k > 0.$$

This is also supported by the following Lemma [85]:

Lemma 3.2. *If U^0 and V^0 are positive and A is both row-allowable and column-allowable (see Section 1.1.6), then U^k and V^k are positive.*

But in practice, when using a finite precision system of number, only numbers whose magnitude is larger than the constant ϵ_M (the smallest representable number of a number system) can be represented by nonzero numbers. Otherwise, they will be rounded to zero. And since we can not bound V_{ij}^k away from zero, it could be smaller than ϵ_M hence represented by 0. Once $V_{ij}^k = 0$, the multiplicative updates will not make it positive again. The algorithm will then be probably trapped into a non-stationary point.

One way to avoid this is to look at every $U_{ij} = 0$ (or $V_{ij} = 0$), if the corresponding element of the gradient is negative, we can set $U_{ij} = \epsilon$ (or $V_{ij} = \epsilon$) with $\epsilon > \epsilon_M$ as suggested in [85] or making a gradient descent step to the inside of the nonnegative orthant.

Remark: it is possible that $[V^k (U^k)^T U^k]_{ij} = 0$ for some (i, j), which results in some zero-division exceptions. We investigate two following possible situations:

- When $V_{ij}^k > 0$, $[V^k(U^k)^T U^k]_{ij} = 0$ implies that

$$0 = \sum_{lt} V_{it}^k U_{lt}^k U_{lj}^k \geq V_{ij}^k U_{lj}^k U_{lj}^k = V_{ij}^k \sum_l U_{lj}^k U_{lj}^k.$$

  This occurs only when $U_{:j}^k = 0$, which is due to a rank-deficient approximation and can be fixed by generating a substitution for $U_{:j}^k$.

- When $V_{ij}^k = 0$, we have a $0/0$ situation where

$$\nabla_{V_{ij}} F = -[AU^k]_{ij} \leq 0.$$

  Then, we should not replace $[V^k(U^k)^T U^k]_{ij}$ by ϵ (a small positive constant with $\epsilon > \epsilon_M$) as suggested by many works, because this will keep $V_{ij}^{k+1} = 0$ which is unfavorable for the multiplicative updates. Setting $V_{ij}^{k+1} = \epsilon > \epsilon_M$ is definitely a better choice.

The multiplicative rules are also extended to the weighted nonnegative matrix factorization (see Chapter 6), to the generalized Kullback-Leibler divergence (see Chapter 5) and to a broader class of cost function namely Bregman divergence [36]. Many other extensions can be found in [59], [119], [83], [122], [68], etc.

3.2 Alternating least squares methods

The first algorithm proposed for solving the nonnegative matrix factorization was the alternating least squares method [97]. It is known that, fixing either U or V, the problem becomes a least squares problem with nonnegativity constraint.

Since the least squares problems in Algorithm 3 can be perfectly decoupled into smaller problems corresponding to the columns or rows of A, we can directly apply methods for the Nonnegative Least Square problem to each of the small problem. Methods that can be applied are [79], [44], [24], etc.

A direct application of Theorem 1.11 can show that if the subproblem (3) and (4) in Algorithm 3 are exactly and *uniquely* solved, every limit

Algorithm 3 Alternating Least Square (ALS)

1: Initialize U and V
2: **repeat**
3: Solve: $\min_{V \geq 0} \frac{1}{2}\|A - UV^T\|_F^2$
4: Solve: $\min_{U \geq 0} \frac{1}{2}\|A^T - VU^T\|_F^2$
5: **until** Stopping condition

Algorithm 4 Inexact Alternating Least Square (IALS)

1: Initialize U and V
2: **repeat**
3: Solve for U in equation: $UV^TV = AV$
4: $U = [U]_+$
5: Solve for V in equation: $VU^TU = A^TU$
6: $V = [V]_+$
7: **until** Stopping condition

point of Algorithm 3 is a stationary point of the nonnegative matrix factorization problem.

But even with the faster implementation of these algorithm, they can not match other methods in terms of running time. A modification has been made by replacing an exact solution of the nonnegative least squares problem by the projection of the solution of the unconstrained least squares problem into the nonnegative orthant [12] as in Algorithm 4. This speeds up the algorithm by sacrifying the convergence property. Figure 3.2 is a typical example of the convergence of the Alternating Least Squares and the Inexact Alternating Least Squares. One can see that while the earlier always makes a descent update, the latter does not. The exact method also produces better approximation errors. But with the same number of iterations, it spends significantly more time than the inexact version does (3.435s vs. 0.02s). Note that the solver for the nonnegative least squares problem in this example is the standard Matlab function *lsqnonneg*. For a faster solver such as [24], it is reported that the exact method is still far behind in terms of the running time. In practice, the *exact* Alternating Least Squares is seldomly used because it is very inefficient. And its inexact version does not, in general, converges

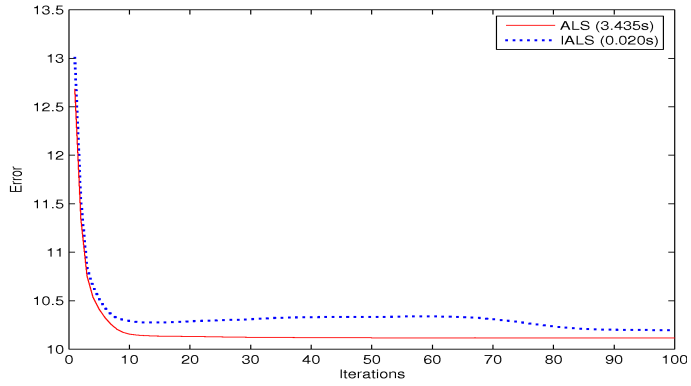


Figure 3.2: Alternating Least Square (ALS) vs. Inexact Alternating Least Square (IALS)

to a stationary point. It is suggested to use the inexact version as a warming-up phase of a hybrid algorithm [48].

Two other versions namely Alternating Constrained Least Square and Alternating Hoyer-Constrained Least Square are also given in [12].

3.3 Gradient descent

We can consider the nonnegative matrix factorization as a nonlinear optimization problem on a convex set, which is the nonnegative orthant. We also know that the projection on this set is very simple and consists of setting any negative element to zero. In this case, the Projected Gradient scheme is often used and characterized by the following three basic steps in each iteration:

- Calculating the gradient $\nabla F(x^k)$,
- Choosing the step size α^k,
- Projecting the update on the nonnegative orthant $\mathbb{R}^n_+$:

$$x^{k+1} = [x^k - \alpha^k \nabla F(x^k)]_+,$$

where x^k is the variable. The last two steps can be merged in one iterative process and must guarantee a sufficient decrease of the objective

function as well as the nonnegativity of the new point. This generates an inner loop inside each iteration.

We will present two simple ways to carry out this idea in the nonnegative matrix factorization. Both methods use the negative gradient as the basic search direction. Only the stepsizes are different.

In Section 3.3.3, some issues of the implementation will be pointed out. Especially for the case where one chooses to use the gradient method in alternating iterations, i.e. minimizing with respect to U and to V in an alternating fashion.

3.3.1 Line search using Armijo rule (Line)

Algorithm 5 Line search (Line)

Initialize x^0, σ, β, $\alpha_0 = 1$ and $k = 1$
repeat
  $\alpha_k = \alpha_{k-1}$
  $y = [x^k - \alpha_k \nabla F(x^k)]_+$
  if $F(y) - F(x^k) > \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$ **then**
    repeat
      $\alpha_k = \alpha_k \cdot \beta$
      $y = [x^k - \alpha_k \nabla F(x^k)]_+$
    until $F(y) - F(x^k) \leq \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$
  else
    repeat
      $lasty = y$
      $\alpha_k = \alpha_k / \beta$
      $y = [x^k - \alpha_k \nabla F(x^k)]_+$
    until $F(y) - F(x^k) > \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$
    $y = lasty$
  end if
  $x^{k+1} = y$
  $k = k + 1$
until Stopping condition

In order to ensure a sufficient descent, the following projected gradi-

ent scheme with Armijo criterion [86] can be applied to minimize

$$x^* = \underset{x}{\operatorname{argmin}} F(x).$$

Algorithm 5 needs two parameters σ and β that may affect its convergence. It requires only the gradient information, and is applied in [86] for two different strategies : for the whole space (U, V) (Algorithm FLine) and for U and V separately in an alternating fashion (Algorithm CLine). With a good choice of parameters ($\sigma = 0.01$ and $\beta = 0.1$) and a good strategy of alternating between variables, it was reported in [86] to be the faster than the multiplicative rules.

3.3.2 Gradient Projection using first order approximation

In order to find the solution to the problem:

$$x^* = \underset{x}{\operatorname{argmin}} F(x)$$

we can also approximate at each iteration the function $F(X)$ using:

$$\tilde{F}(x) = F(x^k) + \left\langle \nabla_x F(x^k), x - x^k \right\rangle + \frac{L}{2}\|x^k - x\|_2^2,$$

where L is a Lipschitz constant, the smallest value of L satisfying $F(x) \leq \tilde{F}(x),\ \forall x$.

Because of this inequality, the solution of the following problem

$$x_{k+1} = \underset{x \geq 0}{\operatorname{argmin}} \tilde{F}(x)$$

also is a point of descent for the function $F(x)$ since

$$F(x_{k+1}) \leq \tilde{F}(x_{k+1}) \leq \tilde{F}(x_k) = F(x_k).$$

Since the constant L is not known a priori, an inner loop is needed. Algorithm 6 presents an iterative way to carry out this scheme. As in the previous algorithm this also requires only the gradient information and can therefore can be applied to two different strategies: to the whole space (U, V) (Algorithm FFO) and to U and V separately in an alternating fashion (Algorithm CFO).

A main difference with the previous algorithm is its stopping criterion for the inner loop. This algorithm requires also a parameter β for which the practical choice is 2.

Algorithm 6 First order approximation (FO)

Initialize x^0, L_0 and $k = 0$
repeat
  $y = [x^k - \frac{1}{L_k}\nabla F(x^k)]_+$
  while $F(y) - F(x^k) > \langle \nabla F(x^k), y - x^k \rangle + \frac{L_k}{2}\|y - x^k\|_2^2$ **do**
    $L_k = L_k/\beta$
    $Y = [x^k - \frac{1}{L_k}\nabla F(x^k)]_+$
  end while
  $x^{k+1} = y$
  $L_{k+1} = L_k \cdot \beta$
  $k = k + 1$
until Stopping condition

3.3.3 Implementation

The most time-consuming job is the test for the sufficient decrease, which is also the stopping condition for the inner loop. As mentioned at the beginning of the chapter, the above methods can be carried out using two different strategies:

Full space search: The exact evaluation of $F(x) = F(U, V) = \|A - UV^T\|_F^2$ need $O(mnr)$ operations. When there is a correction $y = (U + \Delta U, V + \Delta V)$, we have to calculate $F(y)$ which also requires $O(mnr)$ operations. Hence, it requires $O(tmnr)$ operations to determine a stepsize in t iterations of the inner loop.

Coordinate search: when V is fixed, the Euclidean distance is a quadratic function on U:

$$\begin{aligned} F(U) = \|A - UV^T\|_F^2 &= \langle A, A\rangle - 2\left\langle UV^T, A\right\rangle + \left\langle UV^T, UV^T\right\rangle \\ &= \|A\|_F^2 - 2\langle U, AV\rangle + \left\langle U, U(V^TV)\right\rangle. \end{aligned}$$

The most expensive step is the computation of AV, which requires $O(mnr)$ operations. But when V is fixed, AV can be calculated once at the beginning of the inner loop. The remaining computations are $\langle U, AV\rangle$ and $\langle U, U(V^TV)\rangle$, which requires $O(nr)$ and $O(nr^2 + nr)$ operations. Therefore, it requires $O(tnr^2)$ operations to determine a stepsize in t iterations of the inner loop which is much less than $O(tmnr)$ oper-

ations. This is due to the assumption $r \ll n$. Similarly, when U fixed, $O(tmr^2)$ operations are needed to determine a stepsize.

If we consider an iteration is a sweep, i.e. once all the variables are updated, the following table summarizes the complexity of each sweep of the described algorithms:

Algorithm	Complexity per iteration
Mult	$O(mnr)$
FLine	$O(tmnr)$
CLine	$O(t_1nr^2 + t_2mr^2)$
FFO	$O(tmnr)$
CFO	$O(t_1nr^2 + t_2mr^2)$
ALS	$O(2^rmnr)^*$
IALS	$O(mnr)$

where t, t_1 and t_2 are the number of iterations of inner loops, which can not be bounded in general. For algorithm ALS, the complexity is reported for the case where the active set method [79] is used. Although $O(2^rmnr)$ is a very high *theorical* upper bound that count all the possible subsets of r variables of each subproblem, in practice, the active set method needs much less iterations to converge. One might as well use more efficient convex optimization tools to solve the subproblems instead of the active set method.

3.4 Scaling and stopping criterion

For descent methods, several stopping conditions are used in the literature. We now discuss some problems when implementing these conditions for NMF.

The very first condition is the decrease of the objective function. The algorithm should stop when it fails to make the objective function decrease with a certain amount :

$$F(U^{k+1}, V^{k+1}) - F(U^k, V^k) < \epsilon \quad or \quad \frac{F(U^{k+1}, V^{k+1}) - F(U^k, V^k)}{F(U^k, V^k)} < \epsilon.$$

This is not a good choice for all cases since the algorithm may stop at a point very far from a stationary point. Time and iteration bounds can

also be imposed for very slowly converging algorithms. But here again this may not be good for the optimality conditions. A better choice is probably the norm of the projected gradient as suggested in [86]. For the NMF problem it is defined as follows :

$$[\nabla^P_X]_{ij} = \begin{cases} [\nabla_X]_{ij} & \text{if } X_{ij} > 0 \\ \min(0, [\nabla_X]_{ij}) & \text{if } X_{ij} = 0 \end{cases}$$

where X stands for U or V. The proposed condition then becomes

$$\left\| \begin{pmatrix} \nabla^P_{U^k} \\ \nabla^P_{V^k} \end{pmatrix} \right\|_F \leq \epsilon \left\| \begin{pmatrix} \nabla_{U^1} \\ \nabla_{V^1} \end{pmatrix} \right\|_F . \tag{3.7}$$

We should also take into account the scaling invariance between U and V. Putting $\bar{U} = \gamma U$ and $\bar{V} = \frac{1}{\gamma} V$ does not change the approximation UV^T but the above projected gradient norm is affected:

$$\begin{aligned} \left\| \begin{pmatrix} \nabla^P_{\bar{U}} \\ \nabla^P_{\bar{V}} \end{pmatrix} \right\|^2_F &= \|\nabla^P_{\bar{U}}\|^2_F + \|\nabla^P_{\bar{V}}\|^2_F = \frac{1}{\gamma^2}\|\nabla^P_U\|^2_F + \gamma^2 \|\nabla^P_V\|^2_F \quad (3.8) \\ &\neq \left\| \begin{pmatrix} \nabla^P_U \\ \nabla^P_V \end{pmatrix} \right\|^2_F . \end{aligned}$$

Two approximate factorizations $UV^T = \bar{U}\bar{V}^T$ resulting in the same approximation should be considered equivalent in terms of precision. One could choose $\gamma^2 := \|\nabla^P_U\|_F / \|\nabla^P_V\|_F$, which minimizes (3.8) and forces $\|\nabla^P_{\bar{U}}\|_F = \|\nabla^P_{\bar{V}}\|_F$, but this may not be a good choice when only one of the gradients $\|\nabla^P_{\bar{U}}\|_F$ and $\|\nabla^P_{\bar{V}}\|_F$ is nearly zero.

In fact, the gradient $\begin{pmatrix} \nabla_U \\ \nabla_V \end{pmatrix}$ is scale dependent in the NMF problem and any stopping criterion that uses gradient information is affected by this scaling. To limit that effect, we suggest the following scaling after each iteration:

$$\tilde{U}_k \leftarrow U_k D_k \qquad \tilde{V}_k \leftarrow V_k D_k^{-1}$$

where D_k is a positive diagonal matrix:

$$[D_k]_{ii} = \sqrt{\frac{\|V_{:i}\|_2}{\|U_{:i}\|_2}}.$$

This ensures that $\|\tilde{U}_{:i}\|_F^2 = \|\tilde{V}_{:i}\|_F^2$ and hopefully reduces also the difference between $\|\nabla^P_{\tilde{U}}\|_F^2$ and $\|\nabla^P_{\tilde{V}}\|_F^2$. Moreover, it may help to avoid some numerically unstable situations.

The same scaling should be applied to the initial point as well (U_1, V_1) when using (3.7) as the stopping condition.

3.5 Initializations

We know that the nonnegative matrix factorization problem is not convex, and hence expect it to have local minima. Moreover, starting from a point too far from the stationary points costs any iterative method many iterations before getting close to one such stationary point. A good initial approximation could not only shift the interest to good local minima but would also help to reduce the number of iterations. Several techniques are proposed in the literature. Among them, an application of the Spherical K-Means algorithm [127] to generate an initial point shows a nice improvement but is too expensive. This is not surprising since K-Means is in fact an instance of NMF factorization, as we mentioned in the introduction. Other interesting methods proposed in [2] provide some ideas but have a small improvement over the random initialization. The best reported method is the *SVD-centroid* which requires a partial SVD decomposition of the original matrix, which is not always available and needs to be computed. Recently proposed method [20] takes the rank-r SVD decomposition $U\Sigma V^T$ of the target matrix A, removes all the negative values of U and V and creates a simple rank-$2r$ nonnegative factorization as an initialization. Although a *loose* upper bound of the error can be derived, this method again needs a SVD decomposition. These mentioned methods is reported to behave well for some specific datasets and algorithms, but they may not be as good when applied to other cases. A deeper investigation will be necessary.

Since nothing is known about the distribution of the minima, we believe that a random initialization maybe a good starting point. In fact, in most existing applications, a random initialization is used because of its simplicity. We have to point out that, using directly a random initialization is very naive, and a preprocessing step can improve significantly the quality of the initial point. For example, the following step can be

applied to algorithms which are sensitive to scalings (multiplicative updates and gradient methods):

$$\alpha := \frac{\left\langle A, U_0 V_0^T \right\rangle}{\left\langle U_0 V_0^T, U_0 V_0^T \right\rangle}, \qquad U_0 = U_0 \sqrt{\alpha}, \qquad V_0 = V_0 \sqrt{\alpha},$$

where the scaling factor α is in fact the optimal solution of the problem:

$$\min_{\alpha} \|A - \alpha U_0 V_0^T\|_F^2.$$

4
RANK-ONE RESIDUE ITERATION

In the previous chapter, we have seen that it is very appealing to decouple the problem into convex subproblems. But this may "converge" to solutions that are far from the global minimizers of the problem.

This chapter is an extension of the internal report [62], where we proposed to decouple the problem based on rank-one approximations to create a new algorithm. A convergence analysis, numerical experiments and some extensions were also presented for this algorithm. During the completion of the revised version of this report, we were informed that two other independent reports [31] and [49] had also proposed this algorithm.

This algorithm has several advantages : it allows us to formulate a very simple basic subproblem, which can be solved in closed form, seems to have useful convergence results and can be extended to more general types of factorizations such as for nonnegative tensors. Moreover, the experiments in Section 4.7 suggest that this new method outperforms the other ones in most cases.

Here, we present an introduction and an analysis of this algorithm, compare it with existing algorithms and test new extensions.

4.1 Motivation

NMF problem is, in fact, a least square problem with nonnegativity constraints. The classical nonnegative least square problem [79] is defined as below:

Problem 4.1 (NNLS)**.**

$$\min_{x \geq 0} \|Ax - b\|_F^2.$$

As we can see NMF problem can be decoupled into several problems 4.1. Therefore it is quite natural to apply any method for the above problem to solve the NMF problem. The standard one is the *active set* method proposed by Lawson and Hanson [79]. But even with the improved version by Bro and Jong [24], using this for NMF problem still appears very slow.

One of the main reasons lies in the reusability of the computations. The decoupling helps to break the NMF problem into smaller problems but for many methods, solving a decoupled NNLS problem may not help for solving other NNLS problems. We are talking about solving a number of separate nonnegative least square problems

$$\min_{x \geq 0} \|A_{:,i} - Ux\|_F^2, \ i = 1, 2, ..., n.$$

The *active set* method actually consists in solving a sequence of normal equations where a number of pseudo-inverse matrices need to be computed. These computations could be reused to speed up solving for multiple columns of A [74][24].

These methods start with all the columns of U to solve the normal equation $U^T Ux = U^T A_{:,i}$ where one needs to compute $U^+ = (U^T U)^{-1} U$. At each iteration, several columns of U are removed creating smaller matrices $U^1, U^2, \ldots$ Then, $U^{i+} = (U^{iT} U^i)^{-1} U^i$ are computed. Clearly, keeping and reusing all $(U^i)^+$ matrices is impossible for large problem, since the number of those matrices might be 2^r, total number of possible subsets from r columns of U.

In [44], a different strategy is deployed to create an *iterative* method. At each iteration, only one element x_i is updated. The elementary problem to be solved is:

Problem 4.2.

$$\min_{x_i \in \mathbb{R}_+} \|x_i U_{:,i} - r^i\|_F^2,$$

where $r^i = A_{:,i} - \sum_{j \neq i} x_j U(:, j)$ is the residue vector.

The optimal solution to this problem is trivially given by

$$x_i^* = \max\left(\frac{\langle r^i, U_{:,i}\rangle}{\langle U_{:,i}, U_{:,i}\rangle}, 0\right).$$

In fact, this is an instance of Problem 4.1 where U has only one column. And $U_{:,i}^+ = \frac{U_{:,i}}{\langle U_{:,i}, U_{:,i}\rangle}$ can be reused when there are multiple columns of A.

We will use this idea to create an iterative method for the standard NMF problem and derive other NMF problems. An iterative method for Nonnegative Tensor Factorization (NTF) will also be derived.

4.2 Column partition of variables

Let the u_i's and v_i's be respectively the columns of U and V. Then the NMF problem can be rewritten as follows :

Problem 4.3 (Nonnegative Matrix Factorization)**.** Given a $m \times n$ nonnegative matrix A, solve

$$\min_{u_i \geq 0\ v_i \geq 0} \frac{1}{2}\|A - \sum_{i=1}^{r} u_i v_i^T\|_F^2.$$

Let us fix all the variables, except for a single vector v_t and consider the following least squares problem:

$$\min_{v \geq 0} \frac{1}{2}\|R_t - u_t v^T\|_F^2, \tag{4.1}$$

where $R_t = A - \sum_{i \neq t} u_i v_i^T$. We have:

$$\begin{aligned} \|R_t - u_t v^T\|_F^2 &= trace\left[(R_t - u_t v^T)^T (R_t - u_t v^T)\right] && (4.2)\\ &= \|R_t\|_F^2 - 2v^T R_t^T u_t + \|u_t\|_2^2 \|v\|_2^2. && (4.3) \end{aligned}$$

From this formulation, one now derives the following lemma.

Lemma 4.4. *If $[R_t^T u_t]_+ \neq 0$, then $v_* := \frac{[R_t^T u_t]_+}{\|u_t\|_2^2}$ is the unique global minimizer of (4.1) and the function value equals $\|R_t\|_F^2 - \frac{\|[R_t^T u_t]_+\|_2^2}{\|u_t\|_2^2}$.*

Proof. Let us permute the elements of the vectors $x := R_t^T u_t$ and v such that

$$Px = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad Pv = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \quad \text{with} \quad x_1 \geq 0, \quad x_2 < 0$$

and P is the permutation matrix. Then

$$\|R_t - u_t v^T\|_F^2 = \|R_t\|_F^2 - 2v_1^T x_1 - 2v_2^T x_2 + \|u_t\|_2^2 (v_1^T v_1 + v_2^T v_2).$$

Since $x_2 < 0$ and $v_2 \geq 0$, it is obvious that $\|R_t - u_t v^T\|_F^2$ can only be minimal if $v_2 = 0$. Our assumption implies that x_1 is nonempty and $x_1 > 0$. Moreover $[R_t^T u_t]_+ \neq 0$ and $u_t \geq 0$ imply $\|u_t\|_2^2 > 0$, one can then find the optimal v_1 by minimizing the remaining quadratic function

$$\|R_t\|_F^2 - 2v_1^T x_1 + \|u_t\|_2^2 v_1^T v_1$$

which yields the solution $v_1 = \frac{x_1}{\|u_t\|_2^2}$. Putting the two components together yields the result

$$v_* = \frac{[R_t^T u_t]_+}{\|u_t\|_2^2} \quad \text{and} \quad \|R_t - u_t v_*^T\|_F^2 = \|R_t\|_F^2 - \frac{\|[R_t^T u_t]_+\|_2^2}{\|u_t\|_2^2}.$$

□

Remark 1: The above lemma has of course a dual form, where one fixes v_t but solves for the optimal u to minimize $\|R_t - u v_t^T\|_F^2$. This would yield the updating rules

$$v_t \leftarrow \frac{[R_t^T u_t]_+}{\|u_t\|_2^2} \quad \text{and} \quad u_t \leftarrow \frac{[R_t v_t]_+}{\|v_t\|_2^2} \tag{4.4}$$

which can be used to recursively update approximations $\sum_{i=1}^r u_i v_i^T$ by modifying each rank-one matrix $u_t v_t^T$ in a cyclic manner. This problem is different from the NMF, since the error matrices $R_t = A - \sum_{i \neq t} u_i v_i^T$ are no longer nonnegative. We will therefore call this method the *Rank-one Residue Iteration* (RRI), i.e. Algorithm 7. The same algorithm was independently reported as Hierarchical Alternating Least Squares (HALS) [31] and Alternating NMF (ANMF) [49].

Algorithm 7 (RRI)

Initialize u_i's, v_i's, for $i = 1$ to r
repeat
  for $t = 1$ to r **do**
    $R_t = A - \sum_{i \neq t} u_i v_i^T$

    if $[R_t^T u_t]_+ \neq 0$ **then**
      $v_t \leftarrow \frac{[R_t^T u_t]_+}{\|u_t\|_2^2}$
    else
      $v_t = 0$
    end if

    if $[R_t v_t]_+ \neq 0$ **then**
      $u_t \leftarrow \frac{[R_t v_t]_+}{\|v_t\|_2^2}$
    else
      $u_t = 0$
    end if
  end for
until Stopping condition

Remark 2: In case where $[R_t^T u_t]_+ = 0$, we have a trivial solution for $v = 0$ that is not covered by Lemma 4.4. In addition, if $u_t = 0$, this solution is no longer unique. In fact, v can be arbitrarily taken to construct a rank-deficient approximation. The effect of this on the convergence of the algorithm will be discussed further in the next section.

Remark 3: Notice that the optimality of Lemma 4.4 implies that $\|A - UV^T\|$ can not increase. And since $A \geq 0$ fixed, $UV^T \geq 0$ must be bounded. Therefore, its component $u_i v_i^t$ (i=1...r) must be bounded as well. One can moreover scale the vector pairs (u_i, v_i) at each stage as explained in Section 3.4 without affecting the local optimality of Lemma 4.4. It then follows that the rank one products $u_i v_i^T$ and their scaled vectors remain bounded.

4.3 Convergence

In the previous section, we have established the partial updates for each of the variable u_i or v_i. And for a NMF problem where the reduced rank is r, we have in total $2r$ vector variables (the u_i's and v_i's). The described algorithm can be also considered as a projected gradient method since the update (4.4) can be rewritten as:

$$\begin{aligned} u_t &\leftarrow \frac{[R_t v_t]_+}{\|v_t\|_2^2} = \frac{[(A - \sum_{i \neq t} u_i v_i^T) v_t]_+}{\|v_t\|_2^2} = \frac{[(A - \sum_i u_i v_i^T + u_t v_t^T) v_t]_+}{\|v_t\|_2^2} \\ &= \frac{[(A - \sum_i u_i v_i^T) v_t + u_t v_t^T v_t]_+}{\|v_t\|_2^2} = \left[u_t - \frac{1}{\|v_t\|_2^2} \nabla_{u_t} \right]_+ . \end{aligned}$$

Similarly, the update for v_i can be rewritten as

$$v_t \leftarrow \left[v_t - \frac{1}{\|u_t\|_2^2} \nabla_{v_t} \right]_+ .$$

Therefore, the new method follows the projected gradient scheme described in the previous section. But it produces the optimal solution in closed form. For each update of a column v_t (or u_t), the proposed algorithm requires just a matrix-vector multiplication $R_t^T u_t$ (or $R_t v_t$), wherein the residue matrix $R_t = A - \sum_{i \neq t} u_i v_i^T$ does not have to be calculated explicitly. Indeed, by calculating $R_t^T u_t$ (or $R_t v_t$) from $A^T u_t$ (or $A v_t$) and $\sum_{i \neq t} v_i (u_i^T u_t)$ (or $\sum_{i \neq t} u_i (v_i^T v_t)$), the complexity is reduced from $O(mnr + mn)$ to only $O\left(mn + (m+n)(r-1)\right)$ which is majored by $O(mn)$. This implies that the complexity of each sweep through the $2r$ variables $u_t's$ and $v_t's$ requires only $O(mnr)$ operations, which is equivalent to a sweep of the multiplicative rules and to an inner loop of any gradient methods. This is very low since the evaluation of the whole gradient requires already the same complexity.

Because at each step of the $2r$ basic steps of Algorithm 7, we compute an optimal rank-one nonnegative correction to the corresponding error matrix R_t the Frobenius norm of the error can not increase. This is a reassuring property but it does not imply convergence of the algorithm.

Each vector u_t or v_t lies in a convex set $\mathbb{U}_t \subset \mathbb{R}_+^m$ or $\mathbb{V}_t \subset \mathbb{R}_+^n$. Moreover, because of the possibility to include scaling we can set an

upper bound for $\|U\|$ and $\|V\|$, in such a way that all the $\mathbb{U}_t$ and $\mathbb{V}_t$ sets can be considered as closed convex. Then, we can use the following Theorem 4.5, to prove a stronger convergence result for Algorithm 7.

Theorem 4.5. *Every limit point generated by Algorithm 7 is a stationary point.*

Proof. We notice that, if $u_t = 0$ and $v_t = 0$ at some stages of Algorithm 7, they will remain zero and no longer take part in all subsequent iterations. We can divide the execution of Algorithm 7 into two phases.

During the first phase, some of the pairs (u_t, v_t) become zero. Because there are only a finite number ($2r$) of such vectors, the number of iterations in this phase is also finite. At the end of this phase, we can rearrange and partition the matrices U and V such that

$$U = (U_+ \; 0) \text{ and } V = (V_+ \; 0),$$

where U_+ and V_+ do not have any zero column. We temporarily remove zero columns out of the approximation.

During the second phase, no column of U_+ and V_+ becomes zero, which guarantees the updates for the columns of U_+ and V_+ are unique and optimal. Moreover, $\frac{1}{2}\|A - \sum_{i=1}^{r} u_i v_i^T\|_F^2$ is continuously differentiable over the set $\mathbb{U}_1 \times \ldots \times \mathbb{U}_r \times \mathbb{V}_1 \times \ldots \times \mathbb{V}_r$, and the $\mathbb{U}_i$'s and $\mathbb{V}_i$'s are closed convex. A direct application of Theorem 1.11 proves that every stationary point (U_+^*, V_+^*) is a stationary point. It is then easy to prove that if there are zero columns removed at the end of the first phase, adding them back yields another stationary point: $U^* = (U_+^* \; 0)$ and $V^* = (V_+^* \; 0)$ of the required dimension. However, in this case, the rank of the approximation will then be lower than the requested dimension r. □

In Algorithm 7, variables are updated in this order: u_1, v_1, u_2, v_2, ... We can alternate the variables in a different order as well, for example u_1, u_2, ..., u_r v_1, v_2, ..., v_r, ... Whenever this is carried out in a cyclic fashion, the Theorem 4.5 still holds and this does not increase the complexity of each iteration of the algorithm.

As pointed above, stationary points given by Algorithm 7 may contain useless zero components. To improve this, one could replace

$u_t v_t^T (\equiv 0)$ by any nonnegative rank-one approximation that reduces the norm of the error matrix. For example, the substitution

$$u_t = e_{i^*} \qquad v_t = [R_t^T u_t]_+, \tag{4.5}$$

where $i^* = \text{argmax}_i \, \|[R_t^T e_i]_+\|_2^2$, reduces the error norm by $\|[R_t^T e_i]_+\|_2^2 > 0$ unless $R_t \leq 0$. These substitutions can be done as soon as u_t and v_t start to be zero. If we do these substitutions in only a *finite* number of times before the algorithm starts to converge, Theorem 4.5 still holds. In practice, only a few such substitutions in total are usually needed by the algorithm to converge to a stationary point without any zero component. Note that the matrix rank of the approximation might not be r, even when all u_t's and v_t's ($t = 1 \ldots r$) are nonzero.

A possibly better way to fix the problem due to zero components is to use the following *damped RRI algorithm* in which we introduce new $2r$ dummy variables $w_i \in \mathbb{U}_i$ and $z_i \in \mathbb{V}_i$, where $i = 1...r$. The new problem to solve is:

Problem 4.6 (Damped Nonnegative Matrix Factorization)**.**

$$\min_{\substack{u_i \geq 0 \; v_i \geq 0 \\ w_i \geq 0 \; z_i \geq 0}} \frac{1}{2} \|A - \sum_{i=1}^{r} u_i v_i^T\|_F^2 + \frac{\psi}{2} \sum_i \|u_i - w_i\|_2^2 + \frac{\psi}{2} \sum_i \|v_i - z_i\|_2^2,$$

where the damping factor ψ is a positive constant.

Again, the coordinate descent scheme is applied with the cyclic update order: $u_1, w_1, v_1, z_1, u_2, w_2, v_2, z_2, \ldots$ to result in the following optimal updates for u_t, v_t, w_t and z_t:

$$u_t = \frac{[R_t v_t]_+ + \psi w_t}{\|v_t\|_2^2 + \psi}, \; w_t = u_t, \; v_t = \frac{[R_t^T u_t]_+ + \psi z_t}{\|u_t\|_2^2 + \psi} \text{ and } z_t = v_t \tag{4.6}$$

where $t = 1 \ldots r$. The updates $w_t = u_t$ and $z_t = v_t$ can be integrated in the updates of u_t and v_t to yield Algorithm 8. We have the following results:

Theorem 4.7. *Every limit point generated by Algorithm 8 is a stationary point of NMF problem 4.3.*

Algorithm 8 (Damped RRI)

1: Initialize u_i's, v_i's, for $i = 1$ to r
2: **repeat**
3: **for** $t = 1$ to r **do**
4: $R_t = A - \sum_{i \neq t} u_i v_i^T$
5: $v_t \leftarrow \frac{[R_t^T u_t + \psi v_t]_+}{\|u_t\|_2^2 + \psi}$
6: $u_t \leftarrow \frac{[R_t v_t + \psi u_t]_+}{\|v_t\|_2^2 + \psi}$
7: **end for**
8: **until** Stopping condition

Proof. Clearly the cost function in Problem 4.6 is continuously differentiable over the set $\mathbb{U}_1 \times \ldots \times \mathbb{U}_r \times \mathbb{U}_1 \times \ldots \times \mathbb{U}_r \times \mathbb{V}_1 \times \ldots \times \mathbb{V}_r \times \mathbb{V}_1 \times \ldots \times \mathbb{V}_r$, and the $\mathbb{U}_i$'s and $\mathbb{V}_i$'s are closed convex. The uniqueness of the global minimum of the elementary problems and a direct application of Theorem 1.11 prove that every limit point of Algorithm 8 is a stationary point of Problem 4.6.

Moreover, at a stationary point of Problem 4.6, we have $u_t = w_t$ and $v_t = z_t$, $t = 1...r$. The cost function in Problem 4.6 becomes the cost function of the NMF problem 4.3. This implies that every stationary point of Problem 4.6 yields a stationary point of the standard NMF problem 4.3. □

This *damped* version not only helps to eliminate the problem of zero components in the convergence analysis but may also help to avoid zero columns in the approximation when ψ is carefully chosen. But it is not an easy task. Small values of ψ provide an automatic treatment of zeros while not changing much the updates of RRI. Larger values of ψ might help to prevent the vectors u_t and v_t ($t = 1 \ldots r$) from becoming zero too soon. But too large values of ψ limit the updates to only small changes, which will slow down the convergence.

In general, the rank of the approximation can still be lower than the requested dimension. Patches may still be needed when a zero component appears. Therefore, in our experiments, using the *undamped* RRI algorithm 7 with the substitution (4.5) is still the best choice.

This damping technique will be reused to create convergent algo-

rithms in later chapters, when the uniqueness problem shows up again.

4.4 Variants of the RRI method

We now extend the Rank-one Residue Iteration by using a factorization of the type XDY^T where D is diagonal and nonnegative and the columns of the nonnegative matrices X and Y are normalized. The NMF formulation then becomes

$$\min_{\substack{x_i \in \mathbb{X}_i \; y_i \in \mathbb{Y}_i \\ d_i \in \mathbb{R}_+}} \frac{1}{2} \| A - \sum_{i=1}^{r} d_i x_i y_i^T \|_F^2,$$

where $\mathbb{X}_i$'s and $\mathbb{Y}_i$'s are sets of normed vectors.

The variants that we present here depend on the choice of $\mathbb{X}_i$'s and $\mathbb{Y}_i$'s. A generalized Rank-one Residue Iteration method for low-rank approximation is given in Algorithm 9. This algorithm needs to solve a sequence of elementary problems of the type:

$$\max_{s \in \mathbb{S}} y^T s \tag{4.7}$$

where $y \in \mathbb{R}^n$ and $\mathbb{S} \subset \mathbb{R}^n$ is a set of normed vectors. We first introduce a permutation vector $I_y = (i_1 \; i_2 \; \ldots \; i_n)$ which reorders the elements of y in non-increasing order : $y_{i_k} \geq y_{i_{k+1}}, k = 1 \ldots (n-1)$. The function $p(y)$ returns the number of positive entries of y.

Algorithm 9 GRRI

1: Initialize x_i's, y_i's and d_i's, for $i = 1$ to r
2: **repeat**
3: **for** $t = 1$ to r **do**
4: $R_t = A - \sum_{i \neq t} d_i x_i y_i^T$
5: $y_t \leftarrow \text{argmax}_{s \in \mathbb{Y}_t} \left(x_t^T R_t s \right)$
6: $x_t \leftarrow \text{argmax}_{s \in \mathbb{X}_t} \left(y_t^T R_t^T s \right)$
7: $d_t = x_t^T R_t y_t$
8: **end for**
9: **until** Stopping condition

Let us first point out that for the set of normed nonnegative vectors the solution of problem (4.7) is given by $s^* = \frac{y_+}{\|y_+\|_2}$. It then follows that Algorithm 9 is essentially the same as Algorithm 7 since the solutions v_i and u_i of each step of Algorithm 9, given by (4.4), correspond exactly to those of problem (4.7) via the relations $y_i = u_i/\|u_i\|_2$, $y_i = v_i/\|v_i\|_2$ and $d_i = \|u_i\|_2\|v_i\|_2$.

Below we list the sets for which the solution s^* of (4.7) can be easily computed.

- *Set of normed nonnegative vectors*: $s = \frac{y_+}{\|y_+\|_2}$.
- *Set of normed bounded nonnegative vectors* $\{s\}$: where $0 \leq l_i \leq s_i \leq p_i$. The optimal solution of (4.7) is given by:

$$s = \max\left(l,\ \min\left(p,\ \frac{y_+}{\|y_+\|_2}\right)\right).$$

- *Set of normed binary vectors* $\{s\}$: where $s = \frac{b}{\|b\|}$ and $b \in \{0,1\}^n$. The optimal solution of (4.7) is given by:

$$[s^*]_{i_t} = \begin{cases} \frac{1}{\sqrt{k^*}} & \text{if } t \leq k^* \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } k^* = \underset{k}{\operatorname{argmax}} \frac{\sum_{t=1}^{k} y_{i_t}}{\sqrt{k}}.$$

- *Set of normed sparse nonnegative vectors*: all normed nonnegative vectors having at most K nonzero entries. The optimal solution for (4.7) is given by norming the following vector p^*

$$[p^*]_{i_t} = \begin{cases} y_{i_t} & \text{if } t \leq \min(p(y), K) \\ 0 & \text{otherwise} \end{cases}$$

- *Set of normed fixed-sparsity nonnegative vectors*: all nonnegative vectors s a fixed sparsity, where

$$sparsity(s) = \frac{\sqrt{n} - \|s\|_1/\|s\|_2}{\sqrt{n} - 1}.$$

The optimal solution for (4.7) is given by using the projection scheme in [68].

One can also imagine other variants, for instance by combining the above ones. Depending on how data need to be approximated, one can create new algorithms provided it is relatively simple to solve problem (4.7). There have been some particular ideas in the literatures such as NMF with sparseness constraint [68], Semidiscrete Matrix Decomposition [76] and Semi-Nonnegative Matrix Factorization [38] for which variants of the above scheme can offer an alternative choice of algorithm.

Remark: Only the first two sets are the normed version of a closed convex set, as required for the convergence by Theorem 4.5. Therefore the algorithms might not converge to a stationary point with the other sets. However, the algorithm always guarantees a non-increasing update even in those cases and can therefore be expected to return a *good* approximation.

4.5 Regularizations

The regularizations are common methods to cope with the ill-posedness of inverse problems. Having known some additional information about the solution, one may want to imposed a priori some constraints to algorithms, such as: smoothness, sparsity, discreteness, etc. To add such regularizations in to the RRI algorithms, it is possible to modify the NMF cost function by adding some regularizing terms. We will list here the update for u_i's and v_i's when some simple regularizations are added to the original cost function. The proof of these updates are straight-forward and hence omitted.

- One-Norm $\|.\|_1$ regularization: the one-norm of the vector variable can be added as a heuristic for finding a sparse solution. This is an alternative to the fixed-sparsity variant presented above. The regularized cost function with respect to the variable v_t will be

$$\frac{1}{2}\|R_t - u_t v^T\|_F^2 + \beta\|v\|_1, \quad \beta > 0$$

  where the optimal update is given by

$$v_t^* = \frac{[R_t^T u_t - \beta \mathbf{1}_{n\times 1}]_+}{\|u_t\|_2^2}.$$

The constant $\beta > 0$ can be varied to control the trade-off between the approximation error $\frac{1}{2}\|R_t - u_t v^T\|_F^2$ and $\|v\|_1$. From this update, one can see that this works by zeroing out elements of $R_t^T u_t$ which are smaller than β, hence reducing the number of nonzero elements of v_t^*.

- Smoothness regularization $\|v - B\hat{v}_t\|_F^2$: where $\hat{v}_t$ is the current value of v_t and the matrix B helps to calculate the average of the neighboring elements at each element of v. When v is a 1D smooth function, B can be the following $n \times n$ matrix:

$$B = \begin{pmatrix} 0 & 1 & \dots & \dots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{4.8}$$

This matrix can be defined in a different way to take the true topology of v into account, for instance $v = vec(F)$ where F is a matrix. The regularized cost function with respect to the variable v_t will be

$$\frac{1}{2}\|R_t - u_t v^T\|_F^2 + \frac{\delta}{2}\|v - B\hat{v}_t\|_F^2, \quad \delta > 0$$

where the optimal update is given by

$$v_t^* = \frac{[R_t^T u_t + \delta B\hat{v}_t]_+}{\|u_t\|_2^2 + \delta}.$$

The constant $\delta \geq 0$ can be varied to control the trade-off between the approximation error $\frac{1}{2}\|R_t - u_t v^T\|_F^2$ and the smoothness of v_t at the fixed point. From the update, one can see that this works by searching for the optimal update v_t^* with some preference for the neighborhood of $B\hat{v}_i$, i.e., a smoothed vector of the current value $\hat{v}_t$.

The two above regularizations can be added independently to each of the columns of U and/or V. The trade-off factor β (or δ) can be

different for each column. A combination of different regularizations on a column (for instance v_t) can also be used to solve the multi-criterion problem

$$\frac{1}{2}\|R_t - u_t v^T\|_F^2 + \frac{\gamma}{2}\|v\|_2^2 + \frac{\delta}{2}\|v - B\hat{v}_t\|_F^2, \quad \beta, \gamma, \delta > 0$$

where the optimal update is given by

$$v_t^* = \frac{[R_t^T u_t - \beta \mathbf{1}_{n\times 1} + \delta B\hat{v}_t]_+}{\|u_t\|_2^2 + \delta}.$$

The one-norm regularizations as well as the two-norm regularization can be found in [2] and [12]. A major difference with that method is that the norm constraints is added to the rows rather than on the columns of V or U as done here. However, for the two versions of the one-norm regularization, the effects are somehow similar. While the two-norm regularization on the columns of U and V are simply scaling effects, which yield nothing in the RRI algorithm. We therefore only test the smoothness regularization at the end of the chapter with some numerical generated data.

In Chapter 7, a regularization will be added as an attempt to find a symmetric nonnegative approximation.

4.6 Algorithms for NMF extensions

In Section 2.4, two important extensions are introduced. In this section, we will derive the above algorithms to those extensions.

4.6.1 Multilayer nonnegative matrix factorization

We restate the multilayer nonnegative matrix factorization

$$\min_{X_i \geq 0} \frac{1}{2}\|A - X_1 X_1 \dots X_N\|_F^2, \tag{4.9}$$

where A is the nonnegative data matrix and V and X_i's are nonnegative matrix of compatible sizes.

We derive here a coordinate descent algorithm based on the above rank-one iteration. We create some intermediate matrices:

$$U_k = X_1 X_2 \dots X_k \text{ and } V_k = X_N^T X_{N-1}^T \dots X_k^T, \quad \text{with } k = 1, 2, \dots, N,$$

and $U_0 = I_m$ and $V_{N+1} = I_n$.

With those matrices, restricting to only one factor X_k, Problem 4.9 can be related as follows:

$$\min_{X_k \geq 0} \frac{1}{2} \|A - U_{k-1} X_k V_{k+1}^T\|_F^2, \quad \text{with } k = 1, \dots, N.$$

The updates for X_1 and X_N are easy. Since the problem becomes exactly that of the two-factor factorization, i.e.

$$\min_{X_1 \geq 0} \frac{1}{2} \|A - X_1 V_2^T\|_F^2 \quad \text{and} \quad \min_{X_N \geq 0} \frac{1}{2} \|A - U_{N-1} X_N\|_F^2.$$

To derive updates for the elements of X_k, $k = 2, \dots, (N-1)$, we transform the product $U_{k-1} X_k V_{k+1}^T$ into a vector using the operator $vec()$

$$vec(U_{k-1} X_k V_{k+1}^T) = (V_{k+1} \otimes U_{k-1}) vec(X_k).$$

Then the optimal X_k can be found by the following nonnegative least square

$$\min_{vec(X) \geq 0} \frac{1}{2} \|vec(A) - (V_{k+1} \otimes U_{k-1}) vec(X)\|_F^2$$

to which algorithms such as [79] and [24] can be applied. One can also derive updates for each element of X_k, as done in [44], which gives the following closed form update

$$[X_k]_{ij} = max\left(0, [X_k]_{ij} + \frac{[U_{k-1}]_{:i}^T E [V_{k+1}]_{:j}}{\|[U_{k-1}]_{:i}\|_2^2 \, \|[V_{k+1}]_{:j}\|_2^2}\right),$$

where $E = A - U_{k-1} X_k V_{k+1}^T$.

The approach proposed here is different from that in [29, 32] where a new layer is created by factorizing the last achieved layer using a NMF algorithm. First, a NMF factorization is created: $A \approx X_1 A_1$. Then the newly created matrix A_1 is approximated by $X_2 A_2$. This is repeated to create X_3, X_4, ... until the desired number of layers is reached. Clearly, the approximation error will be higher than our proposed method, since the cost function is minimized by taking into account all the layers.

4.6.2 Nonnegative tensor factorization

If we refer to the problem of finding the nearest nonnegative vector to a given vector a as the nonnegative approximation in one dimension, the NMF is its generalization in two dimensions and naturally, it can be extended to even higher-order tensor approximation problems. Algorithms described in the previous section use the closed form solution of the one dimensional problem to solve the two-dimensional problem. We now generalize this to higher orders. Since in one dimension such an approximation is easy to construct, we continue to use this approach to build the solutions for higher order problems.

For a low-rank tensor, there are two popular kinds of factored tensors, namely those of Tucker and Kruskal [5]. We only give an algorithm for finding approximations of Kruskal type. It is easy to extend this to tensors of Tucker type, but this is omitted here.

Given a d dimensional tensor T, we will derive an algorithm for approximating a nonnegative tensor by a rank-r nonnegative Kruskal tensor $S \in \mathbb{R}_+^{n_1 \times n_2 \times \ldots \times n_d}$ represented as a sum of r rank-one tensors:

$$S = \sum_{i=1}^{r} \sigma_i u_{1i} \star u_{2i} \star \ldots \star u_{di}$$

where $\sigma_i \in \mathbb{R}_+$ is a scaling factor, $u_{ti} \in \mathbb{R}_+^{n_t}$ is a normed vector (i.e. $\|u_{ti}\|_2 = 1$) and $a \star b$ stands for the outer product between two vectors or tensors a and b.

The following update rules are the generalization of the matrix case to the higher order tensor:

$$y = (\ldots((\ldots(R_k u_{1k})\ldots u_{(t-1)k})u_{(t+1)k})\ldots)u_{dk} \tag{4.10}$$

$$\sigma_k = \|[y]_+\|_2, \quad u_{tk} = \frac{[y]_+}{\sigma_k}, \tag{4.11}$$

where $R_k = T - \sum_{i \neq k} \sigma_i u_{1i} \star u_{2i} \star \ldots \star u_{di}$ is the residue tensor calculated without the k^{th} component of S and $R_k u_{ij}$ is the ordinary tensor/vector product in the corresponding dimension.

We can then produce an algorithm which updates in a cyclic fashion all vectors u_{ji}. This is in fact a direct extension to Algorithm 7, one can carry out the same discussion about the convergence here to guarantee

that each limit point of this algorithm is a stationary point for the nonnegative tensor factorization problem and to improve the approximation quality.

Again, as we have seen in the previous section, we can extend the procedure to take into account different constraints on the vectors u_{ij} such as discreteness, sparseness, etc.

The approach proposed here is again different from that in [29, 32] where a similar cascade procedure for multilayer nonnegative matrix factorization is used to compute a 3D tensor approximation. Clearly, the approximation error will be higher than our proposed method, since the cost function is minimized by taking into account all the dimensions.

4.7 Numerical experiments

Here we present several experiments to compare the different descent algorithms presented in this paper. For all the algorithms, the scaling scheme proposed in section 3.4 was applied.

4.7.1 Random matrices

We generated 100 random nonnegative matrices of different sizes whose elements are uniformly distributed. We used seven different algorithms to approximate each matrix:

- the multiplicative rule (**Mult**),
- alternating least squares using Matlab function *lsqnonneg* (**ALS**),
- a full space search using line search and Armijo criterion (**FLine**),
- a coordinate search alternating on U and V, and using line search and Armijo criterion (**CLine**),
- a full space search using first-order approximation (**FFO**),
- a coordinate search alternating on U and V, and using first-order approximation (**CFO**)
- an iterative rank-one residue approximation (**RRI**).

Table 4.1: Comparison of algorithms: Time limit is 45 *seconds. The given figures are the average successful running time over* 100 *random matrices. The figure* 0.02 (96) *means that the algorithm returns a result with the required precision* ϵ *within* 45 *seconds for only* 96 *(of* 100*) test matrices of which the average running time is* 0.02 *seconds.* 45 $*$ (0) *means that the algorithm fails in all* 100 *matrices.*

	Mult	ALS	FLine	CLine	FFO	CFO	RRI
(m=30, n=20, r=2)							
ϵ=10^{-2}	0.02 (96)	0.40	0.04	0.02	0.02	0.01	0.01
ϵ=10^{-3}	0.08 (74)	1.36	0.12	0.09	0.05	0.04	0.03
ϵ=10^{-4}	0.17 (71)	2.81	0.24	0.17	0.11	0.08	0.05
ϵ=10^{-5}	0.36 (64)	4.10	0.31	0.25	0.15	0.11	0.07
ϵ=10^{-6}	0.31 (76)	4.74	0.40	0.29	0.19	0.15	0.09
(m=100, n=50, r=5)							
ϵ=10^{-2}	45$*$ (0)	3.48	0.10	0.09	0.09	0.04	0.02
ϵ=10^{-3}	45$*$ (0)	24.30 (96)	0.59	0.63	0.78	0.25	0.15
ϵ=10^{-4}	45$*$ (0)	45$*$ (0)	2.74	2.18	3.34	0.86	0.45
ϵ=10^{-5}	45$*$ (0)	45$*$ (0)	5.93	4.06	6.71	1.58	0.89
ϵ=10^{-6}	45$*$ (0)	45$*$ (0)	7.23	4.75	8.98	1.93	1.30

Table 4.1 (cont.): Comparison of algorithms

	Mult	ALS	FLine	CLine	FFO	CFO	RRI
(m=100, n=50, r=10)							
ϵ=10^{-2}	45∗	11.61	0.28	0.27	0.18	0.11	0.05
	(0)						
ϵ=10^{-3}	45∗	41.89	1.90	2.11	1.50	0.74	0.35
	(0)	(5)					
ϵ=10^{-4}	45∗	45∗	7.20	5.57	5.08	2.29	1.13
	(0)	(0)					
ϵ=10^{-5}	45∗	45∗	12.90	9.69	10.30	4.01	1.71
	(0)	(0)					
ϵ=10^{-6}	45∗	45∗	14.62	11.68	13.19	5.26	2.11
	(0)	(0)	(99)	(99)			
(m=100, n=50, r=15)							
ϵ=10^{-2}	45∗	25.98	0.66	0.59	0.40	0.20	0.09
	(0)						
ϵ=10^{-3}	45∗	45∗	3.90	4.58	3.18	1.57	0.61
	(0)	(0)					
ϵ=10^{-4}	45∗	45∗	16.55	13.61	9.74	6.12	1.87
	(0)	(0)	(98)	(99)			
ϵ=10^{-5}	45∗	45∗	21.72	17.31	16.59	7.08	2.39
	(0)	(0)	(97)	(92)	(98)		
ϵ=10^{-6}	45∗	45∗	25.88	19.76	19.20	10.34	3.66
	(0)	(0)	(89)	(98)	(98)		
(m=100, n=100, r=20)							
ϵ=10^{-2}	45∗	42.51	1.16	0.80	0.89	0.55	0.17
	(0)	(4)					
ϵ=10^{-3}	45∗	45∗	9.19	8.58	10.51	5.45	1.41
	(0)	(0)					
ϵ=10^{-4}	45∗	45∗	28.59	20.63	29.89	12.59	4.02
	(0)	(0)	(86)	(94)	(69)		
ϵ=10^{-5}	45∗	45∗	32.89	27.94	34.59	18.83	6.59
	(0)	(0)	(42)	(68)	(34)	(90)	
ϵ=10^{-6}	45∗	45∗	37.14	30.75	36.48	22.80	8.71
	(0)	(0)	(20)	(60)	(8)	(87)	

Table 4.1 (cont.): Comparison of algorithms

	Mult	ALS	FLine	CLine	FFO	CFO	RRI
(m=200, n=100, r=30)							
ϵ=10^{-2}	45∗	45∗	2.56	2.20	2.68	1.31	0.44
	(0)	(0)					
ϵ=10^{-3}	45∗	45∗	22.60	25.03	29.67	12.94	4.12
	(0)	(0)	(99)	(98)	(90)		
ϵ=10^{-4}	45∗	45∗	36.49	39.13	45∗	33.33	14.03
	(0)	(0)	(2)	(13)	(0)	(45)	
ϵ=10^{-5}	45∗	45∗	45∗	39.84	45∗	37.60	21.96
	(0)	(0)	(0)	(2)	(0)	(6)	(92)
ϵ=10^{-6}	45∗	45∗	45∗	45∗	45∗	45∗	25.61
	(0)	(0)	(0)	(0)	(0)	(0)	(87)
(m=200, n=200, r=30)							
ϵ=10^{-2}	45∗	45∗	2.42	1.74	5.22	2.02	0.65
	(0)	(0)					
ϵ=10^{-3}	45∗	45∗	27.64	20.08	41.48	22.25	6.17
	(0)	(0)	(99)		(15)		
ϵ=10^{-4}	45∗	45∗	45∗	45∗	45∗	45∗	25.10
	(0)	(0)	(0)	(0)	(0)	(0)	(90)
ϵ=10^{-5}	45∗	45∗	45∗	45∗	45∗	45∗	30.42
	(0)	(0)	(0)	(0)	(0)	(0)	(51)
ϵ=10^{-6}	45∗	45∗	45∗	45∗	45∗	45∗	33.85
	(0)	(0)	(0)	(0)	(0)	(0)	(32)

For each matrix, the same starting point is used for every algorithm. We create a starting point by randomly generating two matrices U and V and then rescaling them to yield a first approximation of the original matrix A as proposed in Section 3.4:

$$U = UD\sqrt{\alpha}, \qquad V = VD^{-1}\sqrt{\alpha},$$

where

$$\alpha := \frac{\langle A, UV^T \rangle}{\langle UV^T, UV^T \rangle} \quad \text{and} \quad D_{ij} = \begin{cases} \sqrt{\frac{\|V_{:i}\|_2}{\|U_{:i}\|_2}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

The algorithms are all stopped when the projected gradient norm is lower than ϵ times the gradient norm at the starting point or when it takes more than 45 seconds. The relative precisions ϵ are chosen equal to 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-6}. No limit was imposed on the number of iterations.

For alternating gradient algorithms **CLine** and **CFO**, we use different precisions ϵ_U and ϵ_V for each of the inner iteration for U and for V as suggested in [86] where ϵ_U and ϵ_V are initialized by 10^{-3}. And when the inner loop for U or V needs no iteration to reach the precision ϵ_U or ϵ_V, one more digit of precision will be added into ϵ_U or ϵ_V (i.e. $\epsilon_U = \epsilon_U/10$ or $\epsilon_V = \epsilon_V/10$).

Table 4.1 shows that for all sizes and ranks, Algorithm **RRI** is the fastest to reach the required precision. Even though it is widely used in practice, algorithm **Mult** fails to provide solutions to the NMF problem within the allocated time. A further investigation shows that the algorithm gets easily trapped in boundary points where some U_{ij} and/or V_{ij} is zero while $\nabla_{U_{ij}}$ and/or $\nabla_{V_{ij}}$ is negative, hence violating one of the KKT conditions (2.5). The multiplicative rules then fail to move and do not return to a local minimizer. A slightly modified version of this algorithm was given in [85], but it needs to wait to get sufficiently close to such points before attempting an escape, and is therefore also not efficient. The **ALS** algorithm can return a stationary point, but it takes too long.

We select five methods: **FLine**, **CLine**, **FFO**, **CFO** and **RRI** for a more detailed comparison. For each matrix A, we run these algorithms with 100 different starting points. Figure 4.1, 4.2, 4.3 and 4.4 show

the results with some different settings. One can see that, when the approximated errors are almost the same between the algorithms, **RRI** is the best overall in terms of running times. It is probably because the RRI algorithm chooses only one vector u_t or v_t to optimize at once. This allows the algorithm to move *optimally* down on partial direction rather than just a *small step* on a more global direction. Furthermore, the computational load for an update is very small, only one matrix-vector multiplication is needed. All these factors make the running time of the RRI algorithm very attractive.

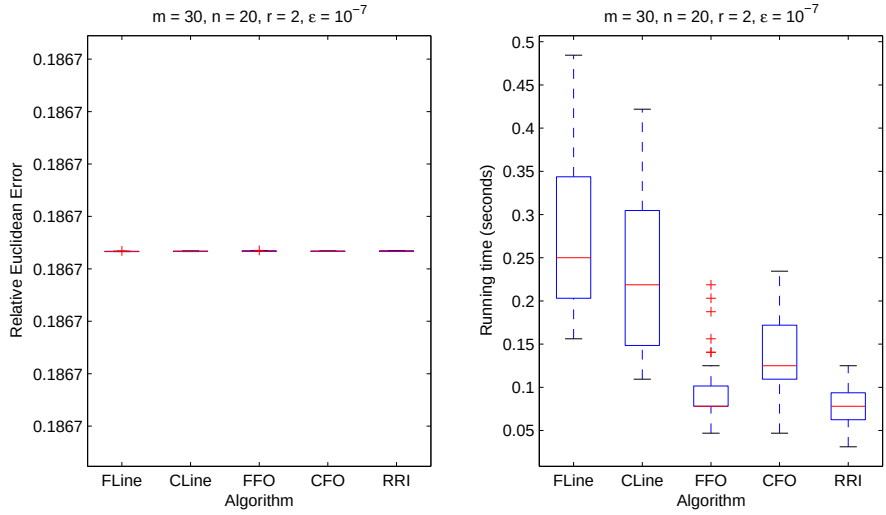


Figure 4.1: Comparison of selected algorithms: 100 *runs with* 100 *different initial points*

4.7.2 Image data

The following experiments use the Cambridge ORL face database as the input data. The database contains 400 images of 40 persons (10 images per person). The size of each image is 112×92 with 256 gray levels per pixel representing a front view of the face of a person. The images are then transformed into 400 "face vectors" in $\mathbb{R}^{10304}$ ($112 \times 92 = 10304$) to form the data matrix A of size 10304×400. We used three weight matrices of the same size of A (ie. 10304×400). Since it was used in [80],

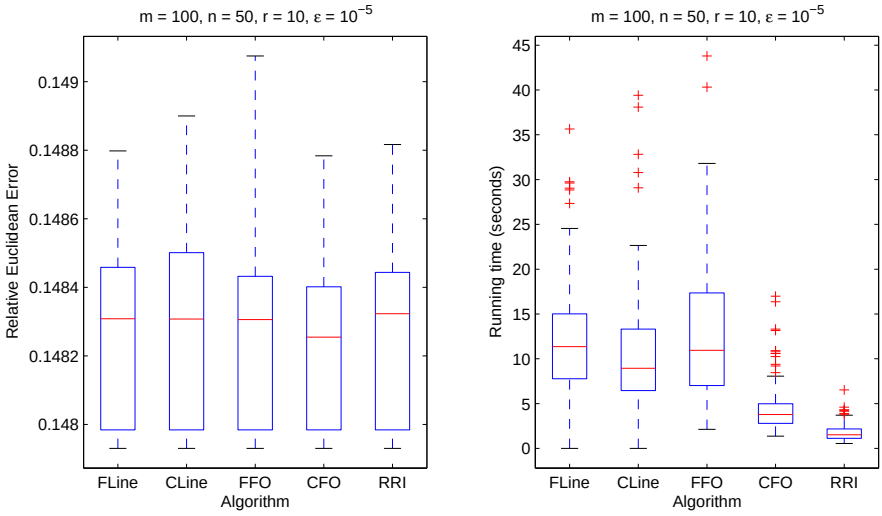


Figure 4.2: Comparison of selected algorithms: 100 *runs with* 100 *different initial points*

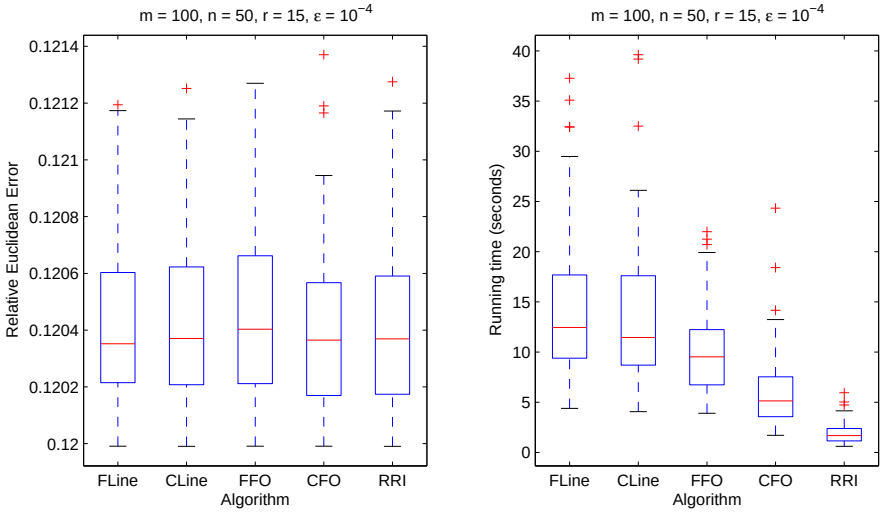


Figure 4.3: Comparison of selected algorithms: 100 *runs with* 100 *different initial points*

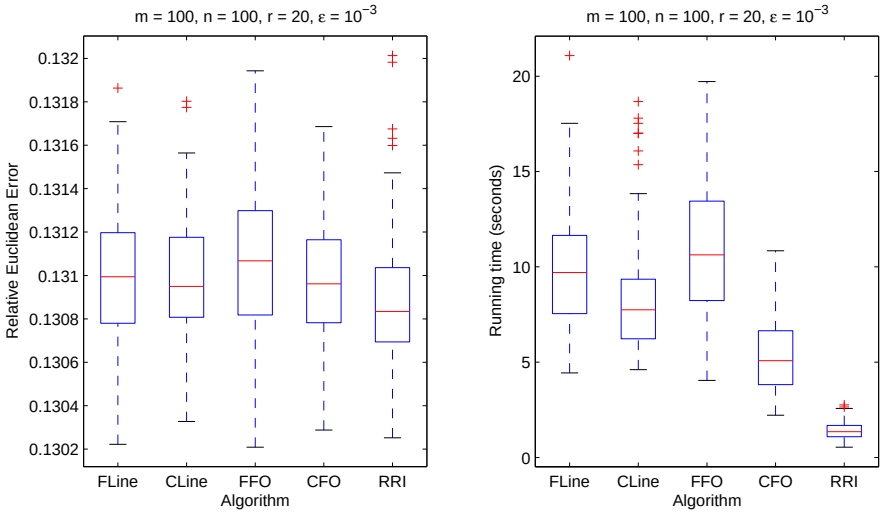


Figure 4.4: Comparison of selected algorithms: 100 *runs with* 100 *different initial points*

this data has become the standard benchmark for NMF algorithms.

In the first experiment, we run six NMF algorithms described above on this data for the reduced rank of 49. The original matrix A is constituted by transforming each image into one of its column. Figure 4.5 shows for the six algorithms the evolution of the error versus the number of iterations. Because the minimization process is different in each algorithm, we will say that one iteration corresponds to all elements of both U and V being updated. Figure 4.6 shows the evolution of the error versus time. Since the work of one iteration varies from one algorithm to another, it is crucial to plot the error versus time to get a fair comparison between the different algorithms. In the two figures, we can see that the RRI algorithm behaves very well on this dataset. And since its computation load of each iteration is small and constant (without inner loop), this algorithm converges faster than the others.

In the second experiment, we construct a third-order nonnegative tensor approximation. We first build a tensor by stacking all 400 images to have a $112 \times 92 \times 400$ nonnegative tensor. Using the proposed algorithm, a $rank - 142$ nonnegative tensor is calculated to approximate this tensor. Figure 4.7 shows the result for six images chosen

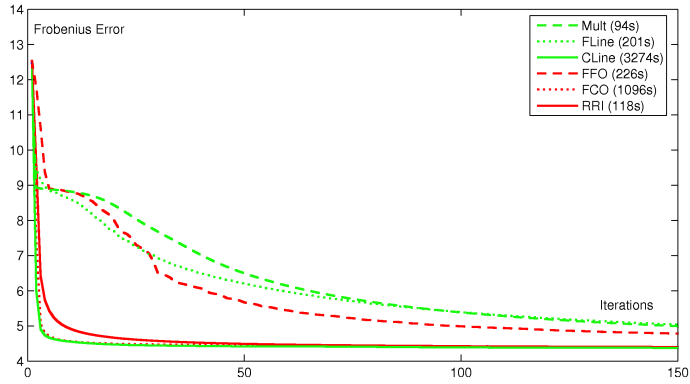


Figure 4.5: NMF: Error vs. Iterations

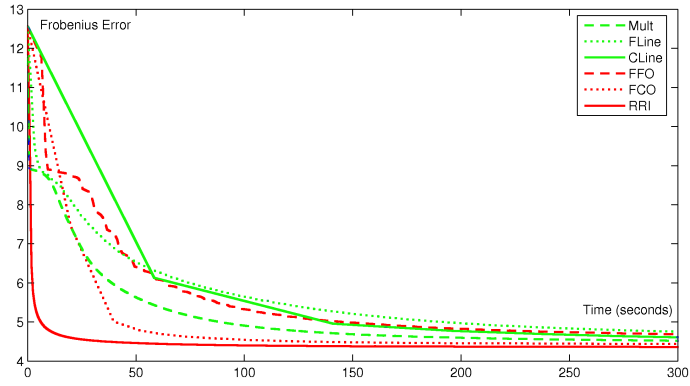


Figure 4.6: NMF: Error vs. Time

randomly from the 400 images. Their approximations given by the rank-142 nonnegative tensor are much better than that given by the rank-8 nonnegative matrix, even though they require similar storage space: $8 * (112 * 92 + 400) = 85632$ and $142 * (112 + 92 + 400) = 85768$. The rank-8 truncated SVD approximation (i.e. $[A_8]_+$) is also included for reference.

In the third experiment, we apply the variants of RRI algorithm mentioned in Section 4.4 to the face databases. The following settings are compared:

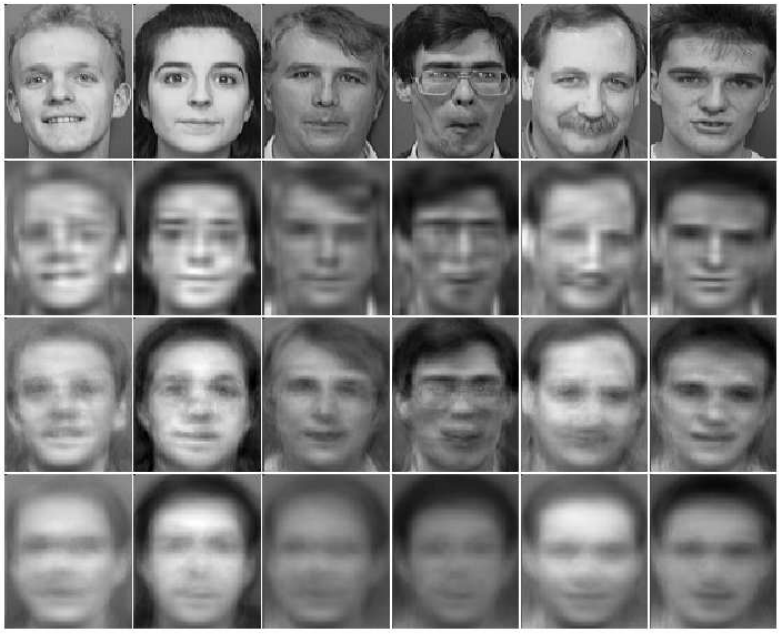


Figure 4.7: Tensor Factorization vs. Matrix Factorization on facial data. Six randomly chosen images from 400 of ORL dataset. From top to bottom: original images, their rank − 142 nonnegative tensor approximation, their rank − 8 truncated SVD approximation and their rank − 8 nonnegative matrix approximation.

- **Original**: original faces from the databases.
- **49NMF**: standard factorization (nonnegative vectors), $r = 49$.
- **100Binary**: columns of U are limited to the scaled binary vectors, $r = 100$.
- **49Sparse10**: columns of U are sparse. Not more than 10% of the elements of each column of A are positive. $r = 49$.
- **49Sparse20**: columns of U are sparse. Not more than 20% of the elements of each column of A are positive. $r = 49$.
- **49HSparse60**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.6. $r = 49$.

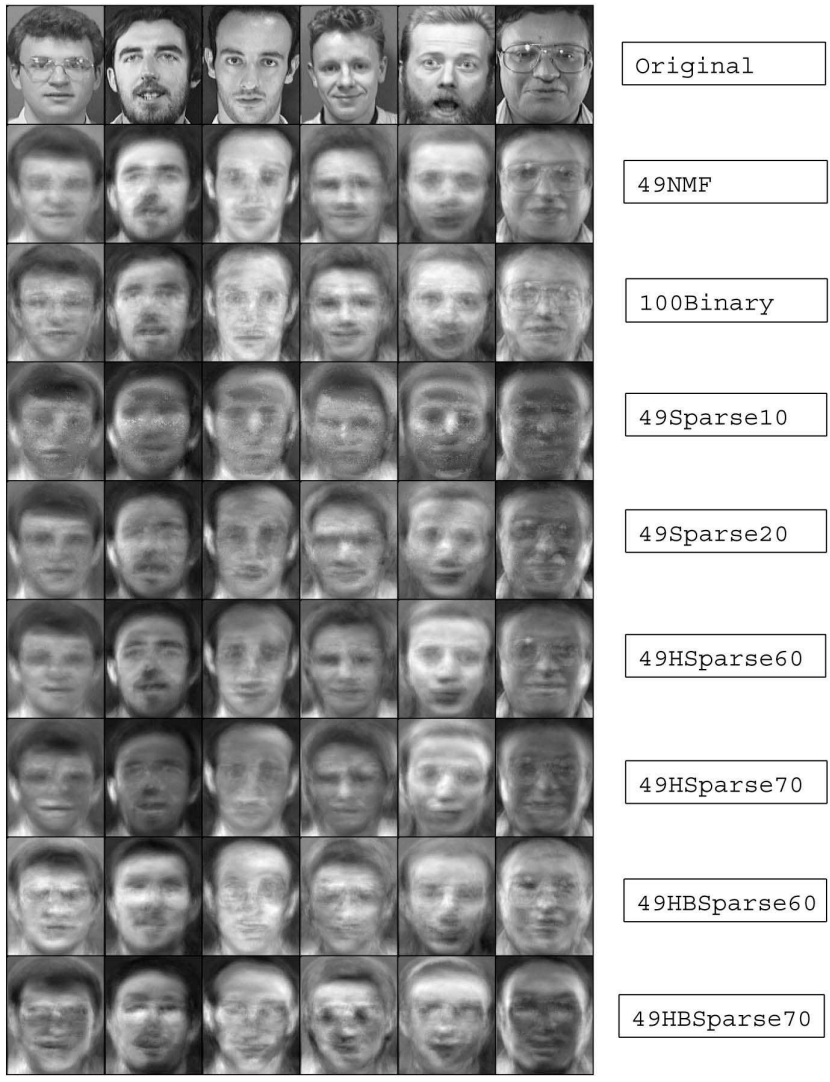


Figure 4.8: Nonnegative matrix factorization with several sparse settings

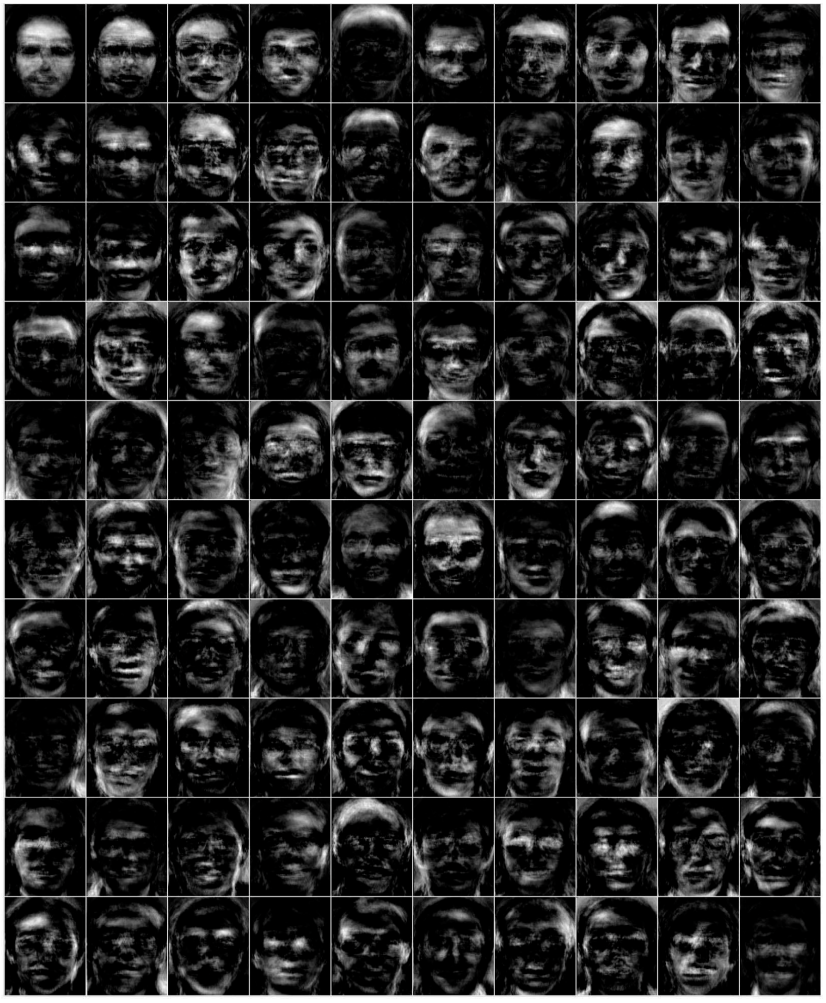


*Figure 4.9: Bases from **100Binary** setting*

- **49HSparse70**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.7. $r = 49$.
- **49HBSparse60**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.6. Columns of V are scaled binary. $r = 49$.
- **49HBSparse70**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.7. Columns of V are scaled binary. $r = 49$.

For each setting, we use RRI algorithm to compute the corresponding factorization. Some randomly selected faces are reconstructed by these settings as shown in Figure 4.8. For each setting, RRI algorithm produces a different set of bases to approximate the original faces. When the columns of V are constrained to scaled binary vectors (**100Binary**), the factorization can be rewritten as $UV^T = \hat{U}B^T$, where B is a binary matrix. This implies that each image is reconstructed by just the presence or absence of 100 bases shown in Figure 4.9.

Figure 4.10 and 4.11 show nonnegative bases obtained by imposing some sparsity on the columns of V. The sparsity can be easily controlled by the percentages of positive elements or by the Hoyer sparsity measure.

Figure 4.12 combines the sparsity of the bases (columns of U) and the binary representation of V. The sparsity is measured by he Hoyer measure as in Figure 4.11. Only with the absence or presence of these 49 *features*, faces are approximated as shown in the last two rows of Figure 4.8.

The above examples show how to use the variants of the RRI algorithm to control the sparsity of the bases. One can see that the sparser the bases are, the less storage is needed to store the approximation. Moreover, this provides a part-based decomposition using *local* features of the faces.

4.7.3 Smooth approximation

We carry out this experiment to test the new smoothness constraint introduced in the previous section:

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \frac{\delta}{2}\|v - B\hat{v}_i\|_F^2, \quad \delta > 0$$

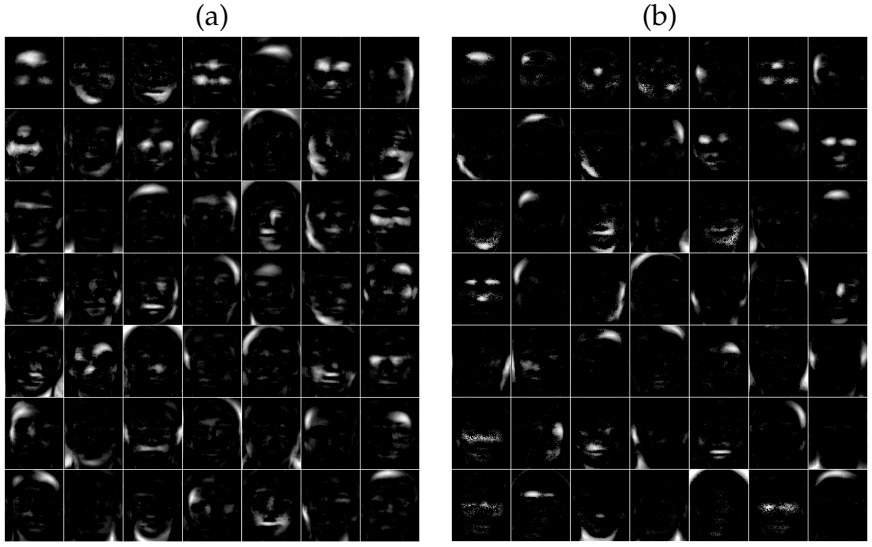


*Figure 4.10: Sparse bases **49Sparse20** and **49Sparse10**. Maximal percentage of positive elements is* 20% *(a) and* 10% *(b)*

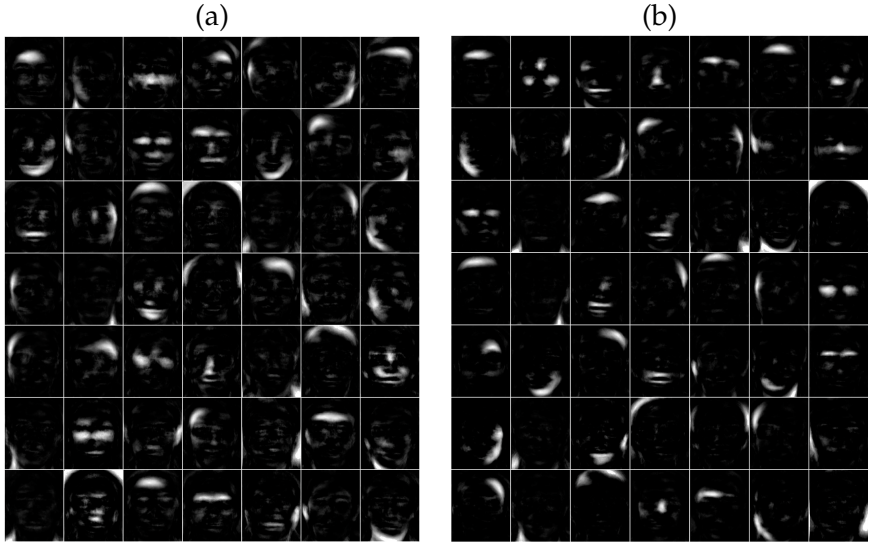


*Figure 4.11: Hoyer sparse bases **49HSparse60** and **49HSparse70**. Sparsity of bases is* 0.6 *(a) and* 0.7 *(b)*

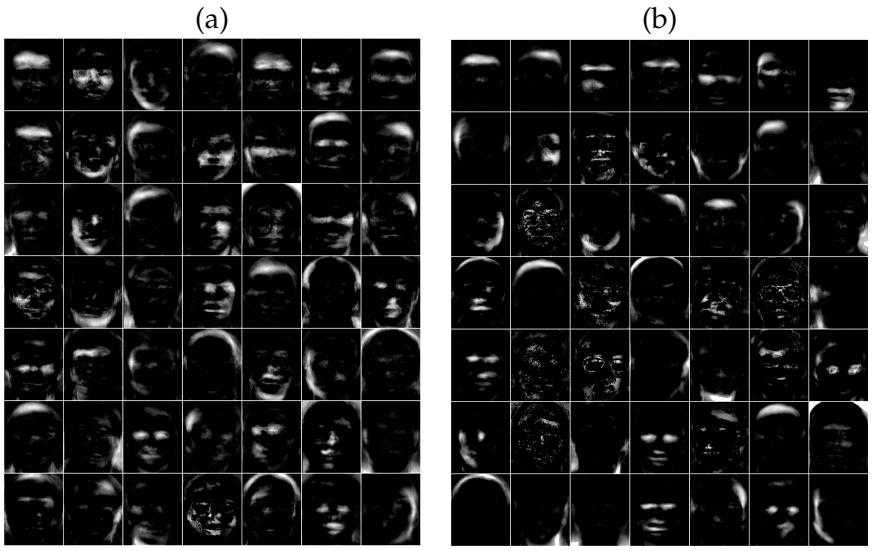


*Figure 4.12: Hoyer sparse bases **49HBSparse60** and **49HBSparse70**. Sparsity of bases is* 0.6 *(a) and* 0.7 *(b). V is binary matrix.*

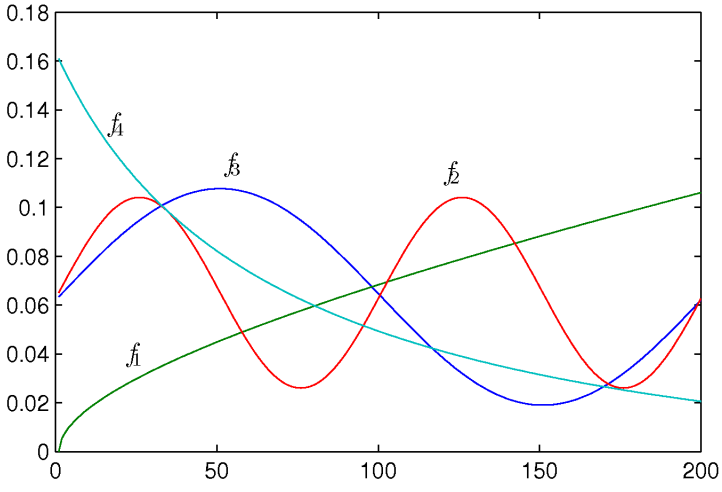


Figure 4.13: Smooth functions

where B is defined in (4.8).

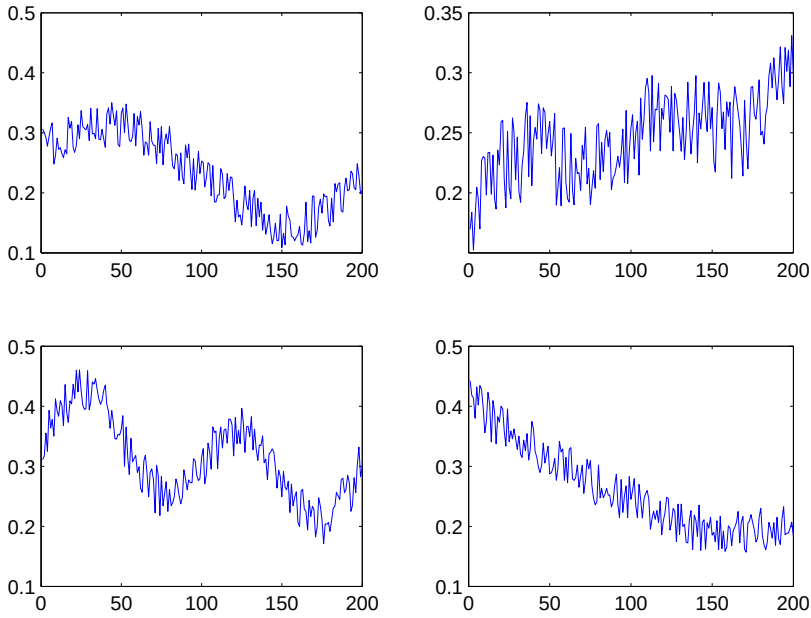


Figure 4.14: Randomly selected generated data

We generate the data using four smooth nonnegative functions f_1, f_2, f_3 et f_4, described in Figure 4.13, where each function is represented as a nonnegative vector of size 200.

We then generate a matrix A containing 100 mixture of these functions as follows

$$A = max(FE^T + N, 0)$$

where $F = [f_1\ f_2\ f_3\ f_4]$, E is a random nonnegative matrix and N is normally distributed random noise with $\|N\|_F = 0.2\|FE^T\|_F$. Four randomly selected columns of A are plotted in Figure 4.14.

We run the regularized RRI algorithm to force the smoothness of columns of U. We apply, for each run, the same value of δ for all the columns of U: $\delta = 0, 10, 100$. The results obtained through these runs are presented in Figure 4.15. We see that, without regularization, i.e. $\delta = 0$, the noise is present in the approximation, which produces nonsmooth solutions. When increasing the regularizing terms, i.e. $\delta = 10, 100$, the

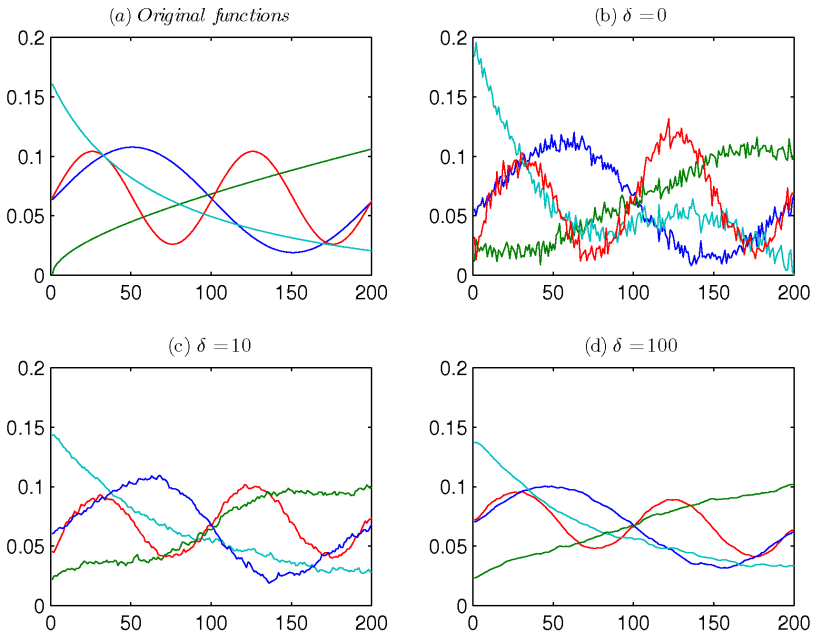


Figure 4.15: Original functions vs. reconstructed functions

reconstructed functions become smoother and the shape of the original functions are well preserved.

This smoothing technique can be used for applications like that in [99], where smooth spectral reflectance data from space objects is unmixed. The multiplicative rules are modified by adding the two-norm regularizations on the factor U and V to enforce the smoothness. This is a different approach, therefore, a comparison should be carried out.

We have described a new method for nonnegative matrix factorization that has a good and fast convergence. Moreover, it is also very flexible to create variants and to add some constraints as well. The numerical experiments show that this method and its derived variants behave very well with different types of data. This gives enough motivations to extend to other types of data and applications in the future. In the last two chapters of this thesis, it is applied to weighted cost functions and to symmetric factorizations.

5
NONNEGATIVE MATRIX FACTORIZATION WITH FIXED ROW AND COLUMN SUMS

An important class of nonnegative matrices are stochastic matrices. Typical applications of this are PageRank [23] used for ordering the web pages, Markov Clustering [117] used for clustering vertices in a graph and Hidden Markov Models [100] used for learning and predicting sequential data. The number of such applications is growing, but the size of the underlying problems is growing as well. The size of the stochastic matrices representing Markov chains related to the web is e.g. of the order of billions. In order to cope with such large-scale stochastic matrices, one could use NMF to approximate a large stochastic matrix by a lower-rank one.

We will begin with the introduction of the problem. We will show that the constraint under which the row and column sums are preserved is automatically satisfied by using the generalized Kullback-Leibler divergence as the cost function for the NMF problem. In [42], a discussion is made leading to the preservation of the matrix sum when using the generalized Kullback-Leibler divergence in the NMF. But this is just a straightforward result from the preservation of row and column sums, which will be pointed out.

5.1 Problem statement

The general NMF problem with preserved column and row sums is described as follows:

Problem 5.1 (NMF with preserved column and row sums)**.** Given a $m \times n$ nonnegative matrix A, solve

$$\min_{\substack{U \in \mathbb{R}_+^{m \times r} \; V \in \mathbb{R}^{n \times r} \\ UV^T \mathbf{1}_{n \times 1} = A \mathbf{1}_{n \times 1} \\ VU^T \mathbf{1}_{m \times 1} = A^T \mathbf{1}_{m \times 1}}} F(A, UV^T)$$

where $F(A, UV^T)$ is the chosen "distance" function.

The additional constraints are linear in each factor U or V (not on both). In fact, fixing for example U, they are linear on V. To see that they are not convex on both U and V, taking two feasible points, i.e. (U_1, V_1) and (U_2, V_2), their convex combinations $\alpha(U_1, V_1) + (1 - \alpha)(U_2, V_2)$, with $\alpha \in (0, 1)$ are, in general, not in the feasible set. When F is the Euclidean distance, one can use alternating minimizations to solve this problem, since the problem is convex in each of the factors U and V.

If we partition the variables as suggested in Chapter 4 and fix all the variables u_i's and v_i's except for one v_k, then the feasible set of v_k contains at most one element and could be empty. This means that we can not modify one variable without modifying the others while staying in the feasible set. In fact, the set of $m \times n$ *rank-one* nonnegative matrix with predefined nonnegative row and column sums is very simple. Let r and c be the vector of row sums and column sums respectively and uv^T is a candidate rank-one matrix. From the sum constraints we have:

$$uv^T \mathbf{1}_{n \times 1} = u \|v\|_1 = r$$

and

$$vu^T \mathbf{1}_{m \times 1} = v \|u\|_1 = c,$$

or u and v are parallel to r and c. Moreover,

$$(\mathbf{1}_{n \times 1}^T uv^T \mathbf{1}_{n \times 1}) = \|u\|_1 \|v\|_1 = r^T \mathbf{1}_{n \times 1} = \mathbf{1}_{n \times 1}^T c$$

we have $uv^T = \frac{rc^T}{\|c\|_1}$. This implies that the rank-one solution to Problem 5.1 is independent of the choice of cost measure and is always

$$(u, v) = \frac{1}{\|c\|_1}(\alpha r, \frac{1}{\alpha}c) \quad \text{with } \alpha > 0.$$

For higher ranks, one could eliminate these sum constraints by rewriting a chosen variable pair (u_k, v_k) as a function of other variables. But this makes the cost function really complicated to analyze. In this case, one might better use a generic optimization tool.

In the context of nonnegative matrix with predefined row and column sums, the Sinkhorn algorithm [107] is well-known for the diagonal scaling problem: given a nonnegative square matrix A and two nonnegative vectors r and c ($\sum_i r_i = \sum_i c_i$), finding two positive diagonal matrix D_1 and D_2 with appropriate size such that the scaled matrix $B = D_1 A D_2$ satisfies $B\mathbf{1} = r$ and $B^T\mathbf{1} = c$. This means that the scaled matrix has c and r as the vectors of column sums and row sums, respectively. The process is very simple and consists of alternatively scaling the rows and columns of A. A necessary and sufficient condition for the convergence was also given by the author.

When the Sinkhorn algorithm converges, it produces a matrix B with the predefined row sums r and column sums c. It was shown in [71] that, among matrices with the predefined row sums r and column sums c, B is the closest to A with respect to the generalized Kullback-Leibler divergence:

$$D(B\|A) = \sum_{ij}\left[B_{ij}\log\frac{B_{ij}}{A_{ij}} - B_{ij} + A_{ij}\right], \tag{5.1}$$

where by convention, we put $\frac{0}{0} = 0$ and $0log0 = 0$. Moreover, given a matrix A and vectors r and c, B is unique.

The Kullback-Leibler divergence [78] is well-known in probability theory and information theory. It is also known as *information divergence*, *information gain* or *relative entropy*, that measures the difference between two probability distributions. Its generalized version above measures the difference between two nonnegative vectors, which is an instance of the Bregman divergences, see for example [91]. These divergences are similar to a distance but do not satisfy the triangle inequality nor

symmetry, except for Euclidean Distance, an important element of the class.

We will show that, when using this measure to approximate a nonnegative matrix, the row and column sums of the input matrix will be retained in the low-rank approximation. Note that the use of the generalized Kullback-Leibler divergence in the diagonal scaling problem above and the problem in the next section are different. For the earlier, the input matrix is the second parameter of the divergence, i.e. $D(.\|A)$. While it is the second parameter, i.e. $D(A\|.)$, in the nonnegative matrix factorization with the generalized Kullback-Leibler divergence, described in this chapter.

5.2 Generalized KL divergence in NMF

In this section, we focus on the use of the generalized Kullback-Leibler (KL) divergence in the problem of nonnegative matrix factorization. We will show that when using the generalized KL divergence as the cost function for NMF, row sums and column sums of the original matrix are preserved in the approximation. We will use this special characteristic in several approximation problems.

It is known that the generalized Kullback-Leibler divergence (5.1) is always nonnegative and zero if and only if $B = A$, [91]. It is not symmetric, i.e. $D(A\|B) \neq D(B\|A)$ and does not satisfy the triangle equation. Figures 5.1 and 5.2 show the level sets of two measures $D(x\|(2,5))$ and $D((2,5)\|x)$.

5.2.1 Row and column sums preserving approximation

The choice of cost function F of course affects the solution of the minimization problem. The nonnegative matrix factorization problem with the generalized Kullback-Leibler divergence as the new cost function is

$$\min_{U_+\in\mathbb{R}^{m\times r}\ V_+\in\mathbb{R}^{n\times r}} D(A\|UV^T) \tag{5.2}$$

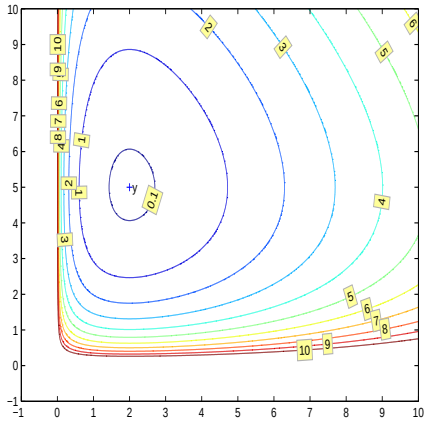


Figure 5.1: $D(y|x)$ *where* $y = (2,5)$

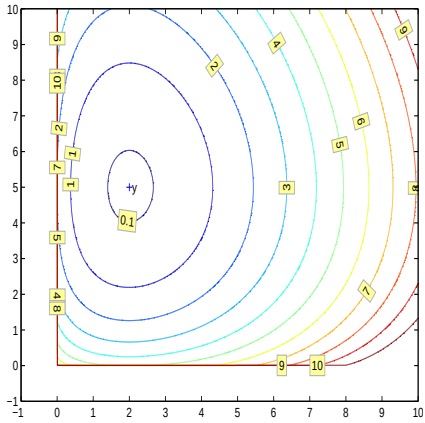


Figure 5.2: $D(x|y)$ *where* $y = (2,5)$

where $A \geq 0$ and r is the reduced inner rank. For this divergence,the gradients are easy to construct (see [80, 81]) :

$$\nabla_{U_{ij}} D(A\|UV^T) = -\sum_k \frac{A_{ik}}{[UV^T]_{ik}} V_{kj} - V_{kj}, \tag{5.3}$$

$$\nabla_{V_{ij}} D(A\|UV^T) = -\sum_k \frac{A_{ki}}{[UV^T]_{ki}} U_{kj} - U_{kj}. \tag{5.4}$$

The Karush-Kuhn-Tucker (KKT) optimality conditions (see Section 1.2.3) can be established for Problem (5.2):

$$U \geq 0 \quad , \quad V \geq 0, \tag{5.5}$$

$$\nabla_U D(A\|UV^T) \geq 0 \quad , \quad \nabla_V D(A\|UV^T) \geq 0, \tag{5.6}$$

$$U \circ \nabla_U D(A\|UV^T) = 0 \quad , \quad V \circ \nabla_V D(A\|UV^T) = 0. \tag{5.7}$$

Remark: Although by using two conventions $\frac{0}{0} = 0$ and $0log0 = 0$, the generalized KL divergence (5.1) and its gradients (5.3) (5.4) are extended continuously to points where $A_{ij} = 0$ for some (i,j), these functions are still not defined when $A_{ij} > 0$ and $[UV^T]_{ij} = 0$. However, even when $UV^T > 0$ at a stationary point, U_{ij} and V_{ij} do not have to be restricted to only *strictly positive* values. This implies that nonnegativity

constraints on U_{ij} and V_{ij} can be active and the above KKT conditions are still available.

It is important to note that the cost function $D(A|UV^T)$ is convex in each of the factors U and V, but it is not convex in the two factors at the same time, hence the problem can have many local minima. A simple iteration proposed by Lee and Seung [80] is similar to that for the Euclidean distance:

$$V \leftarrow \frac{[V]}{[\mathbf{1}_{m\times n}U]} \circ \left(\frac{[A^T]}{[VU^T]} U \right), \quad U \leftarrow \frac{[U]}{[\mathbf{1}_{n\times m}V]} \circ \left(\frac{[A]}{[UV^T]} V \right). \quad (5.8)$$

The convergence of these update rules are investigated in [42] and [85]. In the next section, we will investigate the row sums and the column sums of the stationary points.

5.2.2 Stationary points

In this section, we use the optimality conditions (5.5) and (5.7) to show a particular property of the stationary points of NMF using the generalized KL divergence.

Theorem 5.2. *Let $A_{m\times n}$ a non-negative matrix. Then every stationary point (U, V) of the cost function in (5.2) preserves the column sums of A ($\mathbf{1}_{1\times m}A = \mathbf{1}_{1\times m}(UV^T)$), the row sums of A ($A\mathbf{1}_{n\times 1} = (UV^T)\mathbf{1}_{n\times 1}$) and the matrix sum of A ($\mathbf{1}_{1\times m}A\mathbf{1}_{n\times 1} = \mathbf{1}_{1\times m}(UV^T)\mathbf{1}_{n\times 1}$).*

Proof. At a stationary point, from (5.7), the matrix V must satisfy the following optimality condition

$$V_{ij}\sum_k \frac{A_{ki}}{[UV^T]_{ki}} U_{kj} = V_{ij}\sum_k U_{kj}, \quad \forall i, j.$$

Calculating the sum over j of the left-hand side matrix gives:

$$\sum_j V_{ij}\sum_k \frac{A_{ki}}{[UV^T]_{ki}} U_{kj} = \sum_k (\sum_j V_{ij}U_{kj}) \frac{A_{ki}}{[UV^T]_{ki}} = \sum_k A_{ki},$$

and the sum over j of the right-hand side matrix gives:

$$\sum_j V_{ij}\sum_k U_{kj} = \sum_k \sum_j V_{ij}U_{kj} = \sum_k [UV^T]_{ki}.$$

This implies that $\sum_k A_{ki} = \sum_k [UV^T]_{ki}$ or $\mathbf{1}_{1\times m}A = \mathbf{1}_{1\times m}(UV^T)$. For the row sums, one can easily prove the equality by the same development using the optimality condition of V. The matrix sum is preserved as a consequence of the preservation of column sums or row sums. □

Using the above theorem, one obtains the following *standard form* for every stationary point of the KL divergence iteration.

Corollary 5.3. *Let $A_{m\times n}$ be a non-negative matrix. Every stationary point $(U_{m\times r}, V_{n\times r})$ of the KL minimization problem has the form*

$$UV^T = P_{m\times r}D_{r\times r}Q^T_{n\times r},$$

where P, Q are column stochastic, D is diagonal non-negative, and $\sum_i D_{ii} = \sum_{ij} A_{ij}$. Furthermore, if A is column stochastic (or row stochastic) then the matrix DQ^T (or PD) are also column stochastic (or row stochastic).

Proof. Define the normalization factors D_U and D_V as the column sums of U and V respectively. Then there exist column stochastic matrices P and Q such that $PD_U = U$, $QD_V = V$. These matrices are obtained by dividing the respective columns by their non-zero column sums, and by choosing an arbitrary stochastic column in P or Q if the corresponding column sum in U or V was zero. Define $D = D_U D_V$, then $\sum_i D_{ii} = \sum_{ij} A_{ij}$ follows since P and Q are column stochastic. Moreover, PDQ^T is easily shown to preserve the matrix sum of A.

It is straightforward then that the column sums of DQ^T and row sums of PD are those of A, which also proves the last assertion for stochastic matrices. □

Furthermore, if there exists a zero column in U (or V), one can remove that zero column in U (or V) and the corresponding column in V (or U) without changing the product UV^T. This amounts to saying that one can also obtain a reduced rank factorization of the same type, in which the diagonal elements of D are restricted to be all strictly positive. By writing the stationary point in this form, one can compare this with the Singular Value Decomposition (SVD) of a matrix $A = U\Sigma V^T$ where the orthogonal matrices U and V are replaced by column stochastic matrices P and Q.

In particular, if the reduced rank k is 1, it follows from Theorem 5.2 that we can have a unique global minimizer:

$$\hat{A} = \sigma u v^t \tag{5.9}$$

where $\sigma = \sum_{i,j} A_{ij}$, $u_j = \sum_j A_{ij}/\sigma$ and $v_i = \sum_i A_{ij}/\sigma$. And if the rank k is equal to $min(m, n)$ we have a trivial solution which is $(U = A, V = I_n)$ or $(U = I_m, V = A^T)$.

If we consider Problem (5.2) for a non-negative matrix with unit element sum (i.e. $\sum_{i,j} A_{ij} = 1$) then the stationary points are in fact solutions of the Probabilistic Latent Semantic Analysis (pLSA) [65] which is used in document classification. The link between pLSA and Problem (5.2) was first pointed out in [64]. The pLSA problem is then to find a low-rank joint-probability matrix that approximates a full rank or higher rank joint-probability matrix $A/(\sum_{i,j} A_{ij})$ using the generalized Kullback-Leibler divergence.

5.2.3 Initialization and normalization

We can prove that the scaling α that minimizes $D(a, \alpha b)$, where a, b are 2 positive vectors, is $\|a\|_1/\|b\|_1$ or, equivalently, when $\|\alpha b\|_1 = \|a\|_1$. This implies that all the normalizations that make the row sums and/or the column sums of the approximation equal to those of the target matrix improve the approximation. When an approximation has the row and column sums of the target matrix, it is called *sum compatible*. Furthermore, the column and row sums of the input matrix are preserved at the stationary points as shown in the previous section. Therefore, it would be natural to start any algorithm with an initial sum compatible approximation. One could randomly choose two matrices U_0 and V_0, then apply the Sinkhorn or faster algorithms, for example in [75], to prescale the product $U_0 V_0^T$ to be sum compatible. Additional scaling could be also carried out after each update of variables. This seems to improve the convergence rate, if we ignore the computation load for the scaling, but that may be very expensive. An affordable alternative is to perform one scaling step after each main iteration. Indeed, this is what the multiplicative update rules (5.8) do.

We will show that each update rule for U or V (5.8) scales automatically the row sums or column sums of the new approximation to those

of the target matrix.

Lemma 5.4. *The update rule*

$$\bar{V} \leftarrow \frac{[V]}{[\mathbf{1}_{m\times n}U]} \circ \left(\frac{[A^T]}{[VU^T]}U \right)$$

scales the column sums of $U\bar{V}^T$ to the column sums of A.

Proof. We rewrite the update as

$$\bar{V} \circ \mathbf{1}_{m\times n}U \leftarrow V \circ \left(\frac{[A^T]}{[VU^T]}U \right)$$

or

$$\bar{V}_{ij}\sum_k U_{kj} \leftarrow V_{ij}\sum_k \frac{A^T_{ik}}{[VU^T]_{ik}}U_{kj}.$$

Summing over j from the two sides

$$\sum_j \bar{V}_{ij}\sum kU_{kj} \leftarrow \sum_j V_{ij}\sum_k \frac{A^T_{ik}}{[VU^T]_{ik}}U_{kj}$$

and rearranging the summations

$$\sum_k\sum_j \bar{V}_{ij}U_{kj} \leftarrow \sum_j\sum_k V_{ij}U_{kj}\frac{A^T_{ik}}{[VU^T]_{ik}}$$

gives

$$\sum_k [\bar{V}U^T]_{ik} \leftarrow \sum_k [VU^T]_{ik}\frac{A^T_{ik}}{[VU^T]_{ik}} = \sum_k A^T_{ik}.$$

□

A similar lemma for the update of U shows that the updated approximation $\bar{U}V^T$ has the row sums of A. Alternatively iterating the update of U and V, the procedure somehow does Sinkhorn scalings while minimizing the generalized Kullback-Leibler divergence. Another

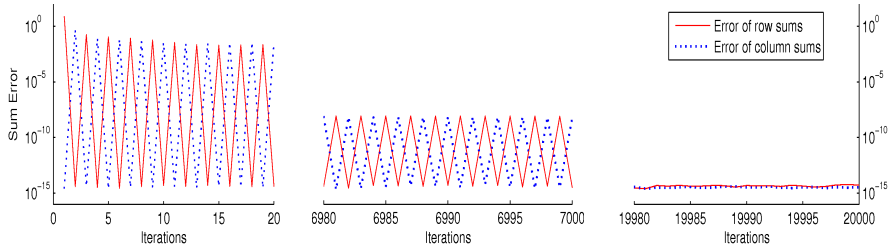


Figure 5.3: Multiplicative updates: Error of row sums $\|A\mathbf{1}_{n\times 1} - UV^T\mathbf{1}_{n\times 1}\|$ *vs. Error of column sums* $\|A^T\mathbf{1}_{m\times 1} - VU^T\mathbf{1}_{m\times 1}\|$.

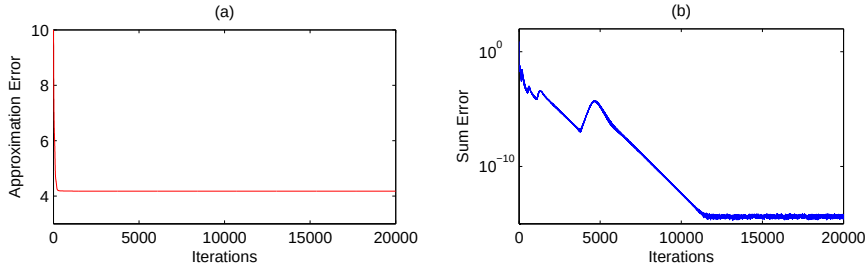


Figure 5.4: Multiplicative updates. (a): approximation error and (b): maximum between error of row sums and error of column sums at each iterations.

consequence is that, in fact, the generalized Kullback-Leibler divergence can be reduced to

$$\sum_{ij} A_{ij} \log \frac{A_{ij}}{[UV^T]_{ij}}$$

since after each update from (5.8) $\sum_{ij} A_{ij} = \sum_{ij} [UV^T]_{ij}$. The results of above lemma is illustrated in Figure Figure 5.3 and Figure 5.4, where a random matrix is approximated using the multiplicative updates. Figure 5.3 shows the evolution of the error of row sums $\|A\mathbf{1}_{n\times 1} - UV^T\mathbf{1}_{n\times 1}\|$ and column sums $\|A^T\mathbf{1}_{m\times 1} - VU^T\mathbf{1}_{m\times 1}\|$. Since the updates alternatively minimize one of the error (row sums or column sums), the other error (column sums or row sums) will be higher. The maximum between them at each iteration is shown in Figure 5.4b. We see that as the algorithm converges (see Figure 5.4a), the sums also converge.

5.3 Application: stochastic matrix approximation

If the input matrix A is stochastic, the stochastic matrix UV^T that minimizes the generalized KL divergence is called a low-rank stochastic approximation of A. Theorem 5.2 shows that if the input matrix A is column stochastic (or row stochastic or doubly stochastic), the stationary points UV^T are actually column stochastic (or row stochastic or doubly stochastic). In other words, the stochasticity of the original matrix is naturally preserved in the approximation.

Using the iterative algorithm in [81], one can numerically obtain a solution for the stochastic matrix approximation problem. Here are some examples of stationary points that are candidates for a solution:

- column stochastic matrix

$$A = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{1}{2} \end{bmatrix} \approx PDQ^T = \begin{bmatrix} 0 & \frac{2}{3} \\ 0 & \frac{1}{3} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{3}{2} & 0 \\ 0 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & 0 & \frac{1}{3} \end{bmatrix}, \tag{5.10}$$

- row stochastic matrix

$$\begin{aligned} A = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix} &\approx PDQ^T \\ &= \begin{bmatrix} 0 & \frac{3}{5} \\ \frac{3}{4} & 0 \\ \frac{1}{4} & \frac{2}{5} \end{bmatrix} \begin{bmatrix} \frac{4}{3} & 0 \\ 0 & \frac{5}{3} \end{bmatrix} \begin{bmatrix} 0 & \frac{5}{8} & \frac{3}{8} \\ \frac{7}{10} & 0 & \frac{3}{10} \end{bmatrix}, \end{aligned} \tag{5.11}$$

- and doubly stochastic

$$\begin{aligned} A = \begin{bmatrix} \frac{3}{8} & \frac{1}{4} & \frac{3}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{3}{8} & \frac{1}{4} & \frac{3}{8} \end{bmatrix} &= PDQ^T = PDP^T \\ &= \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}. \end{aligned} \tag{5.12}$$

In the above examples, the approximations are written in the form presented in Corollary 5.3. Especially, in the third example of a doubly stochastic matrix, we have an exact and symmetric factorization of the form PDP^T. In general, it is not easy to find such a symmetric approximation.

Furthermore, instead of considering the ordinary sum of a vector, we can consider the *weighted sum* of a vector x, defined as

$$s_w(x) = \sum_i w_i x_i = x^T w \tag{5.13}$$

where w is a positive weight vector. One can find an approximation that preserves the weighted column sums and row sums of the original matrix. In fact, suppose w_r and w_c are weight vectors with respect to which we want to find a low rank approximation $\tilde{U}\tilde{V}^T$ of A that preserves the weighted row sums and the weighted column sums respectively, i.e.

$$\tilde{U}\tilde{V}^T w_r = Aw_r \qquad w_c^T \tilde{U}\tilde{V}^T = w_c^T A, \tag{5.14}$$

we can use the following procedure

1. Create $\hat{A} = D_{w_r} A D_{w_c}$,
2. find a low rank non-negative approximation UV^T of $\hat{A}$ by using NMF algorithm for generalized KL divergence,
3. and create the desired approximation $\tilde{U}\tilde{V}^T$ where $\tilde{U} = D_{w_c}^{-1} U$ and $\tilde{V} = D_{w_r}^{-1} V$.

Applying Theorem 5.2, one can easily check that (5.14) does hold for the newly created matrix $\tilde{U}\tilde{V}^T$.

The preservation of weighted column sums and row sums implies that we can use the same procedure to construct a low rank non-negative approximation that preserve the left and right principal eigenvectors of a square non-negative matrix. The technique is then simply to use the left and right principal eigenvectors of the original matrix as the weight vectors described above to construct the desired approximation.

6
WEIGHTS IN NONNEGATIVE MATRIX FACTORIZATION

A data matrix often comes from a data acquisition process. These processes are sometimes susceptible to the loss of precision or even to the absence of some data. Therefore, when approximating such a matrix, more reliable elements should receive more attention. The reliability can be simply a $0/1$ representing the absence/presence of the data. But it is usually coded as a nonnegative number called the weight. The bigger the weight is, the more important the element is.

In this chapter, we will investigate the problem of *Weighted Nonnegative Matrix Factorization (WNMF)* which minimize the *Weighted Euclidean Distance*:

$$\frac{1}{2}\|A - UV^T\|_W^2 := \frac{1}{2}\sum_{ij}[W \circ (A - UV^T) \circ (A - UV^T)]_{ij} \tag{6.1}$$

where $W = \{W_{ij}\} \geq 0$ is a nonnegative weight matrix. Clearly, we can consider the unweighted version as a particular case of the weighted one where all the elements of the weight matrix are equal to 1.

While extending the Euclidean Distance to the weighted case is rather straightforward, the weighted approximation problem is much more difficult. Even when the nonnegativity constraints are omitted, an analytical solution no longer exists, except when the all-one weight matrix is used. Numerical algorithms usually produce local minima [111].

The problem of Weighted Nonnegative Matrix Factorization was first

stated in [96] for the Weighted Euclidean Distance (6.1). Recently [53], a particular type of weighting was proposed for the generalized Kullback-Leibler divergence as a cost function, in order to vary the importance of each column of the matrix A in the approximation $UVD \approx AD$, where D is a nonnegative diagonal scaling matrix. One can easily see that this nonnegative weight matrix is equivalent to a rank-one weighting matrix W in our weighted formulation.

An approach that allows to use weight matrices in a more general context is given in [111], where an Expectation-Maximization algorithm is used in an iterative scheme that produces an *unweighted* low-rank approximation of a *weighted* combination of a previously computed approximation :

$$(U_{k+1}, V_{k+1}) = LowRank(W \circ A + (1 - W) \circ (U_k V_k)). \qquad (6.2)$$

Here there are *no* constraints of non-negativity, but the same idea can also be used to incorporate weights in an algorithm for nonnegative matrix factorizations. This implies that one has to solve an unweighted low-rank nonnegative approximation at each step of the iteration, and this can become quite inefficient in terms of complexity.

The chapter begins with the gradient information of the weight Euclidean distance that every standard gradient method can use. Then we will extend two particular methods: the multiplicative rules and Rank-one Residue Iteration to take into account the weight matrix. This is followed by a discussion about an interesting link between the weighted Euclidean distance and the weighted generalized Kullback-Leibler divergence in which the weighted multiplicative rules for the latter are also derived. We will see next how the weight matrix can also be added into some existing NMF variants such as: Nonnegative Tensor Factorization, Nonnegative Matrix Factorization with Sparseness constraint, etc. The chapter will end with some numerical experiment on the image data of human faces where weights are used to emphasize meaningful features. Let us start with the gradient information.

6.1 Gradient information

We restate here the problem of Weighted Nonnegative Matrix Factorization:

Problem 6.1 (WNMF)**.**

$$\min_{U\in\mathbb{R}_+^{m\times r}\ V\in\mathbb{R}_+^{n\times r}} \frac{1}{2}\|A-UV^T\|_W^2.$$

The gradient of the weighted cost function can be easily calculated:

$$\begin{aligned}\nabla_U \frac{1}{2}\|A-UV^T\|_W^2 &= \Big(W\circ(UV^T-A)\Big)V\\ \nabla_V \frac{1}{2}\|A-UV^T\|_W^2 &= \Big(W^T\circ(VU^T-A^T)\Big)U\end{aligned}$$

and the KKT optimality conditions can also be derived:

$$U\geq 0 \quad , \quad V\geq 0, \tag{6.3}$$
$$\nabla_U\geq 0 \quad , \quad \nabla_V\geq 0, \tag{6.4}$$
$$U\circ\nabla_U = 0 \quad , \quad V\circ\nabla_V = 0. \tag{6.5}$$

Using this gradient information, one can apply the gradient based methods presented in Chapter 3: the projected gradient with line search and the projected first-order approximation to solve Problem 6.1. Furthermore, in the following sections, weights will be also added in the multiplicative rules and the rank-one residue approximation scheme, as we will see in the next two sections.

6.2 Methods

We choose to extend two methods: the multiplicative rules and the rank-one residue iteration to the weighted case because the former will help us to discover an surprising link, as we will see, and the latter is a very flexible, which can be extended to many problems. Furthermore, the rank-one residue iteration outperforms the other ones in the numerical experiments carried out in Chapter 4. We begin with the already popular one, the multiplicative rules.

6.2.1 Weighted multiplicative updates

The multiplicative updates [80] are simple algorithms for NMF problem. Although there convergence are not fully understood, they still can be

generalized to the weighted case. We first need the following simple lemma:

Lemma 6.2. *Let A be a symmetric nonnegative matrix and v be a positive vector, then the matrix $\hat{A} = diag\left(\frac{[Av]}{[v]}\right) - A$ is positive semi-definite.*

Proof. It is easy to see that $diag\left(\frac{[Av]}{[v]}\right) = D_v^{-1} D_{Av}$. The scaled version $\hat{A}_s := D_v \hat{A} D_v$ of $\hat{A}$ satisfies $\hat{A}_s = D_{Av} D_v - D_v A D_v$ and is a *diagonally dominant* matrix since $\hat{A}_s \mathbf{1}_m = (Av) \circ v - v \circ (Av) = 0$ and its off-diagonal elements are negative. Therefore, the matrix $\hat{A}_s$ is positive semi-definite, and so is $\hat{A}$. □

The following theorem gives the multiplicative update rules for the Problem 6.1.

Theorem 6.3. *The weighted Euclidean distance $\frac{1}{2}\|A - UV^T\|_W^2$ is non-increasing under the updating rules:*

$$V \leftarrow V \circ \frac{[(W \circ A)^T U]}{[(W \circ (UV^T))^T U]}, \qquad U \leftarrow U \circ \frac{[(W \circ A)V]}{[(W \circ (UV^T))V]}. \quad (6.6)$$

The weighted Euclidean distance $\frac{1}{2}\|A - UV^T\|_W^2$ is invariant iff the conditions (6.3) and (6.5) hold.

Proof. We only treat the updating rule for V since that of U can be proved in a similar fashion. First, we point out that the cost $F(A, UV^T)$ splits in n independent problems related to each column of the error matrix. We can therefore consider the partial cost function for a single column of A and W, which we denote by a and w, respectively:

$$F(v) = F_w(a, Uv) = \frac{1}{2}\sum_i (w_i(a_i - [Uv]_i)^2) \quad (6.7)$$

$$= \frac{1}{2}(a - Uv)^T D_w (a - Uv) \quad (6.8)$$

where $D_w = diag(w)$ and v is the transpose of the corresponding row of V. Let v^k be the current approximation of the minimizer of $F(v)$ then one can rewrite $F(v)$ as the following quadratic form:

$$F(v) = F(v^k) + (v - v^k)^T \nabla_v F(v^k) + \frac{1}{2}(v - v^k)^T U^T D_w U (v - v^k) \quad (6.9)$$

where $\nabla_v F(v^k)$ is explicitly given by

$$\nabla_v F(v^k) = -U^T D_w (a - U v^k). \tag{6.10}$$

Next, we approximate $F(v)$ by a simpler quadratic model:

$$G(v, v^k) = F(v^k) + (v - v^k)^T \nabla_v F(v^k) + \frac{1}{2}(v - v^k)^T D(v^k)(v - v^k) \tag{6.11}$$

where $G(v^k, v^k) = F(v^k)$ and $D(v^k)$ is a diagonal matrix chosen to make $D(v^k) - U^T D_w U$ positive semi-definite implying that $G(v, v^k) - F(v) \geq 0, \forall v$. The choice for $D(v^k)$ is similar to that proposed by Lee and Seung:

$$D(v^k) = diag\left(\frac{[U^T D_w U v^k]}{[v^k]}\right). \tag{6.12}$$

Lemma 6.2 assures the positive semi-definiteness of $D(v^k) - U^T D_w U$. As a result, we have

$$F(v^k) = G(v^k, v^k) \geq \min_v G(v, v^k) = G(v^{k+1}, v^k) \geq F(v^{k+1}) \tag{6.13}$$

where v^{k+1} is found by solving $\frac{\partial G(v, v^k)}{\partial v} = 0$:

$$\begin{aligned}
v^{k+1} &= v^k - D(v^k)^{-1} \nabla F(v^k) && (6.14)\\
&= v^k + diag\left(\frac{[v^k]}{[U^T D_w U v^k]}\right) U^T D_w (a - U v^k) && (6.15)\\
&= v^k + v^k \circ \frac{[U^T D_w (a - U v^k)]}{[U^T D_w U v^k]} && (6.16)\\
&= v^k \circ \frac{[U^T D_w a]}{[U^T D_w U v^k]} && (6.17)\\
&= v^k \circ \frac{[U^T (w \circ a)]}{[U^T (w \circ (U v^k))]}. && (6.18)
\end{aligned}$$

Transposing the two sides of this gives the update rule for each row of V.

$$(v^{k+1})^T = (v^k)^T \circ \frac{[(w \circ a)^T U]}{[(w \circ (U v^k))^T U]}.$$

Putting together the updating rules for all the rows of V yields the desired result for the whole matrix V in (6.6). The relation (6.13) shows that the weighted Euclidean distance is non-increasing under the updating rule for V, and (6.14) show that $v^{k+1} = v^k$ if and only if $v^k \circ \nabla F(v^k) = 0$. Finally, the non-negativity of v^k is automatically satisfied. □

Similar to the original version of the multiplicative update rules, the above update rules suffer the same problem when approaching the boundary of the nonnegative orthant. When using this, one should start the iteration from the interior of the nonnegative orthant and be cautious when zeros appear in U and V.

6.2.2 Weighted Rank-one Residue Iteration

In Chapter 4, employing the rank partition of the NMF problem offers a simple coordinate descent algorithm for the NMF problem. Here, we will modify that algorithm to incorporate the weight matrix in the cost function. The Weighted Nonnegative Matrix Factorization can be restated as below:

Problem 6.4 (WNMF)**.**

$$\min_{u_i \geq 0 \; v_i \geq 0} \frac{1}{2} \| A - \sum_{i=1}^{r} u_i v_i^T \|_W^2$$

where W is a nonnegative weight matrix.

To construct the update for v_t, $t = 1, 2, \ldots, r$, we fix all variables except v_t. Again, we create the residue matrix:

$$R_t = A - \sum_{i \neq t} u_i v_i^T$$

to restrict Problem 6.4 only to the variable u_t

$$\min_{v \geq 0} \frac{1}{2} \| R_t - u_t v^T \|_W^2. \tag{6.19}$$

We have

$$\begin{aligned}\|R_t - u_t v^T\|_W^2 &= \sum_{ij} W_{ij}([R_t]_{ij} - [u_t]_i v_j)^2 \\ &= \sum_{ij} W_{ij}\left([R_t]_{ij}^2 - 2[R_t]_{ij}[u_t]_i v_j + ([u_t]_i v_j)^2\right) \\ &= \|R_t\|_W^2 - 2\sum_j v_j\left(\sum_i W_{ij}[R_t]_{ij}[u_t]_i\right) \\ &\quad + \sum_j v_j^2\left(\sum_i W_{ij} u_i^2\right) \\ &= \|R_t\|_W^2 - 2v^T(W \circ R_t)^T u_t + (v \circ v)^T W^T(u_t \circ u_t).\end{aligned}$$

From this formulation, one now can derive the following similar lemma to the one in Chapter 4.

Lemma 6.5. *If* $[(W \circ R_t)^T u_t]_+ \neq 0$ *and* $W^T(u_t \circ u_t) > 0$, *then*

$$v_* := \frac{[(W \circ R_t)^T u_t]_+}{W^T(u_t \circ u_t)}$$

is the unique global minimizer of (6.19) and the value of the cost function equals

$$\frac{1}{2}\|R_t - u_t v^{*T}\|_W^2 = \frac{1}{2}\left(\|R_t\|_W^2 - \left(\frac{[(W \circ R_t)^T u_t]_+}{W^T(u_t \circ u_t)}\right)^T [(W \circ R_t)^T u_t]_+\right).$$

Proof. Let us permute the elements of the vectors $x := (W \circ R_t)^T u_t$, $y := W^T(u_t \circ u_t)$ and v such that

$$Px = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad Py = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \text{and} \quad Pv = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

with $x_1 \geq 0$, $x_2 < 0$ and P is the permutation matrix. Then

$$\|R_t - u_t v^T\|_W^2 = \|R_t\|_W^2 - 2v_1^T x_1 - 2v_2^T x_2 + (v_1^T D_{y_1} v_1 + v_2^T D_{y_2} v_2).$$

Since $x_2 < 0$, $v_2 \geq 0$ and $y > 0$, it is obvious that $\|R_t - u_t v^T\|_F^2$ can only be minimal if $v_2 = 0$. Our assumption implies that x_+ is nonempty and

$x_1 \geq 0$. One can then find the optimal v_1 by minimizing the remaining quadratic function

$$\|R_t\|_F^2 - 2v_1^T x_1 + v_1^T D_{y_1} v_1$$

which yields the solution $v_1 = \frac{[x_1]}{[y_1]}$. Putting the two components together yields the result

$$v_* = \frac{[x_+]}{[y]} \quad \text{and} \quad \|R_t - u_t v_*^T\|_W^2 = \|R_t\|_W^2 - \left(\frac{[x_+]}{[y]}\right)^T x_+.$$

□

Remark 1: The above lemma has of course a dual form, where one fixes v_t but solves for the optimal u to minimize $\|R_t - uv_t^T\|_W^2$. These lemmas would yield the updating rules

$$v_t \leftarrow \frac{[(W \circ R_t)^T u_t]_+}{W^T(u_t \circ u_t)} \quad \text{and} \quad u_t \leftarrow \frac{[(W \circ R_t)v_t]_+}{W(v_t \circ v_t)} \tag{6.20}$$

which can be used to recursively update approximations $\sum_{i=1}^r u_i v_i^T$ by modifying each rank-one matrix $u_t v_t^T$ in a cyclic manner. We call this method the *Weighted Rank-one Residue Iteration* (WRRI), which is shown in Algorithm 10.

Remark 2: In the case where $[(W \circ R_t)^T u_t]_+ = 0$, we have a trivial solution for $v = 0$ that is not covered by Lemma 6.5. This solution is unique only when $u_t \neq 0$. This will result in a rank-deficient approximation. To avoid that, one can replace $u_t v_t^T$ by any rank-one approximation that reduces the $\|.\|_W$ norm of the error matrix, such as

$$u_* = [(R_t \circ W)_{i^*:}]_+ \qquad v_* = e_{i^*}$$

where $i^* = \text{argmax}_i \|[(R_t \circ W)_{i:}]_+\|$.

Remark 3: If $[W^T(u_t \circ u_t)]_i = 0$ for some i and because $W \geq 0$ and $u_t \geq 0$, one can conclude that $[(W \circ R_t)^T u_t]_i = 0$. This means that the cost function is unchanged irrespective of the value of v_i^*. We can simply set those elements of v^* to zero to minimize $\|v^*\|_2$.

Algorithm 10 (WRRI)

Initialize u_i's, v_i's, for $i = 1$ to r
repeat
  for $i = 1$ to r **do**
    $R_i = A - \sum_{j \neq i} u_j v_j^T$

    for $j = 1$ to n **do**
      if $[W^T(u_i \circ u_i)]_j > 0$ **then**
        $[v_i]_j \leftarrow \frac{\left([(W \circ R_i)^T u_i]_+\right)_j}{[W^T(u_i \circ u_i)]_j}$
      else
        $[v_i]_j \leftarrow 0$
      end if
    end for

    for $j = 1$ to m **do**
      if $[W(v_i \circ v_i)]_j > 0$ **then**
        $[u_i]_j \leftarrow \frac{([(W \circ R_i)v_i]_+)_j}{[W(v_i \circ v_i)]_j}$
      else
        $[u_i]_j \leftarrow 0$
      end if
    end for
  end for
until Stopping condition

In practice, an extra normalization step can be applied as in Algorithm 9 to avoid some numerically unstable situations. In out experiments, the WRRI method outlined in Algorithm 10 always converges. But the convergence proof is not trivial due to the uniqueness problems described in Remarks 2 and 3. We reuse the *damping technique*, first used in Chapter 4, to yield a *damped WRRI* method which is shown in Algorithm 11.

The following result guarantees the convergence to a stationary point of Algorithm 11.

Theorem 6.6. *Every limit point of Algorithm 11 is a stationary point of*

Algorithm 11 (Damped WRRI)

1: Initialize u_i's, v_i's, for $i = 1$ to r
2: **repeat**
3: **for** $i = 1$ to r **do**
4: $R_i = A - \sum_{j \neq i} u_j v_j^T$
5: $v_i \leftarrow \frac{[(W \circ R_i)^T u_i + \psi v_i]_+}{W^T(u_i \circ u_i) + \psi \mathbf{1}_{m \times 1}}$
6: $u_i \leftarrow \frac{[(W \circ R_i) v_i + \psi u_i]_+}{W(v_i \circ v_i) + \psi \mathbf{1}_{n \times 1}}$
7: **end for**
8: **until** Stopping condition

Problem 6.4.

Proof. Reusing the argument in Theorem 4.7 from Chapter 4 yields the desired result. □

In Algorithm 11, when a zero component appears, one could replace $u_t v_t^T$ by any rank-one approximation that reduces the weighted norm of the error matrix. Patches described above should be applied only in a finite number of times to guarantee the convergence of the algorithm.

Having a sound convergence property is a major advantage of the *damped WRRI* method over the weighted multiplicative rules presented in the preceding section. But by studying these multiplicative rules, we will show an interesting link between the two different measures, as presented in the next section.

6.3 Toward the weighted KL divergence

In the previous chapter, when talking about the problem of Nonnegative Matrix Factorization with fixed column and row sums, the generalized Kullback-Leibler divergence was introduced. Looking at one of the KKT optimality conditions for the two measures: the weighted Euclidean distance

$$V \circ ((W_1 \circ (UV^T - A))^T U) = 0 \tag{6.21}$$

and the weighted KL divergence

$$V \circ ((W_2 \circ (\mathbf{1}_{m\times n} - \frac{[A]}{[UV^T]}))^T U) = 0, \tag{6.22}$$

it is easy to see that if $W_1 = \frac{[W_2]}{[UV^T]}$, these two conditions are identical, where an additional assumption $UV^T > 0$ has to be inserted. This means that, any stationary point given by an algorithm for weighted Euclidean distance is a stationary point of a corresponding weighted generalized KL divergence. This can also be seen when extending the multiplicative updating rules [80] for the generalized KL divergence to the weighted case. For the sake of completeness, we will include following theorem introducing the multiplicative rules for the weighted generalized KL divergence.

Theorem 6.7. *The weighted divergence $D_W(A\|UV)$ is non-increasing under the updating rules :*

$$V \leftarrow \frac{[V]}{[W^T U]} \circ \left(\frac{[W^T \circ A^T]}{[VU^T]} U \right), \quad U \leftarrow \frac{[U]}{[WV]} \circ \left(\frac{[W \circ A]}{[UV^T]} V \right). \tag{6.23}$$

The weighted divergence $D_W(A\|UV)$ is invariant under these updates iff the conditions (5.5) and (5.7) hold.

Proof. Again, we prove the theorem only for V and we also split the divergence into partial divergences corresponding to one column of W and A, denoted by w and a.

$$\begin{aligned} F(v) &= D_w(a\|Uv) \\ &= \sum_i w_i \left(a_i \log a_i - a_i + \sum_j U_{ij} v_j - a_i \log \sum_j U_{ij} v_j \right), \end{aligned} \tag{6.24}$$

where v is the transpose of the corresponding row of V. This partial divergence is approximated by the following auxiliary function:

$$\begin{aligned} G(v, v^k) &= \sum_i \left(w_i \left(a_i \log a_i - a_i + \sum_j U_{ij} v_j \right. \right. \\ &- \left. \left. a_i \sum_j \frac{U_{ij} v_j^k}{\sum_l U_{il} v_l^k} \left(\log U_{ij} v_j - \log \frac{U_{ij} v_j^k}{\sum_l U_{il} v_l^k} \right) \right) \right). \end{aligned} \tag{6.25}$$

Because of the convexity of the function $-log(x)$ and since $\sum_j \frac{U_{ij}v_j^k}{\sum_l U_{il}v_l^k} = 1$, we have

$$\begin{aligned} -a_i \log \sum_j U_{ij}v_j &= -a_i \log \left(\sum_j \frac{U_{ij}v_j^k}{\sum_l U_{il}v_l^k} U_{ij}v_j \frac{\sum_l U_{il}v_l^k}{U_{ij}v_j^k} \right) \\ &\leq -a_i \sum_j \frac{U_{ij}v_j^k}{\sum_l U_{il}v_l^k} \log \left(U_{ij}v_j \frac{\sum_l U_{il}v_l^k}{U_{ij}v_j^k} \right) \\ &\leq -a_i \sum_j \frac{U_{ij}v_j^k}{\sum_l U_{il}v_l^k} \left(\log U_{ij}v_j - \log \frac{U_{ij}v_j^k}{\sum_l U_{il}v_l^k} \right) \end{aligned}$$

implying $G(v, v^k) \geq F(v), \forall v$. Moreover $G(v^k, v^k) = F(v^k)$, so we obtain:

$$F(v^k) = G(v^k, v^k) \geq \min_v G(v, v^k) = G(v^{k+1}, v^k) \geq F(v^{k+1}) \,. \tag{6.26}$$

To obtain the updating rule, it is sufficient to construct the minimizer of G with respect to v, given by:

$$\frac{\partial G(v, v^k)}{\partial v_j} = \sum_i w_i U_{ij} - \frac{v_j^k}{v_j} \sum_i w_i a_i \frac{U_{ij}}{\sum_l U_{il}v_l^k} = 0. \tag{6.27}$$

Then the minimizer of $G(v, v^k)$ is chosen as the next value of v:

$$v^{k+1} = \frac{[v^k]}{[U^T w]} \circ \left(U^T \frac{[a \circ w]}{[Uv^k]} \right). \tag{6.28}$$

Transposing the two sides of this gives the update rule for each row of V.

$$(v^{k+1})^T = \frac{[(v^k)^T]}{[w^T U]} \circ \left(\frac{[a^T \circ w^T]}{[(v^k)^T U^T]} U \right). \tag{6.29}$$

Putting together the updating rules for all the rows of V gives the desired updating rule for the whole matrix V as in (6.23). The relation (6.26) shows that the weighted divergence is non increasing under the updating rule for V. Using (6.29) and the fact that

$$\nabla F(v^k) = U^T w - U^T \frac{[a \circ w]}{[Uv^k]} \tag{6.30}$$

we can easily see that that $v^{k+1} = v^k$ if and only if $v^k \circ \nabla F(v^k) = 0$. Finally, the non-negativity of v^k is automatically satisfied. □

Having the updating rules for V for the weighted generalized KL divergence, one can rewrite them as follows:

$$V \leftarrow \frac{[V]}{[W^T U]} \circ \left(\frac{[W^T \circ A^T]}{[VU^T]} U \right) = V \circ \left(\frac{\left[\frac{[W^T \circ A^T]}{[VU^T]} U \right]}{\left[\frac{[W^T \circ (VU^T)]}{[VU^T]} U \right]} \right) = V \circ \left(\frac{[(W_{UV^T} \circ A)^T U]}{[(W_{UV^T} \circ (UV^T))^T U]} \right),$$

where $W_{UV^T} = \frac{[W]}{[UV^T]}$. This shows that each update in the weighted generalized KL divergence is equivalent to an update in the weighted Euclidean distance with the weight matrix W_{UV^T}. This is an adaptive weighting since the weights change after each update. And at the stationary point of this minimization, V and U converge to the minimizer of the weighted Euclidean distance for which the weight matrix is exactly W_{UV^T}.

Conversely, one can see that each update in the weighted Euclidean distance with the weight matrix W is equivalent to an update in the weighted generalized KL divergence with the weight matrix $W_{UV^T} = W \circ (UV^T)$. And again, at the stationary point of this minimization, U and V converge to the minimizer of the weighted generalized KL divergence for which the weight matrix is exactly W_{UV^T}.

We summarize all the updating rules and the link between the two minimizations in Table 6.1. In the unweighted case, the matrix $\mathbf{1}_{m \times n}$ is included to make it easier to compare it with the matrices W_1 and W_2 of the weighted case. With our updating rules for the weighted case, we have thus shown that even though the two cost functions are very different, their minimizations are closely related. While the convergence of the multiplicative rules still needs further investigations, we can conclude this section by raising this question: which algorithms for weighted Euclidean distance (or weighted generalized KL divergence) converge to a stationary point when we add the adaptive weights as

	Euclidean Distance (ED)	Generalized KL Divergence (KLD)
NMF	$V \leftarrow V \circ \frac{[(\mathbf{1}_{m\times n} \circ A)^T U]}{[(\mathbf{1}_{m\times n} \circ (UV^T))^T U]}$	$V \leftarrow V \circ \frac{\left[\left(\frac{[(\mathbf{1}_{m\times n} \circ A)^T]}{[VU^T]} U\right)\right]}{[\mathbf{1}_{n\times m} U]}$
WNMF	$V \leftarrow V \circ \frac{[(\mathbf{W_1} \circ A)^T U]}{[(\mathbf{W_1} \circ (VU^T))^T U]}$	$V \leftarrow V \circ \frac{\left[\left(\frac{[(\mathbf{W_2} \circ A)^T]}{[VU^T]} U\right)\right]}{[\mathbf{W_2}^T U]}$
ED $\Updownarrow$ KLD	$\mathbf{W_1} = \frac{[\mathbf{W_2}]}{[UV^T]}$	

Table 6.1: Multiplicative rules for Weighted Nonnegative Matrix Factorization

described above? Clearly, if there is one, it will be a common algorithm for both measures.

6.4 Adding weights to existing NMF variants

The *Nonnegative Matrix Factorization with Sparseness Constraint* in [68] imposes sparseness constraints on the matrices U and V. The algorithm uses two separate steps to achieve this: a gradient-descent step and a sparseness control step. Weights can be easily added in the gradient-descent step by setting the cost function to be the weighted Euclidean distance instead of the unweighted one. The sparseness control step is kept unchanged. The fixed sparsity variant presented in Chapter 4 can also be generalized to incorporate the weight matrix.

For other NMF methods like *Fisher Nonnegative Matrix Factorization* [122], *Localized Nonnegative Matrix Factorization* [83] etc., weighted version of iterative algorithms can also easily be obtained.

Weights can also added into the Rank-one Residue Iteration for the

Nonnegative Tensor Factorization by the same manner.

6.5 Application: feature extraction of face images

In [80] Lee and Seung argued that there is a link between human perception and nonnegative data representation. The intuition behind this is that perception is based on a representation that is additive and tends to expose parts of the data. Since then, many researchers have tried to use nonnegative representations of data – such as NMF – in many application areas.

One of the major application of NMF is the representation of human faces. In this section, we show the results of two numerical experiments on human faces. These experiments also illustrate the effect of weights on the obtained approximation.

6.5.1 Experiment settings

Again, we use the Cambridge ORL face database as the input data for the following experiments. Ten randomly chosen images are shown in the first row of Figure 6.1.

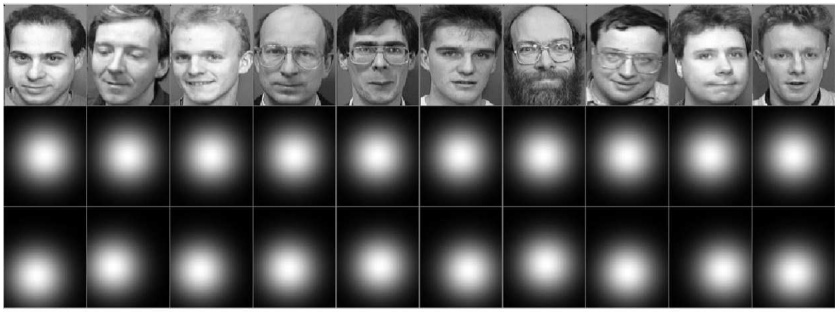


Figure 6.1: Original faces (first row), their image-centered weights W_2 (second row) and their face-centered weights W_3 (last row)

We used three weight matrices of the same size of A (ie. 10304×400).

- **Uniform weight** W_1: a matrix with all elements equal to 1 (i.e. the unweighted case).

- **Image-centered weight** W_2: a nonnegative matrix whose columns are identical, i.e. the same weights are applied to every images. For each image, the weight of each pixel is given by $w_d = e^{-\frac{d^2}{\sigma^2}}$ where $\sigma = 30$ and d is the distance of the pixel to the *center of the image* $(56.5, 46.5)$. This weight matrix has rank one. Ten columns of this matrix are shown in the second row of Figure 6.1.

- **Face-centered weight** W_3: a nonnegative matrix whose columns are *not* identical, i.e. different weights are applied to different images. For each image, the weight of each pixel is given by $w_d = e^{-\frac{d^2}{\sigma^2}}$ where $\sigma = 30$ and d is the distance of the pixel to the *center of the face* in that image, which is manually determined on the tip of the nose. The rank of this matrix is not restricted to one. Ten columns of this matrix are shown in the last row of Figure 6.1.

Next, the matrix A is approximated by nonnegative matrices U and V. The rank chosen for the factorization is 49, the matrices U and V will thus be of dimension 10304×49 and 400×49 respectively. Each column of U is considered as a nonnegative basis vector. The storing space for the approximation will be $(10304 + 400) \times 49$ which is much smaller than 10304×400 for the data matrix A.

6.5.2 NMF versus Weighted NMF

In this experiment, all three weight matrices W_1, W_2 and W_3 are used. For each weight matrix, 49 nonnegative bases, i.e. columns of U, are calculated and shown in Figure 6.2.

Each image in the database can be reconstructed as a weighted sum of these nonnegative bases with nonnegative weights determined by the corresponding row of V. In Figure 6.3, ten selected images are compared with the reconstructed images from the three experiments. The pixel-wise error averages from the three experiments are shown in Figure 6.4.

It can be seen from the results that more important pixels (i.e. those with higher weight, at the center of images or at the center of faces in our example) are better reconstructed than less important ones. This improvement can be seen in both reconstructed images and the pixel-

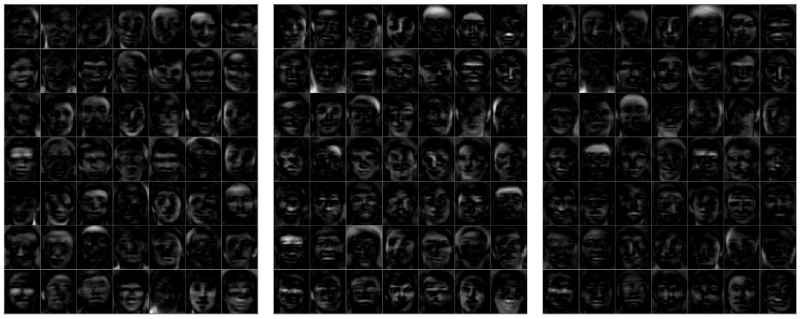


Figure 6.2: Weighted NMF Bases when using: uniform weights (left), image-centered weights (middle) and face-centered weights (right)

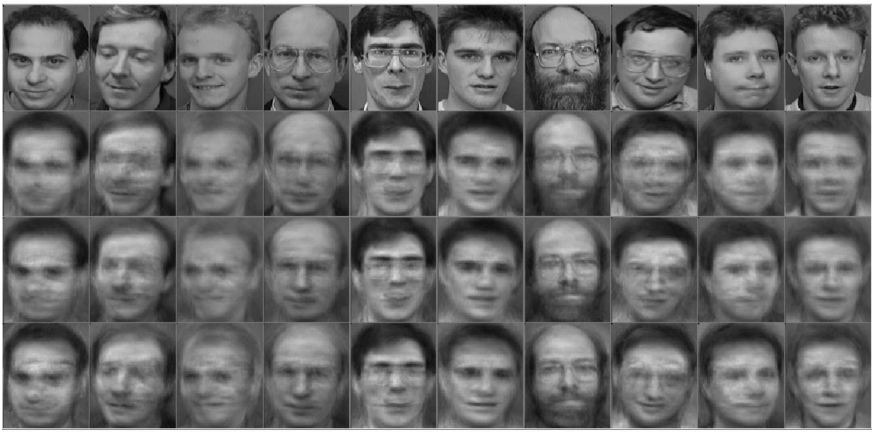


Figure 6.3: Original and reconstructed faces: original (top), using uniform weights (second line), using image-centered weights (third line) and using face-centered weights (bottom)

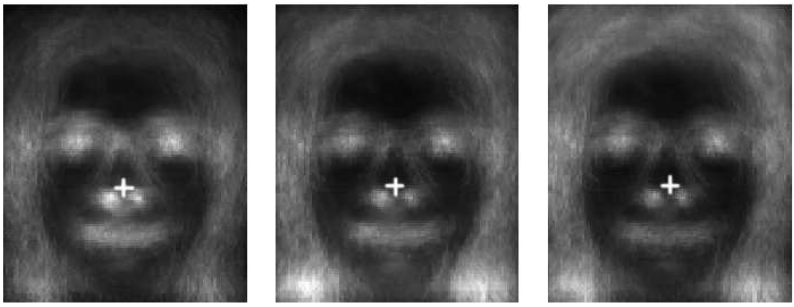


Figure 6.4: Pixel-wise average divergence: unweighted (left), image-centered (middle) and face-centered (right). Brighter colors represent larger errors.

wise average divergence of all the images. In figure 6.4, all the images are shifted to have a common face center. The darker colors correspond to larger errors, which means that the algorithm pays more attention to the center of the images (or to the center of the faces) and that the details at the center areas are privileged in the approximation. More details can be seen on the reconstructed faces when face-centered weights are applied, especially when the center of a face is further away from the center of the image.

The results for weight matrix W_3 also show that our algorithms can deal with weight matrices without rank restriction. And weights can be adapted to each data vector in order to yield better approximations.

6.5.3 NMF versus Weighted NMF with Sparseness Constraint

This second experiment shows the effect of adding weights into the NMF with Sparseness Constraint. Figure 6.5 shows two sets of 49 nonnegative bases obtained by the NMF with Sparseness Constraint with uniform weight W_1 (left) and with face-centered weight W_3 (right).

The NMF with Sparseness Constraint is often used to extract local and independent features on faces. As weights are more centered, more features at the center of faces are retained as we can see in Figure 6.6. This allows us to tune the NMF with Sparseness Constraint algorithm to more relevant parts to give more useful information about the data.

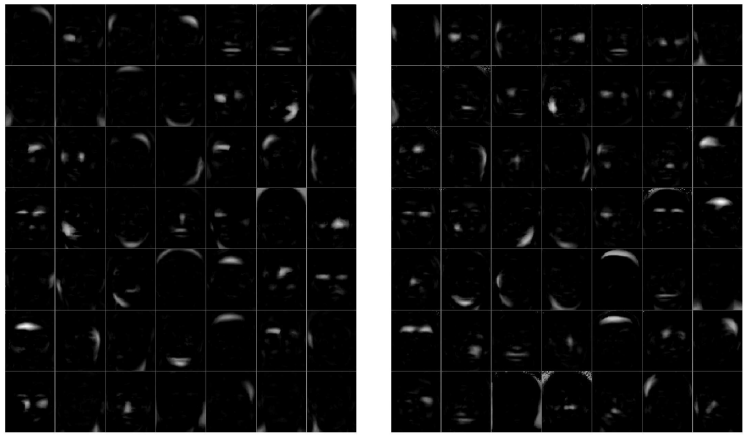


Figure 6.5: Bases of NMF with Sparseness Constraint: unweighted (left) and face-centered (right)

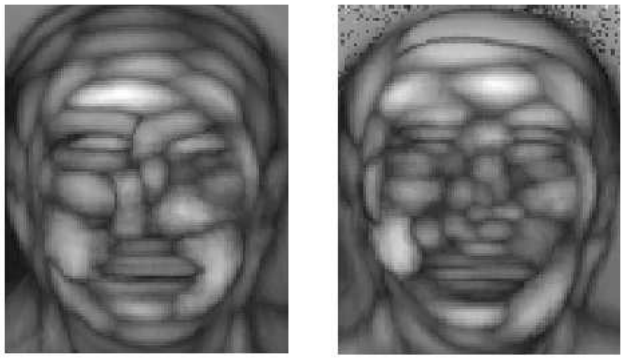


Figure 6.6: Overlapped bases of NMF with Sparseness Constraint: unweighted (left) and image-centered (right). Each pixel in the overlapped image corresponds to the maximum value of all pixels at the same position in all 49 base vectors.

7

SYMMETRY IN NONNEGATIVE MATRIX FACTORIZATION

Many data matrices are symmetric. This is the case for the adjacency matrix of an undirected graph, the matrix of geometric distances between points, correlation matrices, etc. We will investigate in this chapter the symmetry constraint in the Nonnegative Matrix Factorization problem. This leads to two new problems: Symmetric Nonnegative Matrix Factorization and Semi-Symmetric Nonnegative Matrix Factorization.

With the first problem, our objective is not only to obtain the symmetry of the approximation but also a symmetric structure of the corresponding factorization. While the NMF helps to find the UV^T approximation of a nonnegative matrix, its symmetric version tries to find approximations where $U = V$. This allows us to link to other currently open problems related to the *completely positive matrices* [10]. Since many important questions about this topic are still very challenging, it is believed that there is no *efficient* method for the problem SNMF. Until one is found, we will try to solve the problem by looking for *good* approximations rather than doing exhaustive search for the global optimum.

The second problem can be considered as a relaxed version of the Symmetric Nonnegative Matrix Factorization. As we will see, the additional degrees of freedom in the Semi-Symmetric Nonnegative Matrix Factorization guarantees at least an exact approximation with finite inner rank of a arbitrary square nonnegative matrix, which is in general impossible with the Symmetric Nonnegative Matrix Factorization.

The chapter begins with a formal definition of the problem. Com-

pletely positive matrices are also discussed in the second part. Then a method to find *good* approximations is proposed. We mention some applications of this special factorization including Graph Clustering and Approximation of Correlation Matrix.

7.1 Symmetric approximations

We will define two NMF problems in which some symmetries are imposed on the factors. But first, we will discuss which structures of the approximation are considered.

7.1.1 Symmetry of the approximation

As pointed out earlier, only the product of a factored approximation such as those in the NMF problem can be considered as an approximation, since for a matrix B, there might be two different factorizations $B = U_1V_1^T$ and $B = U_2V_2^T$ where (U_1, V_1) and (U_2, V_2) are different. The first natural question to ask is: *given a symmetric nonnegative matrix, when will the NMF approximation also be symmetric?* This question is partially answered in [25] through the two following theorems:

Theorem 7.1. *Let $A = QDQ^T$ be a non-negative symmetric matrix of rank $t, t \leq n$, where $Q \in R^{n\times n}$ is orthogonal and $D = diag(d_1, ..., d_t, 0..0)$, with $d_i \neq -d_j$, for $i, j \in \{1, .., t\}, i \neq j$. Let (U, V) be a solution to the NNMF problem (Euclidean Distance version) with $\nabla_U \|A - UV^T\|_F^2 = 0$ and $\nabla_V \|A - UV^T\|_F^2 = 0$. Then the product UV is (also) symmetric.*

Theorem 7.2. *Let $A = QDQ^T$ be a non-negative symmetric matrix of rank $t, t \leq n$, where $Q \in R^{n\times n}$ is orthogonal and $D = diag(d_1, ..., d_t, 0..0)$, with $d_i \neq d_j$, for $i, j \in \{1, .., t\}, i \neq j$. Let (U, V) be a solution to the NNMF problem (Euclidean Distance version) with $\nabla_U \|A - UV^T\|_F^2 = 0$ and $\nabla_V \|A - UV^T\|_F^2 = 0$. Assume $rank(UV) = k \leq t$. Then the product UV is symmetric if and only if it is of the form $UV^T = QXQ^T$ where $X = diag(x_1, ..., x_t, 0, ..., 0)$ with $x_i = 0$ for all but k values of $i_1, ..., i_k$ with $x_{i_j} = d_{i_j}, j = 1, .., k$.*

These results are derived for NMF problem but only applied to the cases where $\nabla_U \|A - UV^T\|_F^2 = 0$ and $\nabla_V \|A - UV^T\|_F^2 = 0$. This, in fact,

targets only the unconstrained low-rank approximation problem and therefore can be easily derived from Theorem 1.13 for both symmetric and non-symmetric matrices.

Notice that we can always construct a symmetric approximation of the same complexity (probably with higher rank) from a known asymmetric approximation UV^T. Indeed, taking

$$\hat{A} = \frac{UV^T + VU^T}{2}$$

yields a symmetric approximation of A. Furthermore, it is a better approximation since

$$\left\| A - \frac{UV^T + VU^T}{2} \right\| \leq \frac{\|A - UV^T\| + \|A - UV^T\|}{2} = \|A - UV^T\|$$

from the convexity of $\|.\|$ and the symmetry of A.

Because of that, we are interested in the problem where UV^T is not only symmetric but also $U = V$. In other words, we search for the UU^T approximations where U is elementwise nonnegative. As we can see in the following sections, this problem is more difficult to solve but has more practical applications. A slightly different problem is to approximate a square nonnegative matrix by the product USU^T where U and S are elementwise nonnegative [120]. With its special structure, the latter can be also applied to square non-symmetric matrices.

7.1.2 The problems

In this section, the two problems of interest are Symmetric Nonnegative Matrix Factorization and Semi-symmetric Nonnegative Matrix Factorization. The formal definition as well as some properties of these two problems will be presented in preparation for the section of methods.

Symmetric Nonnegative Matrix Factorization

The following problem is referred to as the *Symmetric Nonnegative Matrix Factorization*:

Problem 7.3 (SNMF)**.**

$$\min_{U \in \mathbb{R}_+^{n \times r}} \frac{1}{2} \|A - UU^T\|_F^2$$

where A is a $n \times n$ symmetric matrix and r is an integer.

As an usual optimization problem, we calculate the gradient of the objective function on the factor U:

$$\nabla_U \frac{1}{2} \|A - UU^T\|_F^2 = UU^TU - AU.$$

The KKT conditions are:

$$\begin{aligned} U &\geq 0 && (7.1)\\ UU^TU - AU &\geq 0 && (7.2)\\ U \circ (UU^TU - AU) &= 0 && (7.3) \end{aligned}$$

It is known that the optimal nonnegative rank-one approximation is given by the Perron vectors. This remains true for the rank-one symmetric nonnegative factorization. But for the higher rank, the factorization is no longer easy. This is, indeed, related to the well-known class of completely positive matrices.

Every nonnegative matrix B that can be represented as UU^T where U is nonnegative is a completely positive (*cp*) matrix [10]. The above problem is equivalent to find the nearest completely positive matrix with rank r to the input matrix A. One important notion about the *cp* matrices is the *completely positive rank* (*cp-rank*) that is defined as the minimum number of column of U such that $B = UU^T$. We use $rank^+_{UU^T}(A)$ to denote the *cp-rank* of the matrix A. If A is not a *cp* matrix, we conventionally denote $rank^+_{UU^T}(A) = +\infty$.

The well-known problem of determining *cp-rank* of a symmetric matrix A has been a subject many researches. Knowing a lower bound of this number, which is the ordinary rank of the matrix $k = rank(A)$, the true general upper bound is still unknown. The best proved bound is $\frac{k(k+1)}{2} - 1$ [7]. Two fundamental problems are:

- Deciding whether a matrix is a *cp* matrix.

- Determining the *cp-rank* of a *cp* matrix.

They were open until recently, the application of Renegar algorithm [9] yields a finite method to compute the *cp-rank* of a matrix if it exists. But the complexity bound of this method is still non-polynomial. See [9] or Section 2.3 to derive this bound.

One of the necessary conditions for A symmetric to be completely positive is that it must be positive semidefinite. When A is not, surely we can not have an exact Symmetric Nonnegative Matrix Factorization for finite rank. Knowing that the SNMF can not do better than the nearest semidefinite positive matrix, the following results give a lower bound of the error function of SNMF approximation:

Theorem 7.4. *Let A is be symmetric matrix. Then*

$$\min_{\substack{rank(B)=r \\ B\succeq 0}} \|A-B\|_F^2 \geq \sum_{\lambda_i(A)<0} (\lambda_i(A))^2.$$

This is the simplified version of a theorem found in [57]. This result implies that when approximating a symmetric matrix by a semidefinite positive matrix, one should only take into account the semidefinite positive part $A_{\succeq 0}$ of the original matrix A. And applying the Eckart-Young theorem (Theorem 1.12) to the matrix $A_{\succeq 0}$ allows us to construct the optimal approximation for the unconstrained problem.

A very simple example is the following matrix with its eigenvalue decomposition:

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \sqrt{2}u_1u_1^T - \sqrt{2}u_2u_2^T.$$

where $u_1 = (\frac{\sqrt{2}}{2}\ \frac{1}{2}\ \frac{1}{2})$ and $u_2 = (\frac{-\sqrt{2}}{2}\ \frac{1}{2}\ \frac{1}{2})$. Applying Theorem 7.4, the best semidefinite positive approximation is the rank-one matrix $\hat{A} = A_{\succeq 0} = \sqrt{2}u_1u_1^T$. Observe that this approximation is also the optimal SNMF approximation and therefore it is impossible to create a better SNMF approximation even with higher ranks. This impossibility can be overcome if one uses a more flexible 'symmetric' approximation introduced by the second problem.

Semi-Symmetric Nonnegative Matrix Factorization

The following were first mentioned in [37, 120]. We refer to it as the Semi-Symmetric Nonnegative Matrix Factorization (SSNMF) problem:

Problem 7.5 (SSNMF)**.**

$$\min_{U \in \mathbb{R}_+^{n \times r}} \min_{S \in \mathbb{R}^{r \times r}} \|A - USU^T\|_F^2$$

where A is a $n \times n$ nonnegative matrix and r is an integer.

For the symmetric matrix A, the matrix S is a way to absorb all the nonnegative eigenvalues. This is why the lower bound of the error for this type of approximation is exactly zero. The symmetry of A also allows us to assume further the symmetry of S. Since if S is not symmetric, $S^* = \frac{S+S^T}{2}$ yields a better approximation

$$\begin{aligned} \left\| A - U\left(\frac{S+S^T}{2}\right)U^T \right\| &\leq \frac{\|A - USU^T\| + \|A - US^TU^T\|}{2} \\ &= \|A - USU^T\|. \end{aligned}$$

Consider again the above example. We have already seen that it is impossible to create an exact UU^T NMF approximation, but there exists an exact USU^T approximation with $r = 2$:

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

In fact, every nonnegative matrix admits at least one such exact approximation ($A = USU^T$) trivially given by $U = I$ and $S = A$. And it is not limited to only symmetric nonnegative matrix but can be also applied to an arbitrary square nonnegative matrix. We call the minimum value of r that yields an exact approximation the *Semi-Symmetric Nonnegative Rank* denoted by $rank^+_{USU^T}(A)$. Because it is bounded from above by n, the size of the original matrix, there is a finite algorithm to compute this number using Renegar algorithm (crf. Section 2.3).

The following result can be established to relate the different ranks of nonnegative factorizations presented in this thesis.

Lemma 7.6. *For every $n \times n$ square nonnegative matrix A, we have*

$$rank(A) \leq rank^+_{UV^T}(A) \leq rank^+_{USU^T}(A) \leq n$$

and

$$rank^+_{USU^T}(A) \leq rank^+_{UU^T}(A).$$

Proof. Trivial. Since the rank of every exact factorization of a $rank - k$ matrix can not have a smaller rank than k and we can create a UV^T factorization from USU^T and USU^T from UU^T. The upper bound $rank^+_{USU^T}(A) \leq n$ is given by the trivial factorization $U = I$ and $S = A$. □

With the same argument used in the above Lemma, we can establish, in the following lemma, a similar relation of minimum error between different approximations of the same reduced rank r.

Lemma 7.7. *For every $n \times n$ square nonnegative matrix A and $r = 1, 2, \ldots$*

$$\begin{aligned} \min_{U,V \in \mathbb{R}^{n \times r}} \|A - UV^T\|_F^2 &\leq \min_{U,V \in \mathbb{R}_+^{n \times r}} \|A - UV^T\|_F^2 \\ &\leq \min_{U \in \mathbb{R}_+^{n \times r}} \min_{S \in \mathbb{R}^{r \times r}} \|A - USU^T\|_F^2 \\ &\leq \min_{U \in \mathbb{R}_+^{n \times r}} \|A - UU^T\|_F^2. \end{aligned}$$

7.2 Methods

In this section, we provide some directions that one can follow to solve the two above problems. Since it is not realistic to solve them optimally, we investigate some algorithms that can produce *good* approximations for the above problems.

7.2.1 Direct methods

Since the error function is differential, the gradient informations are always available. Therefore, all gradient methods presented in Chapter 3 can be reused for these problems.

7.2.2 Penalty methods

Our strategy is to *guide* the NMF algorithm to converge to a fixed point of SNMF *if possible*. We start the NNMF algorithm with an asymmetric starting point (i.e. $U_0V_0^T$), and then force the two factors (i.e. U_k and V_k) to be equal by minimizing their normed difference. One way to do is to use the following modified cost function for a symmetric matrix A:

$$F_s(A, UV^T) = \frac{1}{2}\|A - UV^T\|_F^2 + \frac{\alpha}{2}\|U - V\|_F^2,\ \alpha > 0 \tag{7.4}$$

where the earlier part ($\frac{1}{2}\|A - UV^T\|_F^2$) is called the approximation error and the latter part ($\frac{1}{2}\|U - V\|_F^2$) is called the asymmetry penalty.

We will derive two methods based on the Multiplicative Rules and the Rank-one Residue Iteration scheme for solving the following problem:

Problem 7.8.

$$\min_{U\in\mathbb{R}_+^{n\times r}\ V\in\mathbb{R}_+^{n\times r}} F_s(A, UV^T).$$

Regularized multiplicative rules

Incorporating additional linear or quadratic costs with the Euclidean Distance does not change significantly the updating rules. Using the same reasoning for constructing the multiplicative updating rules, one can obtain Algorithm 12 for the SNMF problem.

Different from the original algorithm, the regularized version needs the following balancing step after each update of U (step 4) or V (step 5). These consist in balancing the norm of each column of U and that of the corresponding column of V

$$D_{ii} = \frac{\sqrt{\|V_{:,i}\|}}{\sqrt{\|U_{:,i}\|}} \qquad U = UD \qquad V = VD^{-1}. \tag{7.5}$$

Regularized rank-one residue iteration

We modify the algorithm presented in Chapter 4 by taking into account the regularized cost function. The elementary Rank-one problem be-

Algorithm 12 Multiplicative Rules

$$(U^*, V^*) = \underset{U \geq 0 \; V \geq 0}{\operatorname{argmin}} \frac{1}{2}\|A - UV^T\|_F^2 + \frac{\alpha}{2}\|U - V\|_F^2$$

1: Initialize $U^0 = V^0$ and $k = 0$
2: **repeat**
3: $\quad U^{k+1} = U^k \circ \frac{[AV^k + \alpha(U^k - V^k)]_+}{[U^k(V^k)^T(V^k)]}$
4: $\quad$ Balancing U^{k+1} and V^k
5: $\quad V^{k+1} = V^k \circ \frac{[A^TU^{k+1} + \alpha(V^k - U^{k+1})]_+}{[V^k(U^{k+1})^T(U^{k+1})]}$
6: $\quad$ Balancing U^{k+1} and V^{k+1}
7: $\quad k = k + 1$
8: **until** Stopping condition

comes:

Problem 7.9.

$$\min_{v \geq 0} F_s(v)$$

where $F_s(v) = \frac{1}{2}\left(\|R_t - u_t v^T\|_F^2 + \alpha\|u_t - v\|_F^2\right)$ with $R_t = A - \sum_{i \neq t} u_t v_t^T$.

From this formulation, one now can derive the following similar lemma to the one in Chapter 4.

Lemma 7.10. *If* $[R_t^T u_t + \alpha u_t]_+ \neq 0$, *then*

$$v_* := \frac{[R_t^T u_t + \alpha u_t]_+}{u_t^T u_t + \alpha}$$

is the unique global minimizer of Problem 7.8 and the value of the modified cost function equals

$$F_s(v^*) = \|R_t\|_F^2 + \alpha u_t^T u_t - \frac{\left\|[R_t^T u_t + \alpha u_t]_+\right\|^2}{u_t^T u_t + \alpha}.$$

Proof. Rearranging the cost function gives:

$$\begin{aligned} F_s(v) &= \tfrac{1}{2}\left(\|R_t - u_t v^T\|_F^2 + \alpha\|u_t - v\|_F^2\right) \\ &= \tfrac{1}{2}\left(trace[(R_t - u_t v^T)^T(R_t - u_t v^T)] + \alpha(u_t - v)^T(u_t - v)\right) \\ &= \tfrac{1}{2}\left(\|R_t\|_F^2 - 2v^T(R_t^T u_t + \alpha u_t) + v^T v(u_t^T u_t + \alpha) + \alpha u_t^T u_t\right) \\ &= \tfrac{1}{2}\left(\|R_t\|_F^2 + \alpha u_t^T u_t - 2v^T x + Cv^T v\right) \end{aligned}$$

where $x := R_t^T u_t + \alpha u_t$ and $C := u_t^T u_t + \alpha$.

Let us permute the elements of the vectors x and v such that

$$Px = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{and} \quad Pv = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

with $x_1 \geq 0$, $x_2 < 0$ and P is the permutation matrix. Then

$$F_s(v) = \|R_t\|^2 + \alpha u_t^T u_t - 2v_1^T x_1 - 2v_2^T x_2 + C(v_1^T v_1 + v_2^T v_2).$$

Since $x_2 < 0$, $v_2 \geq 0$ and $C \geq 0$, it is obvious that the modified cost function can only be minimal if $v_2 = 0$. Our assumption implies that x_+ is nonempty and $x_1 \geq 0$. One can then find the optimal v_1 by minimizing the remaining quadratic function

$$\|R_t\|_F^2 + \alpha u_t^T u_t - 2v_1^T x_1 + C v_1^T v_1$$

which yields the solution $v_1 = \frac{x_1}{C}$. Putting the two components together yields the result

$$v_* = \frac{x_+}{C} \quad \text{and} \quad F_s(v^*) = \|R_t\|_F^2 + \alpha u_t^T u_t - \frac{\|x_+\|^2}{C}.$$

□

In the case where $[R_t^T u_t + \alpha u_t]_+ = 0$, we have a *unique* optimal solution: $v^* = 0$ that is not covered by Lemma 7.10. This solution is unique even when $u_t = 0$. But this will result in a zero component in the approximation.

An update for u_t can be similarly formulated. Using these to iterate through all the u_i's and v_i's yields Algorithm 13 for the modified cost function.

In Algorithm 13, balancing steps (7) and (11) are added after each update. This does not increase the approximation error $\|A - UV^T\|_F^2$ but decreases the asymmetry penalty $\|U - V\|_F^2$, hence yields a more balanced factorization that helps the algorithm to converge to the *symmetric* fixed point. This is also a way to restrict all variables inside some compact sets and it is hence easy to show that every limit point is a stationary point.

Algorithm 13 Rank-one Residue Iteration for SNMF

Initialize u_i's, $v_i = u_i$
repeat
  for $t = 1$ to r **do**
    $R_t = A - \sum_{i \neq t} u_i v_i^T$
    $v_t \leftarrow \frac{[R_t^T u_t + \alpha u_t]_+}{u_t^T u_i + \alpha}$
    if $\|u_t\|_2 > 0$ and $\|v_t\|_2 > 0$ **then**
      $d = \sqrt{\|v_t\| / \|u_t\|}$, $u_t = d u_t$ and $v_t = (1/d) v_t$
    end if
    $u_t \leftarrow \frac{[R_t v_t + \alpha v_t]_+}{v_t^T v_t + \alpha}$
    if $\|u_t\|_2 > 0$ and $\|v_t\|_2 > 0$ **then**
      $d = \sqrt{\|v_t\| / \|u_t\|}$, $u_t = d u_t$ and $v_t = (1/d) v_t$
    end if
  end for
until Stopping condition

Theorem 7.11. *Every limit point of Algorithm 13 is a stationary point of the problem 7.8.*

Proof. The additional asymmetry penalty term make the optimal solution of the subproblems *always unique*. Note also that each of the variables u_i and v_i ($i = 1, \ldots, r$) can be bounded to stay in a closed convex set. Applying Theorem 1.11 proves the desired result. □

In Algorithm 13, when a zero component appear, one could replace $u_t v_t^T$ by any rank-one approximation that reduces the modified norm of the error matrix. Note that when adding the asymmetry penalty term, the replacing rank-one matrix is not as easy as for the general NMF case in Chapter 4.

For symmetric substitution, one very simple possibility is to choose the largest element on the diagonal of R_t, for example $[R_t]_{kk}$. If it is positive, then we can use $[R_t]_{kk} e_k e_k^T$ (i.e. $u_t = v_t = \sqrt{[R_t]_{kk}} e_k$) as an alternative.

Asymmetric patch to the rank-deficient can also be used unless this makes the cost function increase. For example, any element $[R_t]_{kl}$ such that $[R_t]_{kl} < 2\alpha$ gives an approximation $u_t = \sqrt{[R_t]_{kl}} e_k$ and $v_t =$

$\sqrt{[R_t]_{kl}}e_l$. Random restart of u_t and/or v_t may also be used when it can make an improvement over the trivial solution $v^* = 0$.

These patches should be applied only in a finite number of times to guarantee the convergence of the algorithm. In practice, only a few are needed. And in many cases, it is not easy to find such a patch. Therefore, we have to accept zero components in the final approximation.

Semi-symmetric nonnegative matrix factorization

We modify the algorithm presented in Chapter 4 by taking into account the asymmetry penalty. The problem to solve is the following:

Problem 7.12.

$$\min_{\substack{U\in\mathbb{R}_+^{n\times r}\ V\in\mathbb{R}_+^{n\times r}\\ S\in\mathbb{R}_+^{r\times r}}} F_{SS}(A, USV^T).$$

where

$$F_{ss}(A, USV^T) = \frac{1}{2}\|A - USV^T\|_F^2 + \frac{\alpha}{2}\|U - V\|_F^2.$$

Let $W = US$ and w_t is the t^{th} column of W. To derive an update for the column v_t of V, we look at the following partial problem:

Problem 7.13.

$$\min_{v\geq 0} F_{ss}(v)$$

where $F_{ss}(v) = \frac{1}{2}\left(\|R_t - w_t v^T\|_F^2 + \alpha\|u_t - v\|_F^2\right)$ with $R_t = A - \sum_{i\neq t} w_i v_i^T$.

The following lemma gives the optimal update of v_t:

Lemma 7.14. *If* $[R_t^T w_i + \alpha u_t]_+ \neq 0$, *then*

$$v_* := \frac{[R_t^T w_t + \alpha u_t]_+}{w_t^T w_t + \alpha}$$

is the unique *global minimizer of Problem 7.8 and the value of the modified cost function equals*

$$F_{ss}(v^*) = \|R_t\|_F^2 + \alpha u_t^T u_t - \frac{\left\|[R_t^T w_t + \alpha u_t]_+\right\|^2}{w_t^T w_t + \alpha}.$$

Remark: when $[R_t^T w_t + \alpha u_t]_+ = 0$, which is not covered by Lemma 7.14, the *unique* optimal update is $v^* = 0$, even when $u_t = 0$.

The proof of this lemma is similar to Lemma 7.10 and is skipped. A similar development yields the update for u_t:

$$u_* := \frac{[Q_t y_t + \alpha v_t]_+}{y_t^T y_t + \alpha}$$

where y_t is the t^{th} column of $Y = VS^T$ and $Q_t = A - \sum_{i \neq t} u_i y_i^T$.

Now, we need to update S given U and V. We choose to update each element of S separately. We can rewrite the approximation error as:

$$\begin{aligned} \|A - USV^T\|_F^2 &= \|A - \sum_{ij} S_{ij} u_i v_j^T\|_F^2 \\ &= \|(A - \sum_{(i,j)\neq(k,l)} S_{ij} u_i v_j^T) - S_{kl} u_k v_l^T\|_F^2 \\ &= \|P_{kl} - S_{kl} u_k v_l^T\|_F^2 \end{aligned}$$

where P_{kl} is the residue matrix $A - \sum_{(i,j)\neq(k,l)} S_{ij} u_i v_j^T$. This is a very simple scaling problem. But the optimal solution is only unique if $\|u_k\|_2 > 0$ and $\|v_l\| > 0$:

$$S_{kl}^* = max\left(\frac{u_k^T P_{kl} v_l}{u_k^T u_k v_l^T v_l}, 0\right).$$

Since the uniqueness is needed for the convergence, therefore, one can again use the damping method described in Section 4.3 for the variable S. This is done by introducing a dummy variable T and a small positive constant ψ and adding the terms $\frac{\psi}{2}\|S - T\|_F^2$ to the cost function. Using the same development in Section 4.3, we have the following update for S_{kl}:

$$S_{kl}^* = max\left(\frac{u_k^T P_{kl} v_l \; + \; \psi S_{kl}}{u_k^T u_k v_l^T v_l \; + \; \psi}, 0\right).$$

Putting all together yields Algorithm 14.

The normalization step (18) in Algorithm 14 consists of finding two positive diagonal matrix D_U and D_V and updating:

$$U = UD_U,\; V = VD_V \text{ and } S = D_U^{-1} S D_V^{-1},$$

Algorithm 14 Rank-one Residue Iteration for SSNMF

Initialize S, u_i's, $v_i = u_i$
repeat
  for $t = 1$ to r **do**
    $W = US$
    $R_t = A - \sum_{i \neq t} w_i v_i^T$
    $v_t \leftarrow \frac{[R_t^T w_t + \alpha u_t]_+}{w_t^T w_t + \alpha}$
    $Y = VS^T$
    $Q_t = A - \sum_{i \neq t} u_i y_i^T$
    $u_t \leftarrow \frac{[Q_t y_t + \alpha v_t]_+}{y_t^T y_t + \alpha}$
  end for

  for $k = 1$ to r **do**
    for $l = 1$ to r **do**
      $P_{kl} = A - \sum_{(i,j) \neq (k,l)} S_{ij} u_i v_j^T$
      $S_{kl} \leftarrow max\left(\frac{u_k^T P_{kl} v_l + \psi S_{kl}}{u_k^T u_k v_l^T v_l + \psi}, 0\right)$
    end for
  end for
  Normalization
until Stopping condition

such that every non-zero vector u_i and v_i have unit norm. This makes the variables stay in a bounded set and hence is easy to show that every limit point of Algorithm 14 is a stationary point. Indeed, each update is the unique optimal solution of a simple convex problem (cfr. Theorem 4.5).

Theorem 7.15. *Every limit point of Algorithm 14 is a stationary point of the problem 7.12.*

Proof. Reusing the argument for the dummy variable T in Theorem 4.7 from Section 4.3 together with the uniqueness of the optimal solution of u_i and v_i yield the desired result. □

Again, if one wants to avoid zero components in the approximation, patches can be applied. But in general, it is more difficult to do that here

than in the previous section. But one can always take a random vector for the corresponding column of S. It must be done with care because this may increase the cost function. If we can not find an acceptable fix, then we should accept the solution $v^* = 0$ and wait for the next sweep. If found, the patches should be applied in only a finite number of times to guarantee the convergence of the algorithm.

Regularizing constant

The parameter α appears in the two previous formulations as the trade-off between the approximation error and the asymmetry penalty of the fixed point. The right choices of α can lead the search direction toward the symmetric fixed points. Overestimating α could lead to a symmetric factorization without taking into account the approximation error. And $\alpha = 0$ means that the penalty is absent, hence becomes the NMF Algorithm 7.

An appropriate value of α is believed to depend on the best possible approximation error, but this error is not known in general. Experiments show that the range of possible values of α is quite large, so that, one can restart the algorithm with a better value of α determined from the previous results of the algorithm.

It is also useful to note that, after all, the results might not be in the desired form in which $U = V$. But by having forced the symmetry, one can expect that the errors $\|A - UU^T\|_F^2$ (or $\|A - USU^T\|_F^2$) and $\|A - VV^T\|_F^2$ (or $\|A - VSV^T\|_F^2$) are not too far from the real approximation error $\|A - UV^T\|_F^2$ (or $\|A - UV^T\|_F^2$). One could even used $U^* = \frac{U+V}{2}$ as a solution.

In practice, the symmetry regularization does not only force the factors to be equal, but it can also lead to a better approximation. There are examples where the symmetric approximation is better than the asymmetric one. But applying the general algorithms for NMF problem may have difficulty to converge to a symmetric approximation. An example is when we applied NMF method to the well-known iris flower data [43]. It consists of a 150×4 matrix X whose rows contain 4 measures of a flower: the sepal length, the sepal width, the petal length and the petal width. There are 3 species in the dataset ($1 - 50$: setosa, $51 - 100$: versicolor and $101 - 150$: virginica). We construct the similarity matrix

between the flowers $S = XX^T$. We apply Algorithm 13 to the similarity matrix S with three different values of $\alpha = 0,\ 1$ and 50 and with the reduced rank $r = 3$. Note again that, $\alpha = 0$ removes the penalty from the cost function. We present here the plot of 4 functions for each iteration of the Rank-one Residue Iteration algorithm:

- the full cost function: $f_1^{\alpha}(U_k, V_k) = \frac{1}{2}\|A - U_k V_k^T\|_F^2 + \frac{\alpha}{2}\|U_k - V_k\|_F^2$,
- the asymmetry error: $f_2(U_k, V_k) = \frac{1}{2}\|U_k - V_k\|_F^2$,
- the approximation error of UV^T: $f_3(U_k, V_k) = \|A - UU^T\|_F^2$
- the approximation error of UU^T: $f_4(U_k) = \|A - UU^T\|_F^2$
- and the approximation error of VV^T: $f_5(V_k) = \|A - VV^T\|_F^2$.

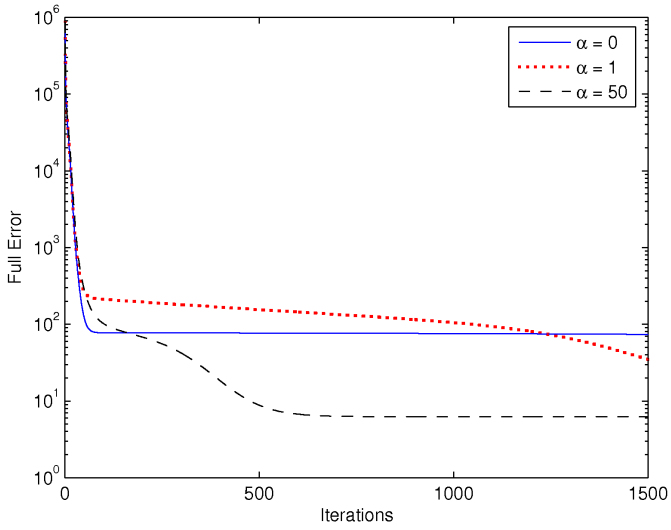


Figure 7.1: Full cost function for different values of α

Figure 7.1 shows that the asymmetric algorithm ($\alpha = 0$) can not converge to the lower error approximation. It was trapped in an asymmetric local minimum, which yields a large approximation error. The asymmetry error function (f_2) shows that the approximation is totally not symmetric.

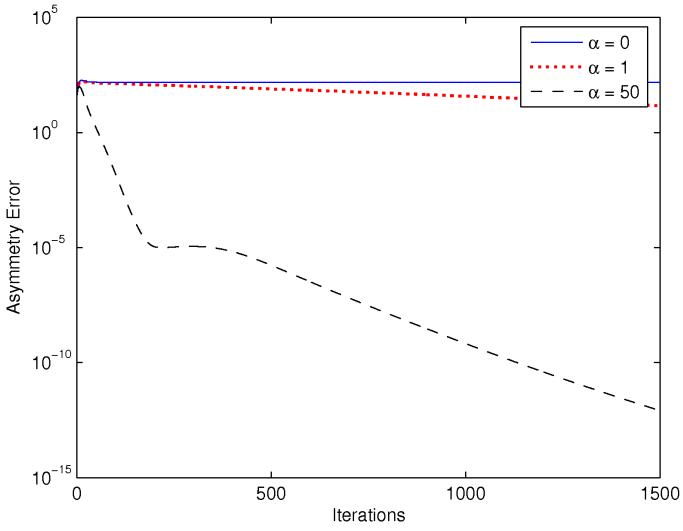


Figure 7.2: Asymmetry error for different values of α

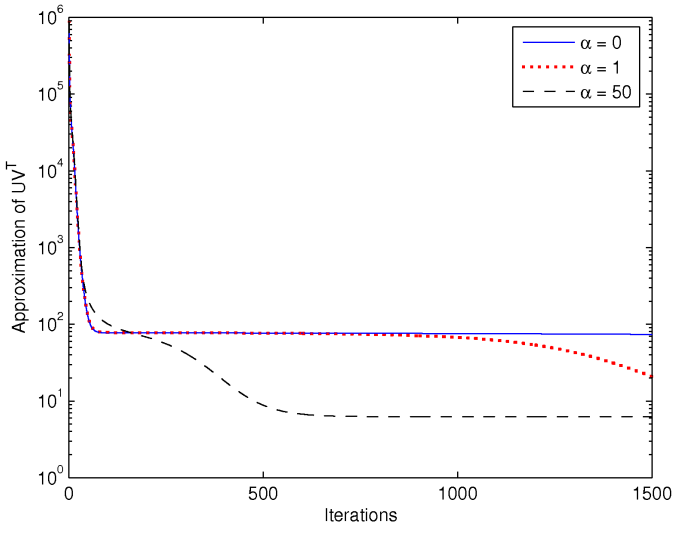


Figure 7.3: Approximation error of UV^T for different values of α

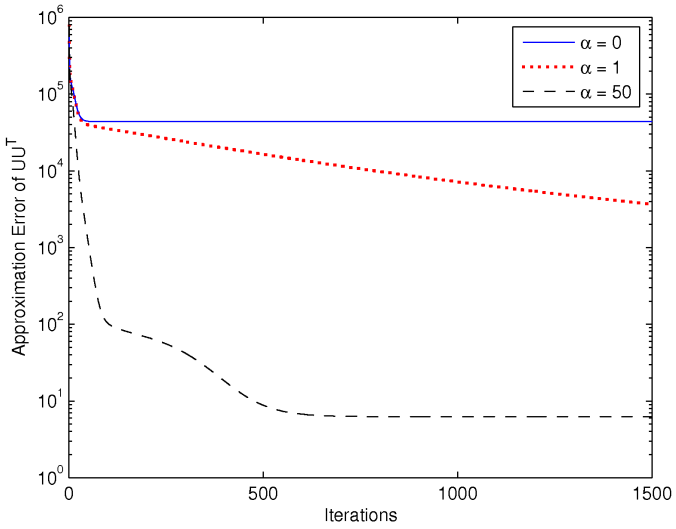


Figure 7.4: Approximation error of UU^T for different values of α

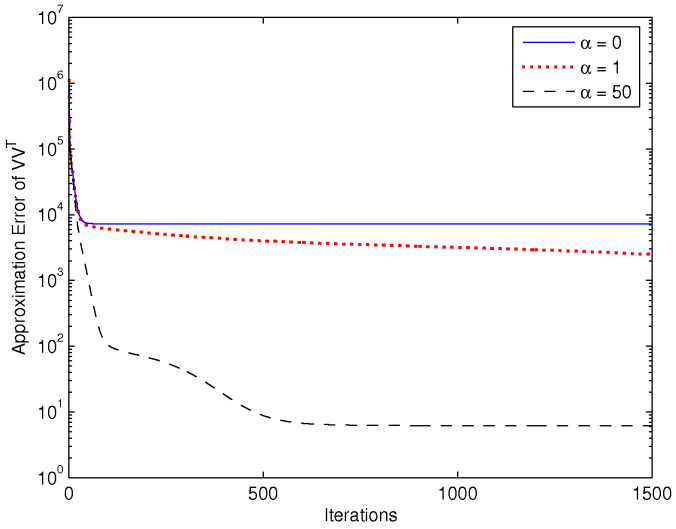


Figure 7.5: Approximation error of VV^T for different values of α

The approximation errors produced by UU^T and VV^T stay high. For $\alpha = 1$, there is an improvement in all measures. But the convergence is still very slow. When we increase α to 50, Algorithm 13 produces a symmetric approximation with a lower error than the asymmetric one. Furthermore, it converges much faster. In fact, in this example, it is very likely that the best approximation is symmetric (i.e. UU^T) because the input matrix S is *cp* by construction.

In general, when approximating a matrix, if we know that the best approximation is symmetric, we can add the described regularization to guide the algorithm eventually to the global minimizer. In this case, the symmetry plays a role of prior knowledge of the problem that one can provide to the algorithm.

For the Semi-Symmetric Nonnegative Matrix Factorization, we observed the same phenomena.

Sparse approximation

The factor U in the two symmetric problems plays a central role in their clustering applications where the sparsity of U is required. We can therefore modify the above algorithms to create sparser approximation. A simple technique is to add extra regularization term $\beta(\sum_i \|u_i\|_1 + \|v_i\|_1)$ to the cost function (7.4), as suggested in [21]. This will slightly modify the update for U and V but a sparser approximation can be obtained.

The modified updates are:

- Symmetric Nonnegative Matrix Factorization:

$$v_t \leftarrow \frac{[R_t^T u_t + \alpha u_t - \beta \mathbf{1}_{n\times 1}]_+}{u_t^T u_t + \alpha} \text{ and } u_t \leftarrow \frac{[R_t v_t + \alpha v_t - \beta \mathbf{1}_{n\times 1}]_+}{v_t^T v_t + \alpha}.$$

- Semi-Symmetric Nonnegative Matrix Factorization:

$$v_t \leftarrow \frac{[R_t^T w_t + \alpha u_t - \beta \mathbf{1}_{n\times 1}]_+}{w_t^T w_t + \alpha} \text{ and } u_t \leftarrow \frac{[Q_t y_t + \alpha v_t - \beta \mathbf{1}_{n\times 1}]_+}{y_t^T y_t + \alpha}.$$

7.3 Applications: graph clustering

Clustering is one of the central problems in data analysis and exploration. Clustering amounts to finding out the structure of data by dividing elements into groups while maximizing the similarity or the connectivity between elements in the same group and at the same time, minimizing these quantities between elements that are not in the same group.

The graph clustering is a particular case where vertices of a graph are partitioned into different group so that inside each group, the connectivity or similarity [18] is high, while the connectivity or the similarity between different groups is low. We will see how the problem of a graph clustering based on the connectivity of graph can be reformulated as a problem of Symmetric Nonnegative Matrix Factorization.

In general, one can define the graph clustering problem as follows: *Given an integer $k > 0$ and a graph $G = (V, E)$, where V ($|V| = n$) is the set of vertices and E is the set of edges, find a partition of V: $C = \{C_1, ..., C_k\}$ that maximizes a specified performance function $f(G, C)$.*

In this definition, each of the sets C_i is a cluster and C is called a clustering.

For the connectivity case, the input graph is usually undirected or made to be undirected, implying that if vertex i is connected to vertex j (i.e. $(i, j) \in E$), then vertex j is also connected to vertex i (i.e. $(j, i) \in E$). We further assume that each vertex of the graph is connected to itself (i.e. $(i, i) \in E$).

The choice of f depends on each application and it will determine the nature of the clusters. For the connectivity case, we use the following *connectivity performance* function [22]:

$$cperf(G, C) = \frac{\#\{(i,j) \in E |\ (i,j) \in C\} \ + \ \#\{(i,j) \notin E |\ (i,j) \notin C\}}{n^2}, \tag{7.6}$$

where $(i, j) \in C$ means that i and j belong to the same cluster in C or in other words, the edge (i, j) is in the clustering C. The first value in the numerator $\#\{(i, j) \in E | (i, j) \in C\}$ is the number of edges which are in common to E and the clustering C. And the second value is the number of edges which are not common to E and C. We see that the numerator

is always less or equal to n^2, so the value of $cperf(G,C)$ varies between 0 and 1 and a higher value of $cperf(G,C)$ implies a better clustering C. In fact, we can rewrite $cperf(G,C)$ as below:

$$cperf(G,C) = 1 - \frac{\#\{(i,j) \in E |\ (i,j) \notin C\} + \#\{(i,j) \notin E |\ (i,j) \in C\}}{n^2}, \tag{7.7}$$

where the first value in the numerator $\#\{(i,j) \in E \mid (i,j) \notin C\}$ is the number of edges which are in E but not in C. And the second value is the number of edges which are in C but not in E. We can call the sum of these two values in the numerator, the *clustering error*.

Let A be the adjacency matrix of G ($A_{ij} = 1$ if $(i,j) \in E$ and $A_{ij} = 0$ otherwise) and let $X_C \in \{0,1\}^{n \times k}$ be the membership matrix corresponding to the clustering C ($X_{ir} = 1$ if $i \in C_r$ and $X_{ir} = 0$ if $i \notin C_r$), then the function $cperf$ in (7.7) can be rewritten as:

$$\begin{aligned} cperf(G,C) &= 1 - \frac{\sum_{ij}\left(A_{ij} - \sum_{r=1}^{k} X_{ir}X_{jr}\right)^2}{n^2} \\ &= 1 - \frac{\|A - XX^T\|_F^2}{n^2}. \end{aligned} \tag{7.8}$$

As a result, the graph clustering problem that maximizes the function $cperf$ is equivalent to the problem:

$$(GC) \qquad \min_{X \in \{0,1\}^{n \times k},\ X\mathbf{1}_{k\times 1} = \mathbf{1}_{n\times 1}} \|A - XX^T\|_F^2.$$

Since this problem is is known to be *NP-hard*, we will relax its constraints to obtain a simpler problem:

$$(SNNMF) \qquad \min_{X \in \mathbb{R}_+^{n \times k}} \|A - XX^T\|_F^2$$

that can be solved for a local minimum using the described algorithm for the Symmetric Nonnegative Matrix Factorization.

For our experiment we take an example from the literature which is called a geometric graph [41, 117] (see figure 7.6 left). Such a graph is

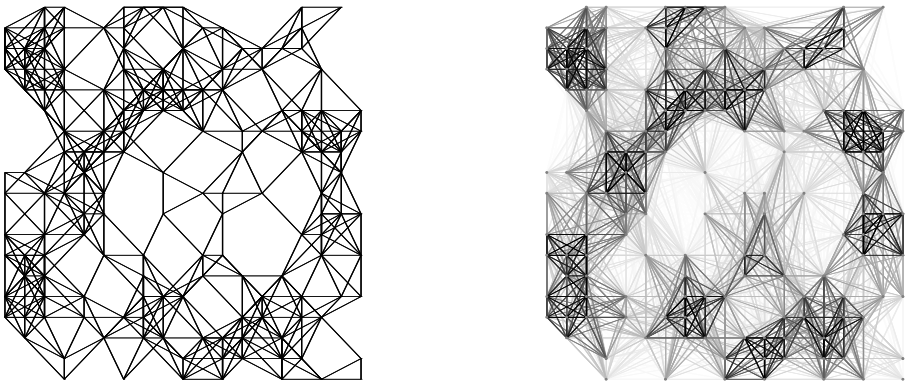


Figure 7.6: Geometric graph: original (left) and reconstructed (right)

constructed as follows: first put randomly n vertices on a $N \times N$ grid whose cells have unit size, then create an edge between every pair of vertices of geometric distance less than d. In the above example, $n = 150$, $N = 20$ and $d = \sqrt{8}$.

We run the algorithm 13 with $k = 14$ (the choice of k will be discussed later) and we obtain a fixed point which is symmetric (i.e. $U \approx V^T$ and $A \approx UU^T$). Although the original graph is unweighted, the reconstructed graph is weighted and is shown on the right of figure 7.6 with different gray levels going from white (weight = 0) to black (weight = 1). Despite this, the main features of the original graph are well maintained.

Figure 7.7 presents 14 component graphs, that are rank-one graphs corresponding to each column of U. One can see that for this particular fixed point, all the component graphs are quite local and independent, reflecting a cluster structure for the given graph. Moreover, if we relabel the vertices such that all the vertices in the same cluster have consecutive labels and make a plot of the nonzero elements of the permuted adjacency matrix, we obtain a *block* structure reflecting the clusters as in the figure 7.8.

In this example, we have chosen the number of clusters $k = 14$ in order to compare the results with those given by *Markov Clustering algorithm* (MCL) [117]. The idea of MCL is different from the one presented here: instead of fixing the number of clusters in advance, it tries to find a k which is referred to as the number of *natural clusters* in the given

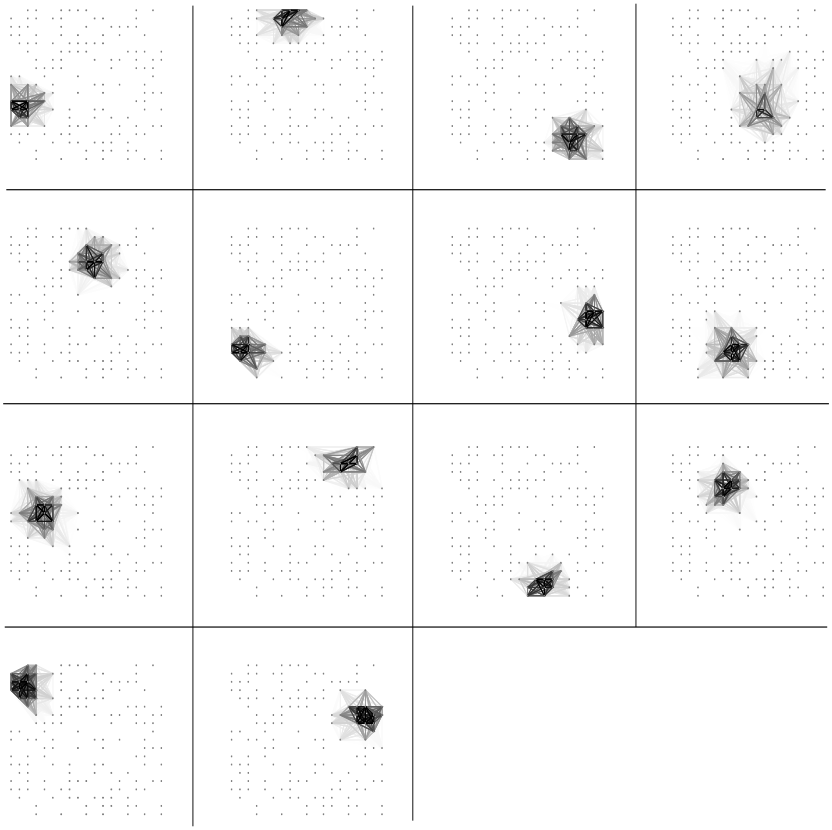


Figure 7.7: 14 *component graph* $U_i U_i^T$

graph, and, 14 was the number of clusters given by MCL for the above geometric graph.

Using the same number of cluster $k = 14$ for NNGC, we do obtain a slightly higher value of performance ($cperf(G)_{NNGC} = 0.9557$) compared to what is given by MCL ($cperf(G)_{MCL} = 0.95$). The following is a brief comparison between the two algorithms:

The pros and cons of *MCL*:

- **Pros**: the resulting matrix is very sparse; the number of cluster is automatically determined by the algorithm; quadratic convergence around the fixed point.

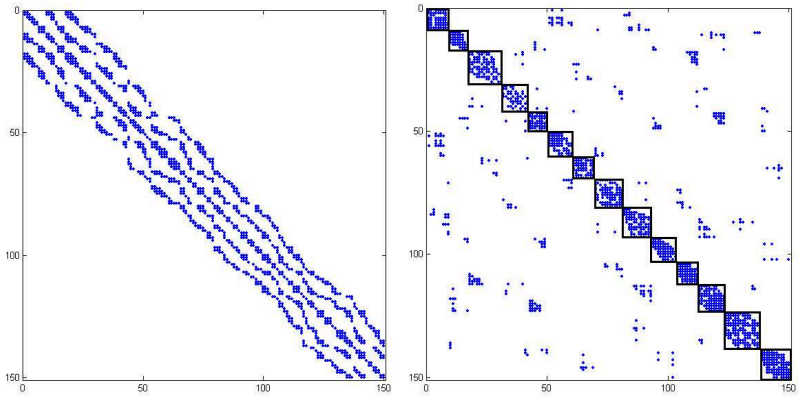


Figure 7.8: Adjacency matrix (left) and its permuted version (right)

- **Cons**: although the resulting matrix is very sparse, the iteration matrices can be dense; there is no convergence proof; there is no optimal criterion, results are based only on experimental results.

The pros and cons of *NNGC*:

- **Pros**: the iterative matrices have fixed rank, hence the computation load and memory requirement are known; the number of cluster can be chosen in advance; well-defined optimality criterion; proved convergence to local minimizer.
- **Cons**: the best number of clusters must be chosen from different possible values; linear convergence.

7.4 Application: correlation matrix approximation

The CreditRisk$^+$ model [112] is one of the industry standards for estimating the credit default risk for a portfolio of credit loans. The natural parameterization of this model requires the default probability to be apportioned using a number of (non-negative) factor loadings. However, in practice only default correlations are often available but not the factor loadings. The standard Credit Risk$^+$ model assumes that there is only one factor. Increasing the number of factors in the CreditRisk$^+$

model is always tied to increasing the rank of the approximation for the default correlation matrix. Ultimately, the use of as many factors as credit exposures would allow to reproduce the correlation matrix exactly. In principle, this would lead to the most accurate approximation. The following is a summary of the application of the Symmetric Nonnegative Matrix Factorization to the CreditRisk$^+$ model. For more details, the interested reader is referred to [119].

In this context, the "Credit Portfolio Loss" is modeled by the following random variable S, :

$$S = \sum_{i=1}^{n} N_i \, c_i, \tag{7.9}$$

where we assume that c_i's are known. The multivariate distribution of $N = \begin{bmatrix} N_1 & N_2 & \dots & N_n \end{bmatrix}^T$ is defined with the assumption that there exists a random vector $\Gamma = \begin{bmatrix} \Gamma_1 & \dots & \Gamma_K \end{bmatrix}^T$ representing the "state of the economy" such that the random variables N_i, conditioned by $\Gamma_k = \gamma_{k,}$ $(k = 1, 2, ..., K)$, are mutually independent and Poisson distributed with intensity:

$$\mathcal{R}_i = q_i \left(w_{0,i} + \sum_{k=1}^{K} w_{k,i} \gamma_k \right), \tag{7.10}$$

where q_i is the probability that the risk i leads to a failure.

The coefficient $w_{0,i}$ reflects the portion of idiosyncratic risk that can be attributed to the i-th risk whereas $w_{k,i}$ reflects its affiliation to the k-th common factor. It is important to note that in order to have positive intensities in (7.10), the coefficients $w_{k,i}$, $k = 0, 1, ..., K$ will be constrained to be non-negative.

The random variables Γ_i are assumed to be independent Gamma distributed and the covariance matrix associated with the random vector Γ is given by:

$$cov[\Gamma] = \Sigma := \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_K^2 \end{bmatrix}.$$

Since the random variable $a\Gamma_k$ will be distributed like a Gamma random variable for any k and $a > 0$, we can assume without loss of generality that $E[\Gamma_i] = 1$.

The modeled covariance matrix $cov_M[N]$ associated with the random vector N is then calculated as:

$$cov_M[N] = QW^T\Sigma WQ + Q \tag{7.11}$$

where $Q = diag(q_1, q_2, \ldots, q_n)$ and $W = w_{ik}{}_{i=1,\,k=1}^{m,k}$

On the other hand, we can calculate the covariance matrix from the known default correlation matrix C:

$$cov_R[N] = [Q(I-Q)]^{1/2}C[Q(I-Q)]^{1/2} + Q.$$

Equating the two covariance matrices $cov_M[N] = cov_R[N]$, we have:

$$QW^T\Sigma WQ = [Q(I-Q)]^{1/2}C[Q(I-Q)]^{1/2}. \tag{7.12}$$

Note again that, the matrix $W \geq 0$ contains the parameter matrix of the model that need to be found. From the relation (7.12), one can therefore approximating the matrix $Q^{-1/2}(I-Q)^{1/2}C(I-Q)^{1/2}Q^{-1/2}$ by using one of the Symmetric Nonnegative Matrix Factorization algorithms, for example RRISNMF (Algorithm 13). To have the matrices W and Σ from the resulted approximation UU^T, one needs only to choose the appropriate positive weight matrix Σ and compute $W = U\Sigma^{-1/2}$.

In [119], we carried out this idea to the real data matrix taken from the default correlation data between clients of a Belgian bank. Because it has millions of clients and the correlation matrix is in general not sparse, the corresponding correlation matrix will be very large even for storing, if we take into account all the clients. One preprocessing step is performed to divide its clients into clusters and to compute only the correlation information between these clusters. The current version of this clustered correlation matrix has dimension 1513 which is quite reasonable. All experiments in this section are carried out on this matrix.

In order to obtain a low-rank non-negative approximation of the correlation matrix, the RRISNMF algorithm and the symmetric multiplicative rules (SMULT) have been used for increasing values of the number of factor K of systematic risks. This number K is also equal to the rank of the non-negative factorization of C. Figure 7.9, 7.10 and 7.11 present the behavior of both algorithms (RRISNMF and SMULT) in calculating low-rank approximations (rank = 5, 20 and 50) of the correlation matrix. It is easy to see that the RRISNMF algorithm converges in

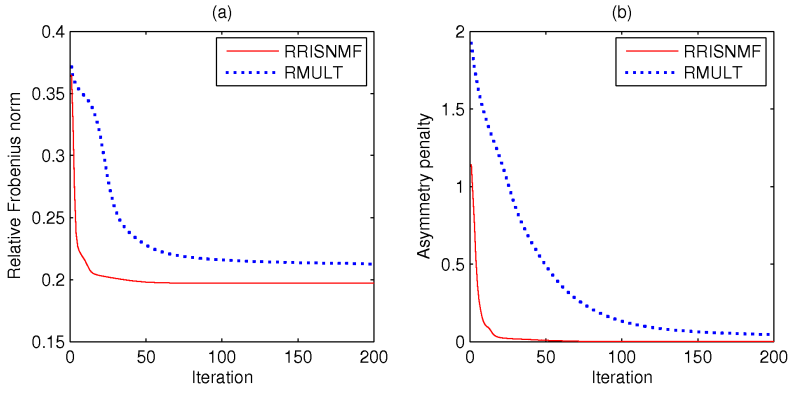


Figure 7.9: Rank-5 Approximation of the Correlation Matrix. (a) shows the relative approximation error $\|C - UV^T\|_F / \|C\|_F$ and (b) shows the asymmetry penalty $\|U - V\|_F$.

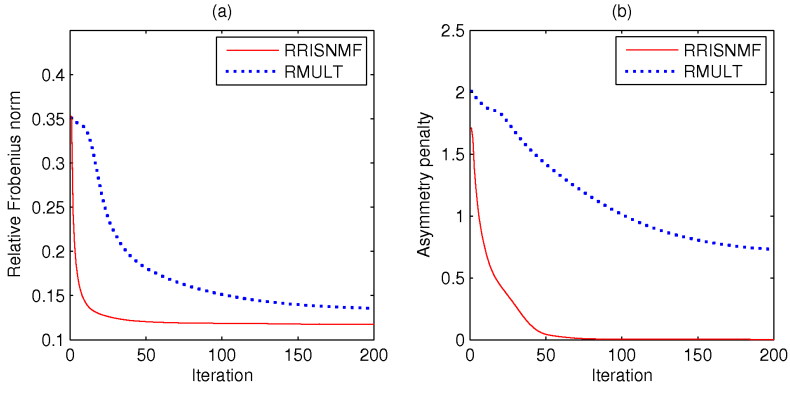


Figure 7.10: Rank-20 Approximation of the Correlation Matrix. (a) shows the relative approximation error $\|C - UV^T\|_F / \|C\|_F$ and (b) shows the asymmetry penalty $\|U - V\|_F$.

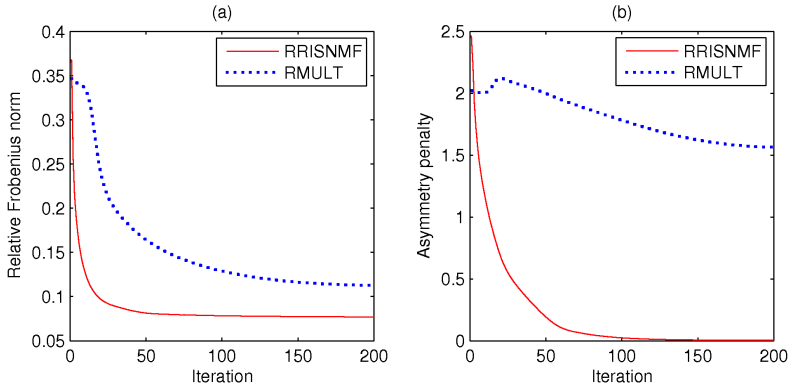


Figure 7.11: Rank-50 Approximation of the Correlation Matrix. (a) shows the relative approximation error $\|C - UV^T\|_F / \|C\|_F$ and (b) shows the asymmetry penalty $\|U - V\|_F$.

less iterations than the SMULT algorithm. Both the approximation error $\|C - UV^T\|_F$ and the asymmetry penalty $\|U - V\|_F$ decrease faster with the RRISNMF algorithm.

In Figure 7.12, the error between the approximated non-negative factorization and C is compared with the error obtained without imposing the non-negativity of the factorization (i.e. by using the SVD decomposition).

Given the approximation of successive ranks, one may ask whether these approximations are coherent. We use the canonical angles between two subspaces to measure the coherence between the successive approximation. Given two approximation UU^T and $U'U'^T$ of C, we first calculate their SVD:

$$UU^T = E\Sigma E^T \text{ and } U'U'^T = E'\Sigma' E'^T$$

where E and E' are orthonormal; Σ and Σ' are diagonal. The canonical angles θ_t between E and E' are defined by $cos(\theta_t) = \sigma_t$ where $\sigma_1, \sigma_2, \ldots, \sigma_k$ are the singular values of the matrix $E^T E'$. The canonical angles are close to 0 when the singular values are close to 1. We calculate the cosines of the canonical angles between all successive approximations $U_i U_i^T$ and $U_{i+1} U_{i+1}^T$ for all $i = 1, 2, ..., 49$ and plot them in Figure 7.13. For each pair of approximations, i.e. $U_i U_i^T$ and $U_{i+1} U_{i+1}^T$, we connect all the cosines

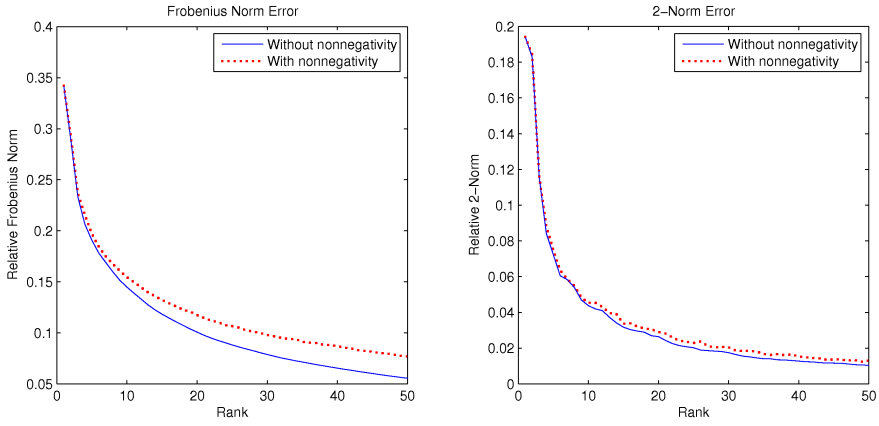


Figure 7.12: Low Rank Approximation of the Correlation Matrix

by a solid line. One can see that most of the angles are close to 0, which shows some agreement between the successive spaces.

Note that, for each approximation, a new initial approximate factorization is generated. One can use the rank-i approximation to create an initial point for the rank-$(i+1)$ approximation by adding a random column to U. This will make approximations even more coherent between the successive ranks.

As we can see, the RRISNMF algorithm performs very well in practice. With this approach, using 10 systematic risks instead of using a single factor approach allowed to reduce the error on the correlations by a factor 3, while still preserving the numerical tractability of the model. We believe that this is a substantial improvement.

Note that the RRISNMF algorithm can be used in any problem where the entries of the matrix C or any product of the type $v^T C v$ have to be evaluated frequently, since the complexity for this is substantially reduced when using the low rank approximation, while the accuracy is still very good.

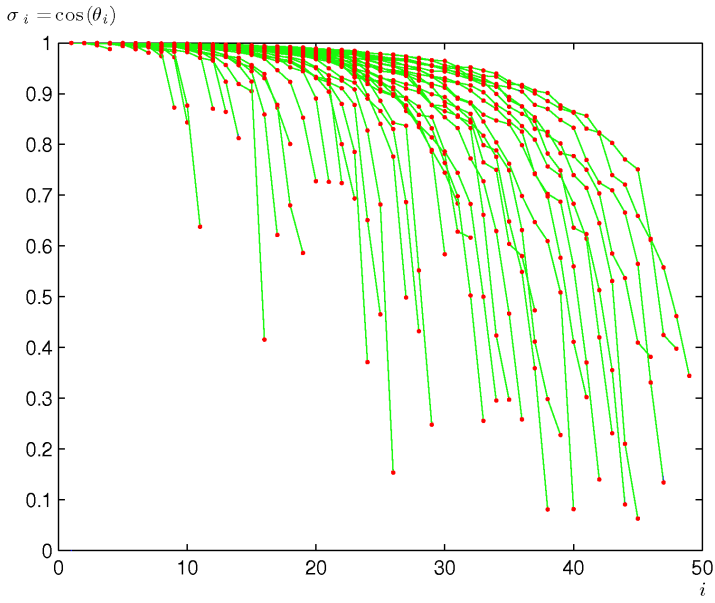


Figure 7.13: Cosine of the canonical angles between successive approximations

CONCLUSION

Starting from some problems of data analysis in large graphs, the first results of this research came from some large nonnegative matrices representing the adjacency matrix of graphs. Many proposed methods such as the Markov Graph Clustering (MCL) [117] work very well for small problems. But applying these to large scale problems require a large memory space and computational cost. This tells us that *low complexity* should be one of the main criteria considered when choosing a method.

A further study of clustering methods reveals that clustering algorithms actually provide a way to reduce the complexity of the data by exploiting the cluster structure. For example, the $K-means$ method [72] represents each group of data points by its centroid. The whole dataset is then represented by the centroids only, which provide a great complexity reduction. However, this yields a big approximation error. The advantage of $K-means$ methods is that the centroid is always situated in the proximity of the data and reflects the physical interpretation of the represented group.

Returning to large nonnegative matrices, our objective was set to find approximation algorithms which can yield a small error while keeping the physical representation of the data. Then the nonnegative matrix factorization [80] was a perfect starting point. Algorithms proposed by this paper possess a low complexity and produces nonnegative bases, which matches our expectation. Studying this method, we came to the following conclusions:

- The convergence of these algorithms requires more investigation. One can obtain a non stationary point if zeros are not properly

treated, as shown in Section 3.1.

- A nice property of the nonnegative matrix factorization can be established when using the generalized Kullback-Leibler divergence as the cost function. It can be seen in Chapter 5 that, the row and column sums are preserved in the approximation.

- Multiplicative rules can be extended to the weighted cases. And thanks to this, an interesting link is revealed between the minimization of the weighted Euclidean distance and the generalized Kullback-Leibler divergence. The use of weights is discussed in Chapter 6.

- A special case where a symmetric factorization UU^T is required links the nonnegative matrix factorization to the class of completely positive matrices. While exact methods have exponential complexity, approximative methods can be derived from the algorithm of Lee and Seung. This proves useful when applied to the problem of approximating correlation matrices and clustering vertices of graphs as shown in Section 7.

While the convergence of the multiplicative rules remains unclear, we seek for alternative methods. We remark that all methods are using the coordinate descent scheme, i.e. only some search directions are allowed. This breaks down the problem into smaller ones that are usually convex. The popular partition of variables allows us to create some completely decoupled problems which sometimes can be solved in parallel. But although the elementary problem is convex, the solution of the inner loop can be very time consuming. Improvements are made by replacing the optimal solution of the elementary problem by a suboptimal solution. This does speed up the algorithms but the convergence property is lost.

In the search for a new algorithm, we then propose another way to partition the variables. The approximation matrix UV^T is now partitioned into a combination of rank-one nonnegative matrices $u_i v_i^T$. Breaking the nonnegative matrix factorization problem into smaller problems according to the sub-variables u_i's and v_i's allows us to formulate a closed form solution for each of the elementary problems. Because the problem is reduced to a problem of iteratively approximating a residue

matrix by a rank-one nonnegative matrix, this method is called *Rank-one Residue Iteration*. The development of the method is presented in Chapter 4, where a good convergence property is also established. Two other independent reports [31] and [49] also proposed this algorithm.

The new algorithm is tested on some numerical experiments to show that it converges faster than the existing algorithms. Without parameters, it is ready to apply to any application. The algorithm requires at each iteration only one matrix-vector multiplication, and therefore has a low complexity. We summarize here some other issues around the *Rank-one Residue Iteration*:

- It can be applied to other classes of nonnegative approximations such as: discrete approximations, sparse approximations, etc. The choice of constraints can be different from column to column of U and V, cfr. Section 4.4.

- Algorithms can be derived for the multilayer nonnegative matrix factorization (see Section 4.6.1) and the nonnegative tensor factorization (cfr. Section 4.6.2). The numerical experiments at the end of Chapter 4 suggest that the nonnegative tensor factorization can provide comparable results with that of the SVD, given the same compression rate.

- It can be adapted to take into account weights, see Section 6.2.2 and to construct the symmetric factorizations of type UU^T and USU^T, cfr. Section 7.2.2 and Section 7.2.2.

There remains several open issues. The first one is the convergence of the multiplicative rules. While the algorithm may return non stationary points, it is still uncertain about whether this is solely due to the finite precision of the computer. Section 3.1 points out that it is probably true that starting from a positive point, with a clever treatment of the zeros, one could improve the convergence of this elegant algorithm.

The second issue is about the existence of many local minima. When representing the rank-k matrix by the product UV^T with appropriate sizes, too many degrees of freedom are added to the problem. This creates many local minima. Current applications often take one of these local minima ignoring the others. It is shown through an example in

Section 7.2.2 that not every local minimum is good. While the use of nonnegative matrix factorization generates an asymmetric approximation with large error, this is shown that by controlling the minimization procedure to a symmetric factorization, the result is significantly improved.

Studying these issues definitely opens many promising research directions. Here are a number of issues that could bring many immediate consequences:

- designing good and cheap techniques to initialize any nonnegative matrix factorization algorithm,
- creating a heuristic of which constraints need to be imposed to obtain a desired approximation,
- testing the block version of RRI algorithm. At the time of writing, it is possible to create updates for two columns of U (or of V) at the same time. We should then ask if it is possible to update even more columns simultaneously and in which order,
- searching for direct methods for the symmetric and semi-symmetric nonnegative matrix factorization to avoid the use of regularization,
- studying higher dimensional analysis such as the nonnegative tensor factorization,
- and searching for methods with a algorithmic complexity bound.

BIBLIOGRAPHY

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Reading, MA, 1974.

[2] R. Albright, J. Cox, D. Duling, A.N. Langville, and C.D. Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *Preprint*, 2006.

[3] S. Ambikkumar and S.W. Drury. A reverse Hadamard inequality. *Linear Algebra and its Applications*, 247:83–95, 1996.

[4] M. Avriel. *Nonlinear Programming: analysis and methods*. Dover Publications, 2003.

[5] B.W. Bader and T.G. Kolda. Efficient MATLAB computations with sparse and factored tensors. Technical Report SAND2006-7592, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Dec. 2006.

[6] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D.S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 509–514, 2004.

[7] F. Barioli and A. Berman. The maximal cp-rank of rank k completely positive matrices. *Linear Algebra and its Applications*, 363:17–33, 2003.

[8] A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Siam, 1994.

[9] A. Berman and U.G. Rothblum. A note on the computation of the CP-rank. *Linear Algebra and its Applications*, 419(1):1–7, 2006.

[10] A. Berman and N. Shaked-Monderer. *The completely positive matrices*. World Scientific, 2003.

[11] M.W. Berry and M. Browne. Email surveillance using nonnegative matrix factorization. *Computational and Mathematical Organization Theory*, 11(3):249–264, 2005.

[12] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.

[13] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, Mass, 1999.

[14] D.P. Bertsekas, A. Nedić, and A.E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific Belmont, Mass, 2003.

[15] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[16] M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. *University of Waterloo, Preprint*, 2007.

[17] A. Bjorck. *Numerical methods for least squares problems*. SIAM Philadelphia, 1996.

[18] V.D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.

[19] V.D. Blondel, N.-D. Ho, and P. Van Dooren. Nonnegative matrix factorization - applications and extensions. Technical Report 005-35, Cesame. University catholique de Louvain. Belgium. 2005, 2005.

[20] C. Boutsidis and E. Gallopoulos. SVD based initialization: a head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[21] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[22] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA 2003)*, pages 568–579. Springer, 2003.

[23] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[24] R. Bro and S. De Jong. A fast non-negativity constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

[25] M. Catral, L. Han, M. Neumann, and RJ Plemmons. On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices. *Linear Algebra and its Applications*, 393:107–126, 2004.

[26] M. Chu, F. Diele, R. Plemmons, and S. Ragni. Optimality, computation and interpretation of nonnegative matrix factorizations. *SIAM Journal on Matrix Analysis*, 2004.

[27] M.T. Chu, R.E. Funderlic, and R.J. Plemmons. Structured low rank approximation. *Linear Algebra and its Applications*, 366:157–172, 2003.

[28] A. Cichocki, S. Amari, and R. Zdunek. Extended SMART algorithms for non-negative matrix factorization. In *Proceeding of the Eighth International Conference on Artificial Intelligence and Soft Computing*, pages 548–562, 2006.

[29] A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization. *Electronics Letters*, 42(16):947–948, 2006.

[30] A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: family of new algorithms. *Lecture Notes in Computer Science*, 3889:32–39, 2006.

[31] A. Cichocki, R. Zdunek, and S. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In *Proceedings of 7th International Conference on Independent Component Analysis and Signal Separation, ICA 2007, London, UK*, pages 169–176, 2007.

[32] A. Cichocki, R. Zdunek, and S. Amari. Nonnegative matrix and tensor factorization. *Signal Processing Magazine, IEEE*, 25(1):142–145, 2008.

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

[34] M. Cooper and J. Foote. Summarizing video using non-negative similarity matrix factorization. In *IEEE Multimedia Signal Processing Workshop*, pages 25–28, 2002.

[35] A. Cutler and L. Breiman. Archetypal Analysis. *Technometrics*, 36(4):338–347, 1994.

[36] I.S. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Proceeding of the Neural Information Processing Systems (NIPS) Conference, Vancouver, BC*, 2005.

[37] C. Ding, X. He, and H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of SIAM Data Mining Conference*, pages 606–610, 2005.

[38] C. Ding, T. Li, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. Technical Report 60428, Lawrence Berkeley National Laboratory, 2006.

[39] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems*, volume 16, pages 1141–1148, 2004.

[40] L. Elsner, R. Nabben, and M. Neumann. Orthogonal bases that lead to symmetric nonnegative matrices. *Linear Algebra and its Applications*, 271(1-3):323–343, 1998.

[41] J. Falkner, F. Rendl, and H. Wolkowicz. A computational study of graph partitioning. *Mathematical Programming*, 66(1):211–239, 1994.

[42] L. Finesso and P. Spreij. Nonnegative matrix factorization and i-divergence alternating minimization. *Linear Algebra and its Applications*, 416:270–287, 2006.

[43] R. Fisher. Iris flower dataset. *http://en.wikipedia.org/wiki/Iris_flower_data_set*.

[44] V. Franc, V. Hlaváč, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *CAIP 2005: Computer Analysis of Images and Patterns*, volume 3691, pages 407–414, September 2005.

[45] H.T. Gao, T.H. Li, K. Chen, W.G. Li, and X. Bi. Overlapping spectra resolution using non-negative matrix factorization. *Talanta*, 66(1):65–73, 2005.

[46] Y. Gao and G. Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[47] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 601–602. ACM Press New York, NY, USA, 2005.

[48] N. Gillis. Approximation et sous-approximation de matrices par factorisation positive: algorithmes, complexité et applications. M.S. Thesis, Université catholique de Louvain, 2007.

[49] N. Gillis and François Glineur. Nonnegative matrix factorization and underapproximation. Preprint, 2008.

[50] G. Golub and C. F. Van Loan. *Matrix computations.3rd ed.* Baltimore, The Johns Hopkins Univ. Press. xxvii, 694 p. , 1996.

[51] E.F. Gonzales and Y. Zhang. Accelerating the Lee-Seung algorithm for nonnegative matrix factorization. Technical Report TR-05-02,

Department of Computational and Applied Mathematics, Rice University, 2005, 2005.

[52] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[53] D. Guillamet, M. Bressan, and J. Vitrià. A weighted nonnegative matrix factorization for local representations. In *IEEE Computer Society Conference on Compute Vision and Pattern Recognition*, pages 942–947, 2001.

[54] D. Guillamet and J. Vitria. Non-negative matrix factorization for face recognition. *Lecture Notes on Artificial Intelligence*, 2504:336–344, 2002.

[55] D. Guillamet, J. Vitrià, and B. Schiele. Introducing a weighted non-negative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.

[56] M. Heiler and C. Schnörr. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *The Journal of Machine Learning Research*, 7:1385–1407, 2006.

[57] N.J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra Application*, 103:103–118, 1988.

[58] N.J. Higham, MJC Gover, and S. Barnett. Matrix nearness problems and applications. *Applications of Matrix Theory*, pages 1–27, 1989.

[59] N.-D. Ho, V. Blondel, and P. Van Dooren. Weighted nonnegative matrix factorization and face feature extraction. *Submitted to Image and Vision Computing*, 2007.

[60] N.-D. Ho and F. Cédrick. Lexical similarity based on quantity of information exchanged - synonym extraction. In *Proceedings of the Research Informatics Vietnam-Francophony, Hanoi, Vietnam*, pages 193–198, 2004.

[61] N.-D. Ho and P. Van Dooren. On the pseudo-inverse of the Laplacian of a bipartite graph. *Applied Mathematics Letters*, 18(8):917–922, 2005.

[62] N.-D. Ho, P. Van Dooren, and V.D. Blondel. Descent algorithms for nonnegative matrix factorization. Technical Report 2007-57, Cesame. University catholique de Louvain. Belgium. 2007. *To appear in Numerical Linear Algebra in Signals, Systems and Control.*, 2007.

[63] N.-D. Ho, P. Van Dooren, and V.D. Blondel. Descent algorithms for Nonnegative Matrix Factorization. *Survey paper. To appear in Numerical Linear Algebra in Signals, Systems and Control*, 2008.

[64] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57. ACM Press New York, NY, USA, 1999.

[65] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1/2):177–196, 2001.

[66] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press. XIII, 561 p., 1985.

[67] R. A. Horn and C. R. Johnson. *Topics in matrix analysis.* Cambridge University Press. viii, 607 p. , 1991.

[68] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[69] P.O. Hoyer. Non-negative sparse coding. In *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[70] S. Impedovo, L. Ottaviano, and S. Occhinegro. Optical character recognition–a survey. *Internatioanl Journal of Pattern Recoginition and Artificial Intelligence*, 5(1):1–24, 1991.

[71] C.T. Ireland and S. Kullback. Contingency tables with given marginals. *Biometrika*, 55(1):179–188, 2001.

[72] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1988.

[73] M. Kaykobad. On nonnegative factorization of matrices. *Linear Algebra and its Applications*, 96:27–33, 1987.

[74] M.H.V. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Chemometrics*, 18:441–450, 2004.

[75] P. A. Knight and D. Ruiz. A fast algorithm for matrix balancing. In *Web Information Retrieval and Linear Algebra Algorithms*, 2007.

[76] T.G. Kolda and D.P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems (TOIS)*, 16(4):322–346, 1998.

[77] R. Kompass. A generalized divergence measure for nonnegative matrix factorization. *Neural Computation*, 19(3):780–791, 2007.

[78] S. Kullback. *Information theory and statistics*. Courier Dover Publications, 1997.

[79] C.L. Lawson and R.J. Hanson. *Solving least squares problems*. Prentice-Hall Englewood Cliffs, NJ, 1974.

[80] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 1999.

[81] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Neural Information Processing Systems*, pages 556–562, 2000.

[82] D.D. Lee and H.S. Seung. Unsupervised learning by convex and conic coding. *Advances in Neural Information Processing Systems*, 9:515–521, 1997.

[83] S.Z. Li, X.W. Hou, H.J. Zhang, and Q.S. Cheng. Learning spatially localized, parts-based representation. In *Proceedings of IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–6, 2001.

[84] J. Liang and D. Fairley. Validation of an efficient non-negative matrix factorization method and its preliminary application in Central California. *Atmospheric environment*, 40(11):1991–2001, 2006.

[85] C.-J. Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. *IEEE Transactions on Neural Networks*, 2007. To appear.

[86] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 2007. To appear.

[87] D.G. Luenberger. *Linear and nonlinear programming*. Springer, 2003.

[88] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(281-297):14, 1967.

[89] M. Merritt and Y. Zhang. Interior-point gradient method for large-scale totally nonnegative least squares problems. *Journal of Optimization Theory and Applications*, 126(1):191–202, 2005.

[90] E. Moore Gordon. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.

[91] N. Murata, T. Takenouchi, T. Kanamori, and S. Eguchi. Information geometry of U-Boost and bregman divergence. *Neural Computation*, 16(7):1437–1481, 2004.

[92] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[93] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 1999.

[94] P.D. O'Grady and B.A. Pearlmutter. Convolutive non-negative matrix factorisation with a sparseness constraint. In *Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on*, pages 427–432, 2006.

[95] Paatero P. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

[96] P. Paatero. A weighted non-negative least squares algorithm for three-way 'parafac' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2):223–242, 1997.

[97] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(1):111–126, 1994.

[98] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R.D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006.

[99] V.P. Pauca, J. Piper, and R.J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 416(1):29–47, 2006.

[100] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[101] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Par I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals. *Journal of Symbolic Computation*, 13(3):255–299, 1992.

[102] R.T. Rockafellar. *Convex analysis*. Princeton University Press, 1970.

[103] F. Shahnaz, M.W. Berry, V.P. Pauca, and R.J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.

[104] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 792–799, 2005.

[105] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *Proceeding of European Conference on Computer Vision*, pages 595–608. Springer, 2006.

[106] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic. *The Annals of Mathematical Statistics*, 35:876–879, 1964.

[107] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and rolumn sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

[108] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[109] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[110] S. Sra and I.S. Dhillon. Nonnegative matrix approximation: algorithms and applications. Technical Report Tr-06-27, Computer Sciences, University of Texas at Austin, 2006, 2006.

[111] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 720–727, 2003.

[112] Credit Suisse. CreditRisk+. *Credit Suisse Financial Products(1997), http://www.csfb.com/creditrisk*, 1997.

[113] A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, 1951.

[114] J. Tropp. Literature survey: nonnegative matrix factorization. *University of Texas at Asutin, preprint*, 2003.

[115] J. M. Van Den Hof. Realization of positive linear systems. *Linear Algebra and its Applications*, 256(1–3):287–308, 1997.

[116] J.M. Van Den Hof and J.H. Van Schuppen. Positive matrix factorization via extremal polyhedral cones. *Linear Algebra and its Applications*, 293(1-3):171–186, 1999.

[117] S.M. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

[118] C.F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 2000.

[119] A. Vandendorpe, N.D. Ho, S. Vanduffel, and P. Van Dooren. On the parameterization of the CreditRisk model for estimating credit portfolio risk. *Accepted for publication in Insurance: Mathematics and Economics*, 2007.

[120] B. Vanluyten, J.C. Willems, and B. De Moor. Matrix factorization and stochastic state representations. *2006 45th IEEE Conference on Decision and Control*, pages 4188–4193, 2006.

[121] S.A. Vavasis. On the complexity of nonnegative matrix factorization. *eprint arXiv: 0708.4149*, 2007.

[122] Y. Wang, Y. Jia, C. Hu, and M. Turk. Fisher non-negative matrix factorization for learning local features. In *Proceedings of Asian Conference on Computer Vision*, 2004.

[123] D.S. Watkins. *Fundamentals of matrix computations*. John Wiley & Sons, Inc. New York, NY, USA, 1991.

[124] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261, 2001.

[125] S. Wild. Seeding non-negative matrix factorizations with the spherical K-Means clustering. M.S. Thesis, University of Colorado, 2003.

[126] S. Wild, J. Curry, and A. Dougherty. Motivating non-negative matrix factorizations. In *Proceedings of Eighth SIAM Conference on Applied Linear Algebra, Philadelphia*, 2003.

[127] S. Wild, J. Curry, and A. Dougherty. Improving non-negative matrix factorizations through structured initialization. *Pattern recognition*, 37(11):2217–2232, 2004.

[128] W. Xu, X. Liu, and Y. 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 Press New York, NY, USA, 2003.