UTS UNIVERSITY OF TECHNOLOGY SYDNEY

CONSTRAINED LOW-RANK MATRIX/TENSOR FACTORISATION

by Shuai Jiang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Assoc. Prof. Richard Yi Da Xu

University of Technology Sydney
Faculty of Engineering and Information Technology

June 2020

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Shuai Jiang declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

I also certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with Beijing Institute of Technology.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 30th June 2020

ABSTRACT

CONSTRAINED LOW-RANK MATRIX/TENSOR FACTORISATION

by

Shuai Jiang

Constrained low-rank matrix and tensor factorisation (MF/TF) have been widely used in machine learning and data analytics. Studies on the way of modelling constraints and the solution of optimisation task in general can provide theoretical supports for applications like image clustering, recommender systems and data compression. This thesis studies three algorithms of constrained low-rank MF/TF.

Imposing constraints on each feature vector of factor matrices is a common practice in many constrained low-rank MF algorithms. However, in many real scenarios, the relationships among features can influence the factorisation results as well. In order to better characterise the relationships among features, a novel MF algorithm, Relative Pairwise Relationship Constrained Non-negative Matrix Factorisation, is proposed. It places soft constraints over relative pairwise distances amongst features as regularisations to retain expected relationships after factorisation. It conforms to the so-called "multiplicative update rules" and detailed convergence proofs are provided. Experiments on both synthetic and real datasets have verified that imposing such constraints can keep most expected relationships unchanged after factorisation.

Directly adopted on tensor data, low-rank TF can effectively avoid the information loss caused by matricisation. The relationships among features of factor matrices in TF have practical meanings in many real scenarios. To describe such relative relationships in low-rank TF, this thesis proposes Relative Pairwise Relationship Constrained Non-negative Tensor Factorisation. It deals with both Camdecomp/Parafac and Tucker decomposition schemes and both squared Euclidean distance and divergence measures. The utilisation of tensor factorisation matricisation

equation simplifies the update rules and greatly improves the computation efficiency. Experiments have demonstrated that the proposed algorithm can achieve higher accuracy when adopted on tensor applications.

There exists a problem of acquiring out-of-bounds and fluctuating values over predictions when applying low-rank MF on recommender systems. The commonly used solutions, truncation and imposing penalties, can cause the decrease in the number of effective predictions and affect the recommendation accuracy. This thesis creatively proposes Magnitude Bounded Matrix Factorisation to handle the above problem by imposing magnitude constraints for the first time. It first converts the original quadratically constrained quadratic programming task to an unconstrained one which is then solved by the well-known stochastic gradient descent. An acceleration approach for improving computation efficiency, an extracting method for magnitude constraints and a variant of MBMF for non-negative data are also introduced. Experiments have demonstrated that the algorithm is superior to existing bounding algorithms on both computing efficiency and recommendation performance.

Dissertation directed by Assoc. Professor Richard Yi Da Xu
School of Electrical and Data Engineering

Dedication

I dedicate my dissertation work to my family without whom I would never accomplish this project. A special feeling of gratitude to my loving mother, Xueqin Jiang whose words of encouragement and push for tenacity have been helping me all throughout the way.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Professor Richard Yi Da Xu and Professor Kan Li (Beijing Institute of Technology) for their continuous support, patience, motivation, enthusiasm and immense knowledge that guided and helped me in my whole PhD candidature. At many stages I benefited from their advice, particularly so when exploring new ideas. Their positive outlook and confidence in research inspired, encouraged and taught me how to become a researcher.

Also I would like to thank my peers for their selfless help in both my research and daily life. Thank you Dr. Jason Traish, Erica Huang, Ember Liang, Hayden Chang, Kelvin Deng, Sammi Zhao, Leon Li, Zyane Zhang and Carter Huang. Special thanks to my best friend David Li. At last, I wish to thank my lovely cats, Polo and Mango for their warmest companion.

Shuai Jiang

Sydney, Australia, 2020.

List of Publications

Journal Papers

J-1. **S. Jiang**, K. Li, and R. Y. D. Xu. Relative Pairwise Relationship Constrained Non-Negative Matrix Factorisation. *IEEE Transactions on Knowledge and Data Engineering*, 31(8), pp.1595-1609, 2018.

J-2. **S. Jiang**, K. Li, and R. Y. D. Xu. Magnitude Bounded Matrix Factorisation for Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering*, doi: 10.1109/TKDE.2020.2998218, 2020. Online.

J-3. **S. Jiang**, K. Li, and R. Y. D. Xu. Non-negative CP Tensor Decomposition with Relative Pairwise Relationship Regularizations. *Pattern Recognition Letters*, 2020. Under review.

J-4. L. Bai, K. Li, J. Pei, and **S. Jiang**. Main objects interaction activity recognition in real images. Neural Computing and Applications, 27(2), pp.335-348, 2016.

Conference Papers

C-1. C. Huang, **S. Jiang**, Y. Li, Z. Zhang, J. Traish, C. Deng, S. Ferguson, R. Y. D. Xu. End-to-end Dynamic Matching Network for Multi-view Multi-person 3d Pose Estimation. *In Proceedings of the European Conference on Computer Vision (ECCV)*, 1267, 2020.

C-2. Y. Li, K. Li, **S. Jiang**, Z. Zhang, C. Huang, and R. Y. D. Xu. Geometry-driven Self-supervised Method for 3D Human Pose Estimation. *AAAI Conference on Artificial Intelligence (AAAI)*, 7454, 2020.

C-3. Z. Zhang, R. Y. D. Xu, **S. Jiang**, Y. Li, C. Huang, and C. Deng. Illumination adaptive person reid based on teacher-student model and adversarial training. *IEEE International Conference on Image Processing (ICIP)*, 2020.

Contents

List of Figures

Abbreviation

MF - Matrix Factorisation

NMF - Non-negative Matrix Factorisation

CNMF - Constrained Non-negative Matrix Factorisation

MUL - Multiplicative Update Rules

GNMF - Graph Regularised Non-negative Matrix Factorisation

LCNMF - Label Constrained Non-negative Matrix Factorisation

RPR-NMF - Relative Pairwise Relationship Constrained Non-negative Matrix Factorisation

BMF - Bounded Matrix Factorisation

BMC-ADMM - Bounded Matrix Completion in Alternating Direction of Multiplier Method

MBMF - Magnitude Bounded Matrix Factorisation

TF - Tensor Factorisation

NTF - Non-negative Tensor Factorisation

CNTF - Constrained Non-negative Tensor Factorisation

LRNTF - Laplacian Regularised Non-negative Tensor Factorisation

RPR-NTF - Relative Pairwise Relationship Constrained Non-negative Tensor Factorisation

SVD - Singular Value Decomposition

CP - Candecomp/Parafac

Nomenclature and Notation

Lower-case non-bold characters denote iterative variables (e.g. i, j, k, n).

Lower-case bold characters denote vectors (e.g. $\boldsymbol{u}$).

Upper-case non-bold characters denote constant scalars (e.g. N, K, I).

Upper-case bold characters denote matrices (e.g. $\boldsymbol{U}$).

Upper-case non-bold Euler characters denote functions (e.g. $\mathcal{F}$).

Upper-case bold Euler characters denote tensors (e.g. $\boldsymbol{\mathcal{X}}$).

$\boldsymbol{U}_{i:}$ denotes the i^{th} row of matrix $\boldsymbol{U}$.

$(.)^T$ denotes the transpose operation.

$\mathbb{R}$ denotes the field of real numbers.

Chapter 1

Introduction

1.1 Background

Compared to conventional dimensionality reduction methods, such as singular value decomposition (SVD), low-rank matrix/tensor factorisation (MF/TF), mostly solving an optimisation task, converge much faster when it comes down to large real-world datasets [3, 4, 5]. Thus MF/TF have been widely used in many applications [6, 7], and algorithms of this kind have been the research foci in many communities, such as image processing and recommender systems [2, 8, 7, 9, 10].

A widely studied variation of MF/TF is to impose non-negative constraints, which requires both the factorising and factorised matrices/tensors being non-negative. This requirement is useful and practical since many real datasets only have non-negative values. Apart from the traditional gradient descent based solution, a seminal approach in non-negative matrix/tensor factorisation (NMF/NTF) is the so-called "multiplicative update rules" (MUL) which guarantees both the convergence of the algorithm and the non-negativity of factorised matrices [1]. Though MUL has been proved not converging to a stationary point numerically [11], and they are not strictly well-defined because of possible zero entries [12], it practically produces satisfactory results, especially for large scale data, which makes it a popular solution for NMF/NTF. However, the original NMF/NTF only imposes the non-negativity constraints on both of the factorising and factorised matrices, which in practice may not be enough to satisfy additional requirements. Thus, researchers in this area have been proposing new algorithms under this framework to cater for

incremental improvements [13, 14, 15], variations [16, 17, 18], and/or application oriented constraints [19, 20].

A main sub category of NMF/NTF is Constrained Non-negative Matrix/Tensor Factorisation (CNMF/CNTF), which imposes constraints based on variables. The three algorithms studied in this thesis all fall under this sub category, among which two impose regularisation terms while another imposes inequalities.

The most commonly used regularisations for NMF/NTF are L1 and L2 norms [21], the former increases the sparseness of the factorised matrices, while the latter makes the results smooth to prevent overfitting. However, these constraints are only imposed on each of the rows or columns and the relationship among pairwise rows or columns has not been considered (such as to specify constraints on relative distance between row vectors A and B and that of A and C). Such pairwise relationships exists widely in many systems, especially in those systems where the factorised matrices are representing features. A typical instance is the MF technique used in recommender systems.

In a recommender system, the factorising matrix is usually the rating matrix whose entries denote the ratings given by the corresponding user (row) to the corresponding item (column), and the factorised matrices are usually regarded as the user feature matrix (the left factorised matrix) and the item feature matrix (the right factorised matrix). Figure 1.1 shows an example of a simple movie recommender system. In this example, after factorisation by NMF, the distance between "Star Wars" and "Titanic" becomes less than the distance between "Star Wars" and "Star Trek". However, as we all know, the movie "Star Wars" should be closer to the movie "Star Trek" – both are within the scientific fiction genre, unlike "Titanic", which is a love story. Thus intuitively, it will result in a better factorisation and make better recommendations if we can incorporate such human-aware relative

relationships into the NMF model.

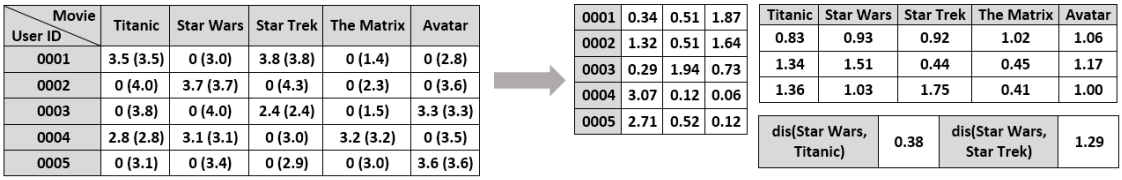


Figure 1.1 : A simple movie recommender system. The factorisation algorithm used here is NMF with Euclidean measure proposed in [1]. Original missing ratings are denoted by 0 and recovered ratings are showed in brackets. As shown under the right factorised matrix, the distance between feature vectors of movies "Star Wars" and "Titanic" is less than the distance between movies "Star Wars" and "Star Trek".

There have been a few studies that attempt to consider the above pairwise relationship in the area of NMF. The techniques used include imposing graph regularisations [22] or partial label constraints [23]. The former requires the construction of a weight matrix which indicates the explicit similarity between pairwise feature vectors, and the latter uses a label matrix to force features being identical if they are labelled the same. Both methods are too restrictive in practice. Chapter 3 introduces a novel algorithm on how to characterise such relationships in NMF.

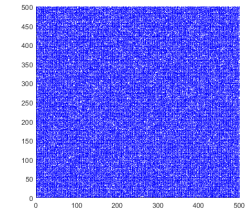
Although NMF algorithms have been widely used in applications, when the data has more than two contexts, applying NMF can cause information loss. For example, in image clustering, if the whole image dataset is modelled by a matrix, then each image data needs to be vectorised as a row/column of the data matrix. The proximity of pixels in an image is lost, so is the independence of pixels in different channels.

One effective way to mitigate the above information loss issue is to apply non-negative tensor factorisation (NTF) directly on tensor data. This is usually done by extending NMF-based algorithms to NTF-based ones [24].

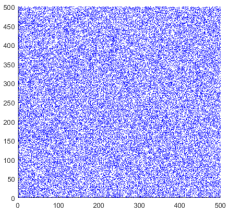
Similar to NMF, the feature vectors in NTF also need to conform to human-aware relative relationships. For example, when applying NTF on a dynamic product recommender system, the three individual factor matrices represent users, products and time-slices respectively. In terms of users, there exist relative distances among user attributes such as gender, age, and occupations, which should be reflected by feature vectors in the user matrix, e.g. features associated with two young female lawyers should be made closer than those between one of them and a middle-aged male constructor. Similar relative relationships can be found among products or time-slices. This is akin to the assumptions in metric learning: some data should be closer to each other than others and some should be further apart. However, the factorisation results from vanilla NTF may violate above relative relationships, thus additional constraints or regularisations are needed to retain them. Chapter 4 studies how to impose such regularisations in NTF.

Apart from NMF/NTF, another popular research topic in constrained MF/TF is to impose bounding constraints, especially in MF algorithms designed for recommender systems. The reason to impose such constraints is, many of the recommender systems have predefined ranges for entries, and the predictions of missing entries from MF algorithms are often unstable and out-of-range [2].

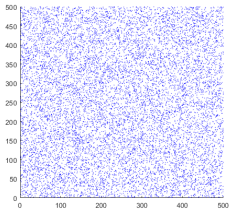
This effect can be demonstrated through an example of a synthetic recommender system in Figure 1.2. In this example, three sparse matrices are randomly generated with the same size of 500×500 and range of $[1, 10]$ but different densities: 0.8, 0.2 and 0.05 (denoted by blue dots in Figure 1.2 (a), (b) and (c)). Then the MF method [2] is run on each sparse matrix to get predicted values (denoted by red dots in Figure 1.2 (d), (e), (f) and (g)). Meanwhile, the maximal and minimal predicted values (denoted by $\max(\hat{V})$ and $\min(\hat{V})$) is recorded, and the maximal and average standard deviations (denoted by $\max(\sigma)$ and $\text{ave}(\sigma)$) is calculated, which are presented in Table 1.1. As seen from the results, the increase in missing



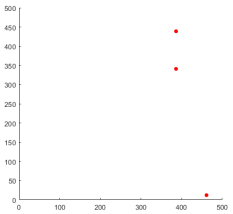
(a) sparse matrix density = 0.8.



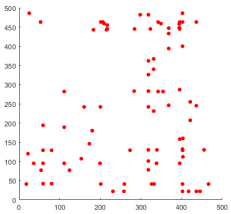
(b) sparse matrix density = 0.2.



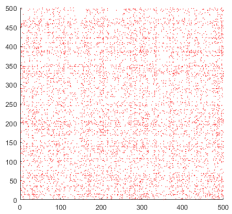
(c) sparse matrix density = 0.05.



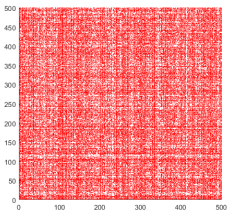
(d) average of 10 repeats, only 3 out-of-bounds predicted values (0.001%).



(e) average of 10 repeats, several out-of-bounds predicted values (0.034%).



(f) average of 10 repeats, many out-of-bounds predicted values (2.781%).



(g) single run, large quantities of out-of-bounds predicted values (34.69%).

Figure 1.2 : Predicted values obtained from MF algorithm [2] with respect to different densities of factorised matrices. Synthetic data are denoted by blue dots in (a), (b) and (c) and the corresponding out-of-bounds predicted values are denoted by red dots in (d), (e), (f) and (g). Note that the red dots in (d) and (e) are depicted 20 times bigger than those in other subfigures to ensure they are clearly visible.

entries led towards not only more out-of-bounds predicted values for more red dots appeared, but also higher prediction fluctuations as both the max(σ) and ave(σ) became greater. Furthermore, according to Figure 1.2 (f) and (g) and the last two rows in Table 1.1, repeating the factorisation algorithm and using the average predicted values can significantly leverage the bounding issues. However, this will cause an increase in computational cost, especially when large datasets are involved.

Table 1.1 : Statistics of Predicted Values in the Synthetic Example

density	error	#oob	$\max(\hat{V})$	$\min(\hat{V})$	$\max(\delta)$	$\text{ave}(\delta)$
0.8	7.25	3	10.69	-0.38	2.08	0.51
0.2	5.25	86	11.67	-1.84	8.98	1.68
0.05	0.29	6953	18.88	-8.67	22.76	5.36
0.05(1)	0.27	86720	62.72	-45.00	-	-

Bounding predicted values and reducing fluctuation on predictions can help improve recommendation accuracy on large and sparse datasets. Since many recommendation systems have a single predefined range for all entries, one straightforward way to reduce fluctuation over predictions is to limit the product of the factorised matrices, to make them fall within a predefined range. There are two types of algorithms in literature seeking to address this bounding problem: one is to project the products that are out-of-range to the boundaries [25, 26], and the other one is to impose strong penalties in the objective function if the products are out-of-range [27]. Projecting (or mapping) methods require in range or out-of-range checks on each of the product entries obtained by the multiplication of factorised matrices, which causes the issue of high computation time especially on large scale dataset (see Section 5.7), while the performance of the penalty methods highly depends on the setting of their penalty coefficients.

In addition to the above mentioned recommendation systems with predefined single range, there are many other ones where data does not share a single range, i.e. users/items' data are not in the same magnitude. For instance, in "user-location check-in frequency" dataset [28], the matrix contains the frequency counts of users appearing in different locations, where the scales of frequencies vary from user to

user and location to location. Another example is the Ali Mob dataset, where some people browsed and bought a lot of products while others might only do a few. In these cases, the above two bounding methods may fail because they can only address datasets with single range: imposing a single pair of bounds, for example, using the maximum and minimum of all known values, will have very little effect on bounding most of the entries, as they are all far from the end points of this range. From another aspect, even when dealing with datasets which have a single predefined range, such as movie ratings in Movielens which are between 1 and 5, bounding the entries in the product matrix into different and more concise ranges (e.g. related to the corresponding user or item) is also likely to gain better recommendations because users/items can have different preferences which can be reflected by their own bounds. More details will be discussed and a novel bounding algorithm will be introduced in Chapter 5.

1.2 Objectives and Contributions

For the aforementioned research problems, this thesis studies and introduces three constrained low-rank MF/TF algorithms: Relative Pairwise Relationship Constrained Non-negative Matrix Factorisation (RPR-NMF), Relative Pairwise Relationship Constrained Non-negative Tensor Factorisation (RPR-NTF) and Magnitude Bounded Matrix Factorisation (MBMF).

Rather than using explicit similarities or previously known labels, RPR-NMF imposes penalties for relative pairwise relationships (RPRs) in a triplet form. The penalties are not limited to be within $[0, 1]$ as for similarities or to be binary values as for labels. To make the algorithm generic and flexible, penalties are imposed on both factorised matrices, and in a setting where only constraining one matrix is needed, constraints of the second can be simply "turned-off". Both of the squared Euclidean distance and the divergence measure are used in the objectives of RPR-

NMF, and the penalties are in exponential and hinge loss forms respectively. The update rules for RPR-NMF conform to the well-known MUL in which the proofs of convergence are essential for the algorithm. Due to the complexity of proof brought by the imposed penalties, partial terms are approximated in the proofs and the algorithm has been verified practically beneficial through numerous experiments.

To describe such relative relationships among features in low-rank TF, RPR-NTF imposes the RPRs on the factor matrices as regularisation terms for the restrictions on retaining expected relative relationships. There are several challenges for RPR-NTF to deal with. First of all, TF has two different basic schemes: the Candecomp/Parafac (CP) decomposition and the Tucker decomposition. Combined with the two distance measures, there are four objective functions that RPR-NTF needs to solve. Second, tensor decomposition involves more calculations such as outer product and n-mode tensor-matrix multiplications rather than sole matrix operations, which causes more difficulties when deducing the update rules. Conforming to the MUL is yet another challenge for RPR-NTF.

As with other bounding approaches, MBMF deals with the fluctuation of recommendations by bounding the entries in the product matrix into ranges. The main difference is that MBMF can be applied to datasets with or without predefined single range and allows the product entries to have individually different bounds. In MBMF, first, an optimisation algorithm is devised where individual range constraints are placed onto the magnitude of latent feature vectors. This allows MBMF to deal with any datasets for which the magnitudes can be set using context-specific information. Secondly, in order to solve this constrained optimisation problem, the original constrained problem is converted into an unconstrained one by using cartesian to spherical coordinate conversions. These conversions make MBMF unique from all other existing bounding algorithms. It neither projects the product values to the boundaries nor imposes penalty terms to constrain the factorisation, and it

can be solved efficiently by unconstrained optimisation solvers or algorithms, for example, the widely used stochastic gradient descent (SGD) approach. Although the unconstrained optimisation task seems easy to be solved, the representation complexity brought by the spherical coordinate makes it hard to compute the derivatives. Therefore, an elegant acceleration approach is proposed for MBMF based on the discovery of patterns of derivatives which involves tensor construction and multiplication. Further, the choice of magnitudes from predefined range or historical data is also explored in detail. In order to demonstrate the efficiency of the proposed MBMF, sufficient numerical experiments have been conducted to analyse its time complexity with comparative algorithms. In the experiments, both synthetic and real-world datasets are used to verify the effectiveness of MBMF, and the results show that MBMF is superior to existing bounding algorithms regarding recommendation accuracy in most cases.

The main contributions of this thesis are as follows:

(1) The use of RPRs is first proposed to characterise the relative pairwise relationships among features in NMF/NTF. Different from existing studies on this topic, RPR-NMF and RPR-NTF can guarantee the expected RPRs retained after factorisation and does not limit the feature vectors to be identical. Both algorithms conform to MUL and the detailed convergence proofs are given.

(2) For both CP and Tucker tensor decomposition, different RPR-NTF algorithms are introduced for both squared Euclidean distance and Divergence measures. It simplifies the update rules by the utilisation of tensor-matricisation equations which greatly improves its computation efficiency.

(3) MBMF is the first bounding algorithm in the literature which allows different bounds for each entry based on MF framework for recommender systems. The magnitude constraints and coordinate conversions used in MBMF are totally different

from all existing bounding algorithms which are either projecting product values to boundaries or imposing penalty constraints. It is so far the fastest bounding algorithm, especially on very large datasets.

1.3 Thesis Organisation

This thesis is organised as follows:

- *Chapter 2*: presents a comprehensive literature survey to this work.
- *Chapter 3*: demonstrates the RPR-NMF algorithm in detail.
- *Chapter 4*: demonstrates the RPR-NTF algorithm in detail.
- *Chapter 5*: demonstrates the MBMF algorithm in detail.
- *Conclusion*: gives a brief summary of the thesis contents and its contributions.

Chapter 2

Literature Survey

Constrained low-rank MF/TF is a fundamental research topic which has been well studied since 2000. In this chapter, related works to the constrained low-rank MF/TF is comprehensively reviewed.

2.1 Constrained Low-rank Matrix Factorisation

MF is one of the most widely used technique in machine learning area which originates from linear algebra. Some commonly seen MF approaches include full rank factorisation, lower-upper (LU) factorisation and SVD. In machine learning, SVD is often used for dimensionality reduction and principal component analysis (PCA). However, it suffers from the high computation cost when dealing with large data because of its complexity of $O(n^3)$. Thus, low-rank MF was proposed to accelerate the factorisation by the sacrifice in accuracy.

The low-rank MF is defined as an optimisation problem. Given a matrix $\boldsymbol{V} \in \mathbb{R}^{N\times M}$, find factor matrices $\boldsymbol{W} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{H} \in \mathbb{R}^{K\times M}$ such that:

$$\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}, \tag{2.1}$$

where K is the predefined latent dimension and $K \ll \min(N, M)$. There are two measures commonly used in literature to evaluate the above approximation. One is the squared Euclidean distance

$$E(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^2, \tag{2.2}$$

and the other is the Divergence

$$D(\boldsymbol{x}||\boldsymbol{y}) = \sum_{i=1}^{K} \boldsymbol{x}_i \log \frac{\boldsymbol{x}_i}{\boldsymbol{y}_i} - \boldsymbol{x}_i + \boldsymbol{y}_i. \tag{2.3}$$

Note that when the variables are two unit vectors/distributions, the Divergence becomes KL-Divergence.

When used as objective functions to evaluate low-rank MF, the above two measures are both non-convex. However, they are convex for $\boldsymbol{W}$ and $\boldsymbol{H}$ individually when fixing the other one. Therefore, it can be solved by using gradient descent (GD). Take the objective function in the squared Euclidean distance as an example. The update rules using gradient descent are [1]:

$$\begin{aligned} \boldsymbol{W} &\leftarrow \boldsymbol{W} - \lambda_{\boldsymbol{W}}(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{V}\boldsymbol{H}^T), \\ \boldsymbol{H} &\leftarrow \boldsymbol{H} - \lambda_{\boldsymbol{H}}(\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^T\boldsymbol{V}), \end{aligned} \tag{2.4}$$

where $\lambda_{\boldsymbol{W}}$ and $\lambda_{\boldsymbol{H}}$ are the learning rates.

2.1.1 Non-negative Matrix Factorisation

NMF is to impose non-negativity on both the factoring and factorised matrices in MF. Such non-negativity is meaningful in practice because many real-world data are non-negative, such as the image pixel values ranging from 0 to 255 and the Netflix movie ratings between 1 and 5. NMF cannot be solved directly by GD but projected GD which simply uses GD to update factor matrices and project negative entries to zeros. Another approach is to use MUL [1] whose update rules are as follows:

$$\boldsymbol{W} \leftarrow \boldsymbol{W}\frac{\boldsymbol{V}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \qquad \boldsymbol{H} \leftarrow \boldsymbol{H}\frac{\boldsymbol{W}^T\boldsymbol{V}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}. \tag{2.5}$$

MUL is actually a GD with special learning rates: $\lambda_{\boldsymbol{W}} = \frac{\boldsymbol{W}}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}$, $\lambda_{\boldsymbol{H}} = \frac{\boldsymbol{H}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}$.

Different from the projected GD, MUL guarantees not only the non-negativity through its update formulas but also the algorithm's convergence. Assume the

objective function is denoted by $\mathcal{F}$. The key idea behind MUL is to construct an auxiliary function $\mathcal{G}$ and iteratively use its minima to approach the minima of $\mathcal{F}$, as shown in Figure 2.1. $\mathcal{G}$ has the following two properties: (1) $\mathcal{G}(h, h') \geqslant \mathcal{F}(h)$, and (2) $\mathcal{G}(h, h) = \mathcal{F}(h)$.

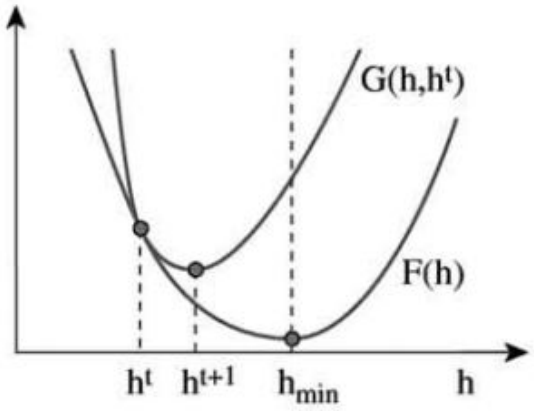


Figure 2.1 : Auxiliary function in MUL [1]

Since the seminal work of MUL, a number of approaches followed pursuit dealing with various NMF issues from different aspects, for example, imposing extra constraints on top of the non-negativity, which has formed a NMF subcategory called CNMF. It was first proposed in [21] and has the following representation:

$$\min_{\boldsymbol{W},\boldsymbol{H}} \{\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \alpha J_1(\boldsymbol{W}) + \beta J_2(\boldsymbol{H})\}, \tag{2.6}$$

for $\boldsymbol{W} \geqslant 0$ and $\boldsymbol{H} \geqslant 0$, where $\|\cdot\|_F$ is the Frobenius norm, α and β are regularisation coefficients. The functions of $J_1(\boldsymbol{W})$ and $J_2(\boldsymbol{H})$ are regularisation terms used to enforce certain constraints on the solution of Eq. (2.6). In their work, the penalties are set $J_1(\boldsymbol{W}) = \|\boldsymbol{W}\|_F^2$ and $J_2(\boldsymbol{H}) = \|\boldsymbol{H}\|_F^2$ in order to strengthen the smoothness in $\boldsymbol{W}$ and $\boldsymbol{H}$ respectively.

Many studies followed the above CNMF framework. [29] used an adaptive potential function as penalties to characterise the piece-wise smoothness of spectral data. [30] imposed three additional constraints on the NMF basis to reveal local

features. [31] imposed both L1 and L2 norms to control the shape of base matrix and increase the sparseness of the coefficient matrix so as to enhance the clustering performance on multiple manifolds.

2.1.2 Relationship Regularisations among Feature Vectors

Most CNMF algorithms imposed regularisations on individual feature vectors, such as the above mentioned L1 and L2 norms. Few considered the relationship among them. There are two existing algorithms which take the relationship among vectors of factorised matrices into consideration: Graph Regularised Non-negative Matrix Factorisation (GNMF) [22], and Labelled Constrained Non-negative Matrix Factorisation (LCNMF) [23]. To better demonstrate the pros and cons of GNMF and LCNMF, Figure 2.2 shows an example of projecting points where the projected points are obtained by using GNMF, LCNMF and RPR-NMF (see Chapter 3).

GNMF, also a CNMF algorithm, is to utilise the relationship among factorised rows or columns for a dimensionality reduction issue. It is in an effort to ensure that similarities between data points at the original space are also retained after they are transformed to the low dimensional subspace through factorisation. Its objective function with Euclidean measure is as following:

$$\mathcal{F} = \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \lambda \operatorname{Tr}(\boldsymbol{H}\boldsymbol{L}\boldsymbol{H}^T), \tag{2.7}$$

where $\boldsymbol{L}$ is a graph Laplacian matrix obtained from the similarity matrix and λ is a regularisation coefficient. As showed in the objective function, GNMF tries to minimise two parts simultaneously: the squared errors between the product of factorised matrices and the factorising matrix, and the similarity matrix. The ideal solution is to minimise them at the same time. However, it often happens that if GNMF minimised the squared error, the relationships implied by similarities might not be satisfied, especially when the similarities are not binaries. As shown in Figure

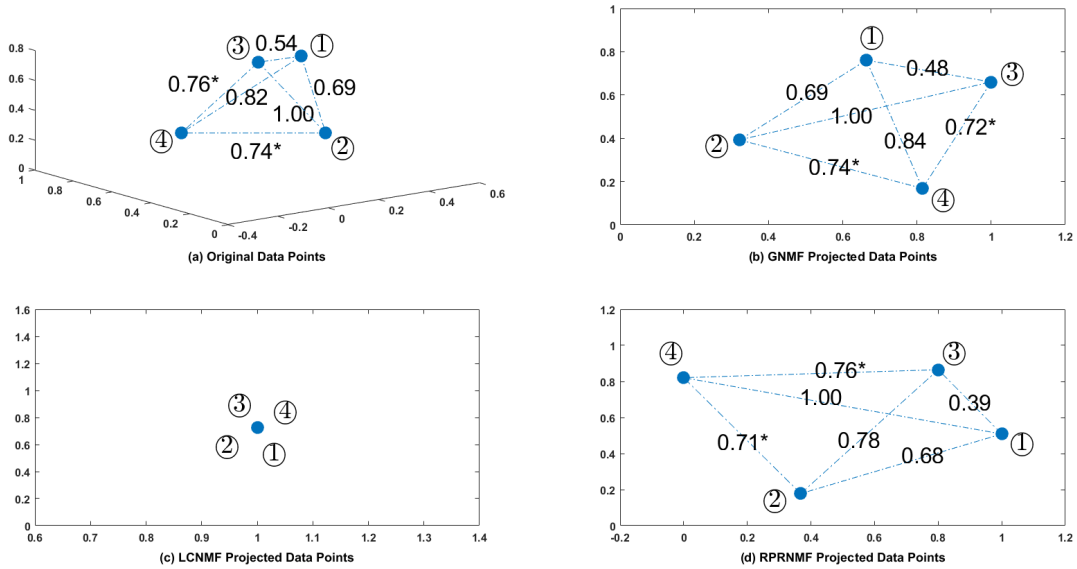


Figure 2.2 : An example of projecting four 3D data points into 2D space. Points are ordered by circled numbers, and the value near each dotted line is the normalised Euclidean distance. (a) The original data space. The distance between point ② and point ④ is less than the distance between point ③ and point ④ (marked by asterisk sign). (b) Projected data points by GNMF using Euclidean distance between each pair of original data points as dissimilarity matrix. The distance between point ② and point ④ becomes bigger than the distance between point ③ and point ④. (c) LCNMF projects all four points onto one when considering all RPRs. (d) RPR-NMF retains all the RPRs after factorisation.

2.2, the RPR among points ②, ③ and ④ was opposite to that when they were in the original 3D space.

LCNMF utilises the relationship among factorised vectors in a different way. Instead of imposing regularised constraints, it represents the relationship by altering the factorising structure. Thus it is not a CNMF method. LCNMF uses partial label information as hard constraints and turns the original NMF task into a semi-supervised problem: they represent the right factorised matrix by a product of a

class matrix and a reduced feature matrix where the class matrix contains binary entries to divide data into a predefined number of classes. Its objective function with Euclidean measure is as:

$$\mathcal{F} = \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{Y}\boldsymbol{B}\|_F^2, \tag{2.8}$$

where $\boldsymbol{Y}$ is the reduced feature matrix and $\boldsymbol{B}$ is the class matrix. The method however, assumes that if two data points have the same label, their corresponding feature vectors must be identical, as showed in Figure 2.2 where all four data points are labelled the same and are projected onto one point after factorisation. Such constraints are too restrictive under many general settings. For example, if two movies are by the same director, in the same genre or featured by the same actors, setting their features identical is to ignore any other differences between them.

RPR-NMF of this thesis in Chapter 3 considers the relationships among feature vectors by imposing relative pairwise relationship regularisations. It neither requires explicit weights between pairwise features, nor forces features with the same labels to be identical. The only needed relative relationships are much more easier to obtain in real-world applications than the weights and labels.

2.1.3 Bounding Constraints in Low-rank MF

Apart from the non-negative constraints, there are many other forms of constraints used in low-rank MF for different application scenarios, for example, the bounding constraints in recommender systems.

In a recommender system, MF algorithms are mostly used in the latent factor models, where the features of users/items are represented by the factorised matrices [2, 32, 33, 34, 35]. Recommendations are based on the predictions of user ratings on items by using the inner product of their corresponding features. As shown in Figure 1.2, without any bounding constraints, vanilla low-rank MF may make

out-of-bounds predictions.

As for studies on the bounds of product matrix, Bounded Matrix Factorisation (BMF) proposed by [25] was the first one focusing on limitations of ranges over prediction values in recommender systems. The key thought of BMF is the following updating formula:

$$\boldsymbol{W}_{ab} = \begin{cases} \max(\mathbf{l}), & if \quad \boldsymbol{W}^*_{ab} < \max(\mathbf{l}) \\ \min(\mathbf{u}), & if \quad \boldsymbol{W}^*_{ab} > \min(\mathbf{u}) \\ \boldsymbol{W}^*_{ab}, & otherwise, \end{cases} \tag{2.9}$$

where $\mathbf{l}$ is the lower bound vector, $\mathbf{u}$ is the upper bound vector, $\boldsymbol{W}_{ab}$ is the element of the left factorised matrix in question and $\boldsymbol{W}^*_{ab}$ is the optimal solution derived from the objective function by letting the derivatives vanish. Under the above updating rules, BMF is proved to achieve better performance than unbounded methods. However, BMF has been proven not to converge to stationary points [26]. Apart from the convergence, its core idea is similar to a truncated approach: instead of directly truncating entries to the predefined bounds, BMF moves the updating values to the inner bounds (the minimum of the upper bound vector and the maximum of the lower bound vector) if they are outside of the given range. As for the time complexity, BMF has to compute for each entry a complementary matrix whose size is close to the observation matrix, which can be time consuming when big datasets are involved.

Recently, an improved bounded matrix completion method was proposed in [26] based on Alternating Direction of Multiplier Method (ADMM) framework (so is called BMC-ADMM). The optimisation problem in their work is defined as:

$$\begin{aligned} &\min_{\boldsymbol{X},\boldsymbol{P},\boldsymbol{T}} \frac{1}{2}\|\boldsymbol{Z} * (\boldsymbol{V} - \boldsymbol{X})\|_F^2 + \lambda\|\boldsymbol{P}\|_* \\ &s.t. \boldsymbol{X} + \boldsymbol{E} = \boldsymbol{P}, \neg\boldsymbol{Z} * \boldsymbol{X} = \mathbf{0}, \boldsymbol{Z} * \boldsymbol{E} = 0, \\ &\boldsymbol{P} = \boldsymbol{T}, r_{\min} \leqslant \boldsymbol{T} \leqslant r_{\max}, \end{aligned} \tag{2.10}$$

where $\boldsymbol{Z}$ is a mask matrix, $*$ denotes the element-wise multiplication, $\|\cdot\|_*$ is the nuclear norm commonly used to approximate the matrix rank, and $\neg$ denotes logical NOT. The authors aimed to improve the theoretical foundation in BMF and reformulated the model so that it can take advantage of the ADMM framework (split the loss, regularisation and constraints) which guarantees their algorithm to converge to the optimal solution. However, the paper does not differ fundamentally from BMF as far as bounding the predicted entries is concerned: during each iteration, BMF-ADMM checks the product of the two factorised matrices and projects any entries which are outside of the range to the boundaries. This checking mechanism is also the most time consuming step in the model which was reported in their work. Another potential obstacle for BMC-ADMM to be applied on very large datasets is the fact that, the updating procedure involves the calculation of partial SVD for a non-sparse matrix having the same size of the observation matrix. Although the authors used several tricks to accelerate each of the updating steps, its efficiency still falls behind the classic methods and the proposed MBMF.

An alternative way to bound the entries is the so-called, Bounded-SVD [27]. In their work, the authors imposed a penalty regularisation term to penalise the updating values which tend to fall outside of the given range:

$$\begin{aligned} F = \sum_{ij} (\boldsymbol{V}_{ij} - \boldsymbol{W}_{i:}\boldsymbol{H}_{:j}) + \\ \lambda(e^{\alpha(\boldsymbol{W}_{i:}\boldsymbol{H}_{:j} - r_{\max})} + e^{\alpha(r_{\min} - \boldsymbol{W}_{i:}\boldsymbol{H}_{:j})}), \end{aligned} \tag{2.11}$$

where λ and α are parameters controlling the strength of penalties, $r_{\max}$ and $r_{\min}$ are the given upper and lower bounds. It also introduces another parameter η as the learning rate when updating the factor matrices. This regularisation method is complex, and cannot guarantee all product values are bounded within the given range simultaneously. Therefore, the objective function only considers a small set of entries instead of bounding all of them. Besides, from the experiments, Bounded-

SVD is very sensitive to its penalty coefficients: when they are set to be large (such as 10), all product values become very small, while when they are set to have small values (such as 0.01), most of the entries cannot be bounded within the predefined range. This algorithm also faces time complexity issues when applied on large datasets.

Introduced in Chapter 5, MBMF is different from all the above existing bounding algorithms. It neither shifts the product values to fall between the boundaries, nor imposes any penalties in the objective function. MBMF introduces the magnitude constraints which guarantee the product of factorised matrices to be within ranges determined by these magnitude constraints. With the help of coordinates conversions, it manages to model the problem with an unconstrained optimisation task which can be solved efficiently by the well-known SGD method.

2.2 Constrained Low-rank Tensor Factorisation

Low-rank TF is usually considered as the extension of low-rank MF for higher dimensional data. However, low-rank TF involves more complicated operations. There are some useful operators when dealing with TF.

Definition 2.1 (Mode, Fibre and Slice)**.**

- *Mode - The number of dimensions of a tensor, also known as way or order.*
- *Fibre - A fibre is defined by fixing every index but one. A mode-n fibre is a vector fixing every index but the dimension n.*
- *Slice - A slice is defined by fixing every index but two.*

Definition 2.2 (Kronecker Product)**.** *The Kronecker Product of two matrices* $\boldsymbol{A} \in$

$\mathcal{R}^{n_1 \times m_1}$ *and* $\boldsymbol{B} \in \mathcal{R}^{n_2 \times m_2}$ *is defined as*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B}, & \cdots, & a_{1m}\boldsymbol{B} \\ \vdots, & \ddots, & \vdots, \\ a_{n1}\boldsymbol{B}, & \cdots, & a_{nm}\boldsymbol{B} \end{bmatrix} \tag{2.12}$$

and $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathcal{R}^{n_1 n_2 \times m_1 m_2}$.

Definition 2.3 (Khatri-Rao Product)**.** *The Khatri-Rao product of two matrices* $\boldsymbol{A} \in \mathcal{R}^{n_1 \times m}$ *and* $\boldsymbol{B} \in \mathcal{R}^{n_2 \times m}$ *is defined as*

$$\boldsymbol{A} \odot \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{b}_1, & \cdots, & a_{1m}\boldsymbol{b}_m \\ \vdots, & \ddots, & \vdots, \\ a_{n1}\boldsymbol{b}_1, & \cdots, & a_{nm}\boldsymbol{b}_m \end{bmatrix} \tag{2.13}$$

and $\boldsymbol{A} \odot \boldsymbol{B} \in \mathcal{R}^{n_1 n_2 \times m}$. *Khatri-Rao product is also called column-wise Kronecker product.*

Definition 2.4 (Hadamard Product)**.** *The Hadamard product of two matrices* $\boldsymbol{A} \in \mathcal{R}^{n \times m}$ *and* $\boldsymbol{B} \in \mathcal{R}^{n \times m}$ *is defined as*

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{11}b_{11}, & \cdots, & a_{1m}b_{1m} \\ \vdots, & \ddots, & \vdots, \\ a_{n1}b_{1n}, & \cdots, & a_{nm}b_{nm} \end{bmatrix} \tag{2.14}$$

and $\boldsymbol{A} * \boldsymbol{B} \in \mathcal{R}^{n \times m}$. *Hadamard product is also called element-wise matrix product.*

Definition 2.5 (The n-Mode Product)**.** *The n-mode product of a tensor* $\boldsymbol{X} \in \mathcal{R}^{I_1 \times \cdots \times I_N}$ *and a matrix* $\boldsymbol{U} \in \mathcal{R}^{J \times I_n}$ *is denoted by* $\boldsymbol{X} \times_n \boldsymbol{U}$ *and is of size* $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ *and its element-wise form is as follows:*

$$(\boldsymbol{X} \times_n \boldsymbol{U})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} u_{j i_n}. \tag{2.15}$$

Definition 2.6 (The Mode-n Matricisation)**.** *The mode-n matricisation of a tensor* $\boldsymbol{X} \in \mathcal{R}^{I_1 \times \cdots \times I_N}$ *is denoted by* $\boldsymbol{X}_{(n)}$ *and arranges the mode-n fibres to be the columns*

of the resulting matrix. The formula used to map tensor element $(i_1, i_2, \ldots, i_N)$ *to matrix element* (i_n, j) *is*

$$j = 1 + \sum_{k=1, k\neq n}^{N} (i_k - 1) J_k \quad with \quad J_k = \prod_{m=1, m\neq n}^{k-1} I_m. \tag{2.16}$$

Two TF schemes are studied in literature: CP decomposition [36] and Tucker decomposition [37].

Definition 2.7 (CP Tensor Decomposition)**.** *Given a tensor* $\boldsymbol{\mathcal{X}} \in \mathcal{R}^{I_1 \times \cdots \times I_N}$*, CP tensor decomposition decomposes* $\boldsymbol{\mathcal{X}}$ *into* K *components where each component is a rank one tensor:*

$$\boldsymbol{\mathcal{X}} \approx \sum_{j=1}^{K} \bigotimes_{l=1}^{N} \boldsymbol{u}_j^{(l)} \tag{2.17}$$

where $\boldsymbol{u}_j^{(l)} \in \mathcal{R}^{I_l}$ *and* $\boldsymbol{u}_j^{(l)} = [u_{1j}^{(l)}, u_{2j}^{(l)}, \ldots, u_{I_l j}^{(l)}]$.

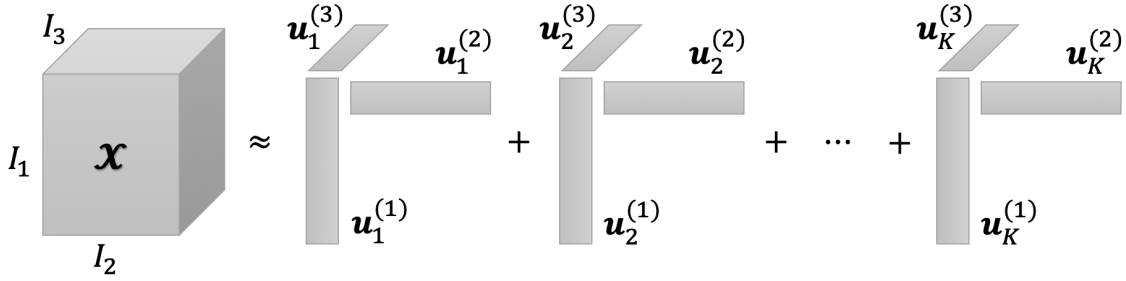


Figure 2.3 : Illustration of CP tensor factorisation of a three-way tensor.

Definition 2.8 (Tucker Tensor Decomposition)**.** *Given a tensor* $\boldsymbol{\mathcal{X}} \in \mathcal{R}^{I_1 \times \cdots \times I_N}$*, Tucker tensor decomposition decomposes it into a core tensor* $\boldsymbol{\mathcal{G}} \in \mathcal{R}^{J_1 \times \cdots \times J_N}$ *along with* N *factor matrices:*

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_N \boldsymbol{U}^{(N)}. \tag{2.18}$$

where $\boldsymbol{U}^{(n)} \in \mathcal{R}^{I_n J_n}$.

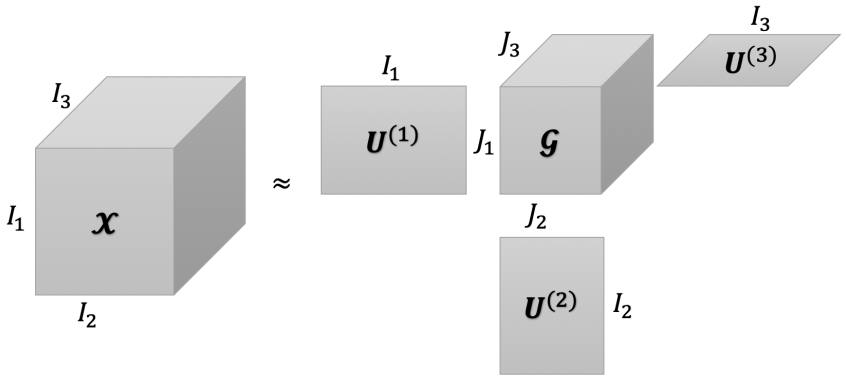


Figure 2.4 : Illustration of Tucker tensor factorisation of a three-way tensor.

Although CP decomposition is actually a special case of Tucker decomposition where the core tensor after factorisation is set to an identity one, researchers usually study them separately because of their different factorisation mechanism, representation format and computation complexity.

2.2.1 Non-negative Tensor Factorisation

NTF, in the same way as NMF, is a constrained and often non-convex optimisation problem. The goal of NTF algorithms is to find factor matrices (and the core tensor in Tucker decomposition) which can restore or approximate the given tensor as close as possible under non-negativity constraints.

Most NTF algorithms emphasised only on one scheme [38, 39, 40], and a few studied both at once [41, 42]. Meanwhile, some only studied three valance tensor cases [43, 44], while others were applied on arbitrary valance tensors [45, 46].

NTF is non-convex generally, however, it is convex when fixing all variable factors but one. Therefore, almost all solutions of NTF grasp this property and update each variable alternatively no matter what kind of algorithms they utilise. Alternating gradient descent [47] and alternating least squares [48, 49] are two widely used solvers for it. Besides, many NTF algorithms conform to the so-called MUL framework

which extends from that for NMF [50]. Furthermore, with additional constraints or regularisations, NTF can cater for more complicated applications, such as in [51], the authors utilised a CP decomposition model with Tikhonov regularisations on NTF to study the Quality-of-Service (QoS) problem.

NTF algorithms have been applied on many real-world applications, such as medical analysis [52, 53], natural language processing [54], speech and music studies [55], image representations [56, 57] and recommender systems [42, 58].

2.2.2 Relationship Regularisations in NTF

Similar to those in NMF, the relationships among feature vectors can be found in the factor matrices or core tensor in NTF. Existing studies on this topic are mainly based on the above mentioned GNMF.

A direct extension of GNMF is the Laplacian Regularised Non-negative Tensor Factorisation (LRNTF) [24] which focuses on CP decomposition. Meanwhile, graph regularisation has been used for spatiotemporal tensor data analysis [59], core tensor [60] or factor matrices [61] in non-negative Tucker decomposition, and EEG analysis [62]. Instead of using a weight matrix, [63] used locally linear embedding to preserve neighbourhood in non-negative CP decomposition for image representation.

In Chapter 4, a different constrained NTF algorithms is introduced to deal with the relationship regularisations. Unlike graph regularised algorithms, it does not require explicit pairwise weights.

Chapter 3

Relative Pairwise Relationship Constrained Non-negative Matrix Factorisation

In this chapter,a new factorisation algorithm is introduced when RPRs are in place, called RPR-NMF. Note that both GNMF and LCNMF only impose constraints on the right factorised matrix because they were proposed as data dimensionality reduction methods. For generality, RPR-NMF imposes constraints on both factorised matrices, and it is trivial to only impose constraints on one factorised matrix. The content of this Chapter is from a published IEEE journal paper [64] and the copyright/credit notice is placed in the reference.

3.1 RPRs and Objective Functions

Consider a dataset represented by a non-negative $N \times M$ matrix $\boldsymbol{V}$. This matrix is then approximately factorised into an $N \times K$ matrix $\boldsymbol{W}$ and a $K \times M$ matrix $\boldsymbol{H}$, where K is usually set to be smaller than both N and M, and is commonly referred to as the latent dimension. The RPR constraints placed on the factorised matrices can be defined as two sets of integer indexed triples:

$$\boldsymbol{L_W} \subseteq \{(q,r,s)|q,r,s \in \mathbb{N}^+, q,r,s \leqslant N, q \neq r \neq s\}, \tag{3.1}$$

$$\boldsymbol{L_H} \subseteq \{(q,r,s)|q,r,s \in \mathbb{N}^+, q,r,s \leqslant M, q \neq r \neq s\}. \tag{3.2}$$

Specifically, each triple represents the relative relationship between two pairs of vectors with one sharing vector. In the thesis, if $\boldsymbol{W}$ was the matrix in question and l^{th} triple specified the distance between vector q and r to be less than the distance between vectors q and s, the relationship could be denoted as $dis(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}) <$

$dis(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:})$ where $\boldsymbol{W}_{q:}$ is the q^{th} row vector of matrix $\boldsymbol{W}$, and $dis(\boldsymbol{x}, \boldsymbol{y})$ measures the distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. The squared Euclidean distance Eq. (2.2) and the Divergence Eq. (2.3) are the two most commonly used measures. Since the Divergence of two vectors is asymmetric ($D(\boldsymbol{x}||\boldsymbol{y}) \neq D(\boldsymbol{y}||\boldsymbol{x})$), when characterising the imposed RPRs, the Symmetric Divergence is used:

$$\begin{aligned} SD(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}(D(\boldsymbol{x}||\boldsymbol{y}) + D(\boldsymbol{y}||\boldsymbol{x})) \\ &= \frac{1}{2}\sum_{i=1}^{K}(\boldsymbol{x}_i - \boldsymbol{y}_i)\log\frac{\boldsymbol{x}_i}{\boldsymbol{y}_i}. \end{aligned} \tag{3.3}$$

Then the constraints are incorporated as regularisation terms in the objective function. In the thesis, the penalty format for Euclidean measure is of an addition of natural exponential functions, while that for Divergence measure is of a hinge loss function. The reason for not using the same format of penalties is that the Divergence measure with exponential penalties cannot guarantee a high proportion of satisfied constrains after factorisation, and that the Euclidean measure with hinge loss penalties cannot steadily converge. Thus with two independent coefficients $\lambda_{\boldsymbol{W}}$ and $\lambda_{\boldsymbol{H}}$, the objective function using Euclidean distance is defined as

$$\begin{aligned} \mathcal{F}_1 &= \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \\ &+ \lambda_{\boldsymbol{W}}\sum_{l=1}^{l_{\boldsymbol{W}}}[\exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:})) + \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))] \\ &+ \lambda_{\boldsymbol{H}}\sum_{l=1}^{l_{\boldsymbol{H}}}[\exp(E(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:r^l})) + \exp(-E(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:s^l}))] \\ &s.t. \lambda_{\boldsymbol{W}} \geqslant 0, \lambda_{\boldsymbol{H}} \geqslant 0, \forall i, j, \boldsymbol{W}_{ij} \geqslant 0, \boldsymbol{H}_{ij} \geqslant 0, \end{aligned} \tag{3.4}$$

and the objective function using Divergence as

$$\begin{aligned}\mathcal{F}_2 &= D(\boldsymbol{V}||\boldsymbol{W}\boldsymbol{H}) \\ &+ \lambda_{\boldsymbol{W}} \sum_{l=1}^{l_{\boldsymbol{W}}} \max(0, SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}) - SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:})) \\ &+ \lambda_{\boldsymbol{H}} \sum_{l=1}^{l_{\boldsymbol{H}}} \max(0, SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:r^l}) - SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:s^l})) \\ &s.t. \lambda_{\boldsymbol{W}} \geqslant 0, \lambda_{\boldsymbol{H}} \geqslant 0, \forall i, j, \boldsymbol{W}_{ij} \geqslant 0, \boldsymbol{H}_{ij} \geqslant 0,\end{aligned} \tag{3.5}$$

where $l_{\boldsymbol{W}}$ and $l_{\boldsymbol{H}}$ are the numbers of constraints.

To solve the above objective functions, one needs to derive the update rules for $\boldsymbol{W}$ and $\boldsymbol{H}$. As a matter of fact, the penalties are not convex even when fixing one of the matrix factors. However, it is found feasible to obtain the update rules by constructing and solving the corresponding Lagrange functions. Once the updating rules are obtained, the objective functions can be minimised by iteratively updating $\boldsymbol{W}$ and $\boldsymbol{H}$.

3.2 Updating rules for Euclidean measure

As for the objective function in Eq. (3.4), first construct a Lagrange function with non-negative constraints $\boldsymbol{W}_{ij} \geqslant 0$ and $\boldsymbol{H}_{ij} \geqslant 0$:

$$\begin{aligned}\mathcal{L}_1 &= \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \sum_{ij} \alpha_{ij}\boldsymbol{W}_{ij} + \sum_{ij} \beta_{ij}\boldsymbol{H}_{ij} \\ &+ \lambda_{\boldsymbol{W}} \sum_{l=1}^{l_{\boldsymbol{W}}} [\exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:})) + \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))] \\ &+ \lambda_{\boldsymbol{H}} \sum_{l=1}^{l_{\boldsymbol{H}}} [\exp(E(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:r^l})) + \exp(-E(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:s^l}))].\end{aligned} \tag{3.6}$$

The partial derivative with respect to $\boldsymbol{W}_{ab}$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{W}_{ab}} = 2[&-(\boldsymbol{V}\boldsymbol{H}^T)_{ab} + (\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T)_{ab} \\ &+ \lambda_{\boldsymbol{W}} C_{row}(\boldsymbol{W}_{ab})] + \alpha_{ab},\end{aligned} \tag{3.7}$$

where

$$
\begin{aligned}
C_{row}(\boldsymbol{W}_{ab}) = \sum_{l=1}^{l_{\boldsymbol{W}}} \Big\{ & \\
& \exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}))[\sum_{q^l=a}(\boldsymbol{W}_{q^lb} - \boldsymbol{W}_{r^lb}) + \sum_{r^l=a}(\boldsymbol{W}_{r^lb} - \boldsymbol{W}_{q^lb})] - \\
& \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))[\sum_{q^l=a}(\boldsymbol{W}_{q^lb} - \boldsymbol{W}_{s^lb}) + \sum_{s^l=a}(\boldsymbol{W}_{s^lb} - \boldsymbol{W}_{q^lb})] \Big\} \\
= \sum_{l=1}^{l_{\boldsymbol{W}}} \Big(& \exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}))(\sum_{q^l=a} \boldsymbol{W}_{q^lb} + \sum_{r^l=a} \boldsymbol{W}_{r^lb}) + \\
& \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))(\sum_{q^l=a} \boldsymbol{W}_{s^lb} + \sum_{s^l=a} \boldsymbol{W}_{q^lb}) \Big) - \\
\sum_{l=1}^{l_{\boldsymbol{W}}} \Big(& \exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}))(\sum_{q^l=a} \boldsymbol{W}_{r^lb} + \sum_{r^l=a} \boldsymbol{W}_{q^lb}) + \\
& \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))(\sum_{q^l=a} \boldsymbol{W}_{q^lb} + \sum_{s^l=a} \boldsymbol{W}_{s^lb}) \Big) \\
= C^+_{row}(\boldsymbol{W}_{ab}) & - C^-_{row}(\boldsymbol{W}_{ab}),
\end{aligned} \tag{3.8}
$$

Let the partial derivative vanish and considering the non-negative constraints ($\alpha_{ab}\boldsymbol{W}_{ab} = 0$ under K.K.T conditions),

$$
\begin{aligned}
((\boldsymbol{V}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^-_{row}(\boldsymbol{W}_{ab}))\boldsymbol{W}_{ab} - \\
((\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^+_{row}(\boldsymbol{W}_{ab}))\boldsymbol{W}_{ab} = 0,
\end{aligned} \tag{3.9}
$$

thus the update rule for $\boldsymbol{W}_{ab}$ is formulated as following:

$$
\boldsymbol{W}_{ab} \leftarrow \boldsymbol{W}_{ab} \frac{(\boldsymbol{V}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^-_{row}(\boldsymbol{W}_{ab})}{(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^+_{row}(\boldsymbol{W}_{ab})}. \tag{3.10}
$$

Similarly, the update rule for $\boldsymbol{H}_{ab}$ is

$$
\boldsymbol{H}_{ab} \leftarrow \boldsymbol{H}_{ab} \frac{(\boldsymbol{W}^T\boldsymbol{V})_{ab} + \lambda_{\boldsymbol{H}} C^-_{col}(\boldsymbol{H}_{ab})}{(\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})_{ab} + \lambda_{\boldsymbol{H}} C^+_{col}(\boldsymbol{H}_{ab})}. \tag{3.11}
$$

As for the update rules, there is a theorem as following:

Theorem 3.1. *The objective function $\mathcal{F}_1$ in Eq. (3.4) is non-increasing under the update rules in Eq. (3.10) and Eq. (3.11) with appropriate penalty coefficients $\lambda_{\boldsymbol{W}}$ and $\lambda_{\boldsymbol{H}}$.*

The prove of the convergence for the above updating rules and Theorem 3.1 is provided in Section 3.4.

3.3 Updating rules for Divergence measure

As for the objective function in Eq. (3.5), first construct a Lagrange function with non-negative constraints $\boldsymbol{W}_{ij} \geqslant 0$ and $\boldsymbol{H}_{ij} \geqslant 0$:

$$\begin{aligned}\mathcal{L}_2 = & D(\boldsymbol{V}||\boldsymbol{W}\boldsymbol{H}) + \sum_{ij}\alpha_{ij}\boldsymbol{W}_{ij} + \sum_{ij}\beta_{ij}\boldsymbol{H}_{ij} \\ & + \lambda_{\boldsymbol{W}}\sum_{l=1}^{l_{\boldsymbol{W}}}\max(0, SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}) - SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:})) \\ & + \lambda_{\boldsymbol{H}}\sum_{l=1}^{l_{\boldsymbol{H}}}\max(0, SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:r^l}) - SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:s^l})).\end{aligned} \tag{3.12}$$

The partial derivative with respect to $\boldsymbol{W}_{ab}$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{W}_{ab}} = & \sum_j (\boldsymbol{H}_{bj} - \frac{\boldsymbol{V}_{aj}\boldsymbol{H}_{bj}}{(\boldsymbol{W}\boldsymbol{H})_{aj}}) \\ & + \frac{1}{2}\lambda_{\boldsymbol{W}} P_{row}(\boldsymbol{W}_{ab}) + \alpha_{ab},\end{aligned} \tag{3.13}$$

where

$$\begin{aligned}P_{row}(\boldsymbol{W}_{ab}) = & \sum_{l=1}^{l_{\boldsymbol{W}}}\Big\{\sum_{q^l=a}[g(\boldsymbol{W}^+_{q^lb}, \boldsymbol{W}^+_{r^lb}) + g(\boldsymbol{W}^+_{q^lb}, \boldsymbol{W}^+_{s^lb})] \\ & + \sum_{r^l=a} g(\boldsymbol{W}^+_{r^lb}, \boldsymbol{W}^+_{q^lb}) - \sum_{s^l=a} g(\boldsymbol{W}^+_{s^lb}, \boldsymbol{W}^+_{q^lb})\Big\},\end{aligned} \tag{3.14}$$

$$\boldsymbol{W}^+_{q^lb} = \begin{cases} 0, & \text{if the constraint } l \text{ is satisfied} \\ \boldsymbol{W}_{q^lb} & \text{otherwise} \end{cases} \tag{3.15}$$

$$g(x, y) = log\frac{x}{y} + (x - y)\frac{1}{x}. \tag{3.16}$$

Let the partial derivative equal to zero as well as considering the non-negative constraints ($\alpha_{ab}\boldsymbol{W}_{ab} = 0$ under K.K.T conditions),

$$-\boldsymbol{W}_{ab}\sum_j \frac{\boldsymbol{V}_{aj}\boldsymbol{H}_{bj}}{(\boldsymbol{W}\boldsymbol{H})_{aj}} + \boldsymbol{W}_{ab}(\frac{1}{2}\lambda_{\boldsymbol{W}}P_{row}(\boldsymbol{W}_{ab}) + \sum_j \boldsymbol{H}_{bj}) = 0, \tag{3.17}$$

thus the update rule for $\boldsymbol{W}_{ab}$ is formulated as following:

$$\boldsymbol{W}_{ab} \leftarrow \boldsymbol{W}_{ab}\frac{\sum_j \boldsymbol{V}_{aj}\boldsymbol{H}_{bj}/(\boldsymbol{W}\boldsymbol{H})_{aj}}{\frac{1}{2}\lambda_{\boldsymbol{W}}P_{row}(\boldsymbol{W}_{ab}) + \sum_j \boldsymbol{H}_{bj}}. \tag{3.18}$$

Similarly, the update rule for $\boldsymbol{H}_{ab}$ is

$$\boldsymbol{H}_{ab} \leftarrow \boldsymbol{H}_{ab}\frac{\sum_i \boldsymbol{V}_{ib}\boldsymbol{W}_{ia}/(\boldsymbol{W}\boldsymbol{H})_{ib}}{\frac{1}{2}\lambda_{\boldsymbol{H}}P_{col}(\boldsymbol{H}_{ab}) + \sum_i \boldsymbol{W}_{ia}}. \tag{3.19}$$

Notice that the value of $P_{row}(\boldsymbol{W}_{ab})$ and $P_{col}(\boldsymbol{H}_{ab})$ may be negative during updating. Thus if the denominator in an iteration is less than zero, then the penalty parts will be abandoned. Besides, the penalty coefficients is dynamically changed to ensure the convergence since the update rules are obtained by approximation (see proof of convergence in Section 3.4).

As for the update rules, there is a theorem as following:

Theorem 3.2. *The objective function $\mathcal{F}_2$ in Eq. (3.5) is non-increasing under the update rules in Eq. (3.18) and Eq. (3.19) with appropriate penalty coefficients $\lambda_{\boldsymbol{W}}$ and $\lambda_{\boldsymbol{H}}$.*

The prove of the convergence for the above updating rules and Theorem 3.2 is provided in Section 3.4.

3.4 Proofs of Convergence and Theorems

An auxiliary function $\mathcal{G}(x, x')$ is constructed to help prove Theorem 3.1 & 3.2, which satisfies the conditions $\mathcal{G}(x, x') \geqslant \mathcal{F}(x)$ and $\mathcal{G}(x, x) = \mathcal{F}(x)$, and guarantees $\mathcal{F}(x)$ to be non-increasing under the following update:

$$x^{t+1} = \arg\min_x \mathcal{G}(x, x'). \tag{3.20}$$

The proofs for $\boldsymbol{W}_{ab}$ are illustrated here and the those for $\boldsymbol{H}_{ab}$ can be derived in a similar fashion. Let $\mathcal{F}_{ab}(\boldsymbol{W})$ denote part of $\mathcal{F}(\boldsymbol{W})$ concerning $\boldsymbol{W}_{ab}$ and so as $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$.

Convergence of Euclidean updating rules and Theorem 1

As for updating rules in Eq. (3.10) & (3.11),

Lemma 3.1. *Function*

$$\begin{aligned}\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) &= \mathcal{F}_{ab}(\boldsymbol{W}^t) + (\boldsymbol{W}_{ab} - \boldsymbol{W}^t_{ab})\mathcal{F}'_{ab}(\boldsymbol{W}^t) \\ &+(\boldsymbol{W}_{ab} - \boldsymbol{W}^t_{ab})^2 \frac{(\boldsymbol{W}^t\boldsymbol{H}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^+_{row}(\boldsymbol{W}^t_{ab})}{\boldsymbol{W}^t_{ab}}\end{aligned} \tag{3.21}$$

is an auxiliary function for $\mathcal{F}_{ab}(\boldsymbol{W})$.

Proof. $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}) = \mathcal{F}_{ab}(\boldsymbol{W})$ is obvious. Applying Taylor Expansion to $\mathcal{F}_{ab}(\boldsymbol{W})$ on the point $\boldsymbol{W}^t_{ab}$,

$$\begin{aligned}\mathcal{F}_{ab}(\boldsymbol{W}) \approx \mathcal{F}_{ab}(\boldsymbol{W}^t) &+ (\boldsymbol{W}_{ab} - \boldsymbol{W}^t_{ab})\mathcal{F}'_{ab}(\boldsymbol{W}^t) \\ &+ \frac{1}{2}(\boldsymbol{W}_{ab} - \boldsymbol{W}^t_{ab})^2 F''_{ab}(\boldsymbol{W}^t).\end{aligned} \tag{3.22}$$

Note that $\mathcal{O}((\boldsymbol{W}^t_{ab})^3)$ is omitted because a linear piecewise function is used to approximate the exponential penalties (see proofs below). Comparing Eq. (3.21) and Eq. (3.22), in order to prove $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) \geqslant \mathcal{F}_{ab}(\boldsymbol{W})$, only need to prove

$$\frac{2[(\boldsymbol{W}^t\boldsymbol{H}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^+_{row}(\boldsymbol{W}^t_{ab})]}{\boldsymbol{W}^t_{ab}} \geqslant F''_{ab}(\boldsymbol{W}^t). \tag{3.23}$$

Rewrite $\mathcal{F}(\boldsymbol{W})$ as an addition of two functions (omitting the penalties for $\boldsymbol{H}$ since it is irrelevant with $\boldsymbol{W}$)

$$\begin{aligned}\mathcal{F}(\boldsymbol{W}) &= \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \\ &+ \lambda_{\boldsymbol{W}} \sum_{l=1}^{l_{\boldsymbol{W}}} [\exp(E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:})) + \exp(-E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}))] \\ &= \mathcal{U}(\boldsymbol{W}) + \lambda_{\boldsymbol{W}} \mathcal{V}(\boldsymbol{W}),\end{aligned} \tag{3.24}$$

Then Eq. (3.23) becomes

$$\frac{\mathcal{U}'_{ab}(\boldsymbol{W}^t)+\lambda_{\boldsymbol{W}}\mathcal{V}'^{+}_{ab}(\boldsymbol{W}^t)}{\boldsymbol{W}^t_{ab}} \geqslant \mathcal{U}''_{ab}(\boldsymbol{W}^t)+\lambda_{\boldsymbol{W}}\mathcal{V}''_{ab}(\boldsymbol{W}^t), \tag{3.25}$$

where $\mathcal{V}'^{+}_{ab}(\boldsymbol{W}^t)$ is the positive part of $\mathcal{V}'_{ab}(\boldsymbol{W}^t)$. Since

$$\begin{aligned}
&\frac{\mathcal{U}'_{ab}(\boldsymbol{W}^t)}{\boldsymbol{W}^t_{ab}}-\mathcal{U}''_{ab}(\boldsymbol{W}^t)\\
=&\frac{(\boldsymbol{W}^t\boldsymbol{H}\boldsymbol{H}^T)_{ab}}{\boldsymbol{W}^t_{ab}}-(\boldsymbol{H}\boldsymbol{H}^T)_{bb}\\
=&\frac{\sum_{k=1}^{K}\boldsymbol{W}^t_{ak}(\boldsymbol{H}\boldsymbol{H}^T)_{kb}}{\boldsymbol{W}^t_{ab}}-(\boldsymbol{H}\boldsymbol{H}^T)_{bb}\\
\geqslant&\frac{\boldsymbol{W}^t_{ab}(\boldsymbol{H}\boldsymbol{H}^T)_{bb}}{\boldsymbol{W}^t_{ab}}-(\boldsymbol{H}\boldsymbol{H}^T)_{bb}\\
\geqslant&\ 0,
\end{aligned} \tag{3.26}$$

the only remaining work is to prove

$$\frac{\mathcal{V}'^{+}_{ab}(\boldsymbol{W}^t)}{\boldsymbol{W}^t_{ab}} \geqslant \mathcal{V}''_{ab}(\boldsymbol{W}^t). \tag{3.27}$$

Recall that the function $\mathcal{V}(\boldsymbol{W})$ is a summation of additions of two natural exponential functions, and the exponents are quadratic functions of $\boldsymbol{W}$. Calculation of their derivatives is intricate and causes difficulties to prove the above inequality. Here gives a proof by applying approximations to the natural exponential functions.

For the general natural exponential function $f(x)=\mathrm{e}^x$ with support $[0,+\infty)$ under partition A, we can use a piecewise function to approximate it

$$\mathrm{e}^x \approx \begin{cases}
\frac{(\mathrm{e}^{A_1}-1)}{A_1}x+1, & x\in[0,A_1)\\
\frac{(\mathrm{e}^{A_2}-\mathrm{e}^{A_1})}{A_2-A_1}x+\frac{A_2\mathrm{e}^{A_1}-A_1\mathrm{e}^{A_2}}{A_2-A_1}, & x\in[A_1,A_2)\\
\vdots & \\
\frac{(\mathrm{e}^{A_n}-\mathrm{e}^{A_{n-1}})}{A_n-A_{n-1}}x+\frac{A_n\mathrm{e}^{A_{n-1}}-A_{n-1}\mathrm{e}^{A_n}}{A_n-A_{n-1}}, & x\in[A_{n-1},A_n)\\
\vdots &
\end{cases}, \tag{3.28}$$

where each sub-function is a linear function of x. The relationship between $f(x)$ and the piecewise function is demonstrated in Figure 3.1. Note that A should have

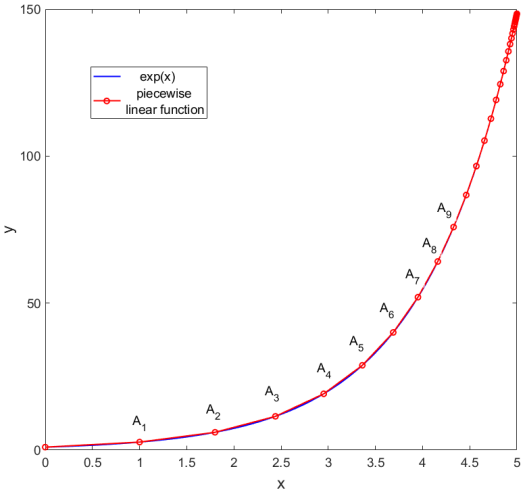


Figure 3.1 : Using a piecewise linear function to approximate e^{x}.

smaller partitions as the curve increasing. In the implementation, the factorising matrix is also normalised to make the exponential penalties as small as possible. Similarly, one can approximate $g(x) = \mathrm{e}^{-x}$ with a piecewise function whose n^{th} sub-function is

$$\frac{(\mathrm{e}^{-A_{n-1}} - \mathrm{e}^{-A_n})}{A_{n-1} - A_n} x + \frac{A_{n-1}\mathrm{e}^{-A_n} - A_n \mathrm{e}^{-A_{n-1}}}{A_{n-1} - A_n}, x \in [A_{n-1}, A_n) \tag{3.29}$$

Now $\mathcal{V}(\boldsymbol{W})$ can be approximated by using the above two piecewise functions. The n^{th} part of it is

$$\begin{aligned} \mathcal{V}(\boldsymbol{W})^{(n)} \approx \sum_{l=1}^{l_{\boldsymbol{W}}} \frac{1}{A_n - A_{n-1}} \Big\{ & (\mathrm{e}^{A_n} - \mathrm{e}^{A_{n-1}}) E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}) \\ & + (\mathrm{e}^{-A_n} - \mathrm{e}^{-A_{n-1}}) E(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}) + A_n (\mathrm{e}^{A_{n-1}} + \mathrm{e}^{-A_{n-1}}) \\ & - A_{n-1} (\mathrm{e}^{A_n} + \mathrm{e}^{-A_n}) \Big\}. \end{aligned} \tag{3.30}$$

Its first and second order derivatives w.r.t $\boldsymbol{W}_{ab}$ are

$$\begin{aligned}\mathcal{V}'_{ab}(\boldsymbol{W})^{(n)} \approx \sum_{l=1}^{l_{\boldsymbol{W}}} \frac{1}{A_n - A_{n-1}} \Big\{ \\ (\mathrm{e}^{A_n} - \mathrm{e}^{A_{n-1}})[\sum_{q^l=a}(\boldsymbol{W}_{q^lb} - \boldsymbol{W}_{r^lb}) + \sum_{r^l=a}(\boldsymbol{W}_{r^lb} - \boldsymbol{W}_{q^lb})] + \\ (\mathrm{e}^{-A_n} - \mathrm{e}^{-A_{n-1}})[\sum_{q^l=a}(\boldsymbol{W}_{q^lb} - \boldsymbol{W}_{s^lb}) + \sum_{s^l=a}(\boldsymbol{W}_{s^lb} - \boldsymbol{W}_{q^lb})] \Big\},\end{aligned} \tag{3.31}$$

$$\begin{aligned}\mathcal{V}''_{ab}(\boldsymbol{W})^{(n)} \approx \sum_{l=1}^{l_{\boldsymbol{W}}} \frac{1}{A_n - A_{n-1}} \Big\{ (\mathrm{e}^{A_n} - \mathrm{e}^{A_{n-1}})(\sum_{q^l=a} 1 + \sum_{r^l=a} 1) \\ + (\mathrm{e}^{-A_n} - \mathrm{e}^{-A_{n-1}})(\sum_{q^l=a} 1 + \sum_{s^l=a} 1) \Big\}.\end{aligned} \tag{3.32}$$

And the positive part of the first derivative is

$$\begin{aligned}\mathcal{V}'^{+}_{ab}(\boldsymbol{W})^{(n)} \approx \sum_{l=1}^{l_{\boldsymbol{W}}} \frac{1}{A_n - A_{n-1}} \Big\{ \\ (\mathrm{e}^{A_n} - \mathrm{e}^{A_{n-1}})(\sum_{q^l=a} \boldsymbol{W}_{q^lb} + \sum_{r^l=a} \boldsymbol{W}_{r^lb}) - \\ (\mathrm{e}^{-A_n} - \mathrm{e}^{-A_{n-1}})(\sum_{q^l=a} \boldsymbol{W}_{s^lb} + \sum_{s^l=a} \boldsymbol{W}_{q^lb}) \Big\}.\end{aligned} \tag{3.33}$$

Then

$$\begin{aligned}\frac{\mathcal{V}'^{+}_{ab}(\boldsymbol{W}^t)^{(n)}}{\boldsymbol{W}^t_{ab}} - \mathcal{V}''_{ab}(\boldsymbol{W}^t)^{(n)} = \sum_{l=1}^{l_{\boldsymbol{W}}} \frac{1}{A_n - A_{n-1}} \Big\{ \\ (\mathrm{e}^{-A_{n-1}} - \mathrm{e}^{-A_n})(\sum_{q^l=a}(\frac{\boldsymbol{W}^t_{s^lb}}{\boldsymbol{W}^t_{q^lb}} + 1) + \sum_{s^l=a}(\frac{\boldsymbol{W}^t_{q^lb}}{\boldsymbol{W}^t_{s^lb}} + 1)) \Big\} \\ \geqslant 0\end{aligned} \tag{3.34}$$

Thus Eq. (3.23) holds. □

Now the convergence of Theorem 3.1 can be proved:

Proof of Theorem 3.1. According to Eq. (3.20) and Eq. (3.21),

$$\begin{aligned}\boldsymbol{W}^{t+1}_{ab} &= \arg\min_{\boldsymbol{W}_{ab}} \mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) \\ &= \boldsymbol{W}^t_{ab} \frac{(\boldsymbol{V}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^-_{row}(\boldsymbol{W}_{ab})}{(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T)_{ab} + \lambda_{\boldsymbol{W}} C^+_{row}(\boldsymbol{W}_{ab})}.\end{aligned} \tag{3.35}$$

As $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$ is an auxiliary function, $\mathcal{F}_{ab}(\boldsymbol{W})$ is non-increasing under this update rule. □

Convergence of Divergence updating rules and Theorem 2

As for updating rules in Eq. (3.18) & (3.19),

Lemma 3.2. *Function*

$$\begin{aligned}\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) = &\sum_j (\boldsymbol{V}_{aj} \log \boldsymbol{V}_{aj} - \boldsymbol{V}_{aj}) + \sum_j (\boldsymbol{W}\boldsymbol{H})_{aj} \\ &- \sum_{jk} \boldsymbol{V}_{aj} \frac{\boldsymbol{W}^t_{ak}\boldsymbol{H}_{kj}}{(\boldsymbol{W}^t\boldsymbol{H})_{aj}} (\log \boldsymbol{W}_{ak}\boldsymbol{H}_{kj} - \log \frac{\boldsymbol{W}^t_{ak}\boldsymbol{H}_{kj}}{(\boldsymbol{W}^t\boldsymbol{H})_{aj}}) \\ &+ \lambda_{\boldsymbol{W}} \sum_{l=1}^{l_{\boldsymbol{W}}} \max(0, SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{r^l:}) - SD(\boldsymbol{W}_{q^l:}, \boldsymbol{W}_{s^l:}) \\ &+ \lambda_{\boldsymbol{H}} \sum_{l=1}^{l_{\boldsymbol{H}}} \max(0, SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:r^l}) - SD(\boldsymbol{H}_{:q^l}, \boldsymbol{H}_{:s^l}).\end{aligned} \tag{3.36}$$

This is an auxiliary function for $\mathcal{F}_{ab}(\boldsymbol{W})$.

Proof. $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}) = \mathcal{F}_{ab}(\boldsymbol{W})$ is obvious. To prove that $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) \geqslant \mathcal{F}_{ab}(\boldsymbol{W})$, utilise the Jensen Inequality

$$-\log \sum_k \boldsymbol{W}_{ak}\boldsymbol{H}_{kj} \leqslant -\sum_k \alpha_k \log \frac{\boldsymbol{W}_{ak}\boldsymbol{H}_{kj}}{\alpha_k} \tag{3.37}$$

which holds for all non-negative α_k that sum to unity. Setting

$$\alpha_k = \frac{\boldsymbol{W}^t_{ak}\boldsymbol{H}_{kj}}{(\boldsymbol{W}^t\boldsymbol{H})_{aj}}, \tag{3.38}$$

then

$$\begin{aligned}-\sum_j \boldsymbol{V}_{aj} \log \sum_k \boldsymbol{W}_{ak}\boldsymbol{H}_{kj} \leqslant \\ -\sum_{jk} \boldsymbol{V}_{aj} \frac{\boldsymbol{W}^t_{ak}\boldsymbol{H}_{kj}}{(\boldsymbol{W}^t\boldsymbol{H})_{aj}} (\log \boldsymbol{W}_{ak}\boldsymbol{H}_{kj} - \log \frac{\boldsymbol{W}^t_{ak}\boldsymbol{H}_{kj}}{(\boldsymbol{W}^t\boldsymbol{H})_{aj}}).\end{aligned} \tag{3.39}$$

This is the only different part between $\mathcal{F}_{ab}(\boldsymbol{W})$ and $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$. Thus, $\mathcal{F}_{ab}(\boldsymbol{W}) \leqslant \mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$ holds. □

Now the convergence of Theorem 3.2 can be proved:

Proof of Theorem 3.2. According to Eq. (3.20) and Eq. (3.36),

$$\begin{aligned} \boldsymbol{W}_{ab}^{t+1} &= \arg\min_{\boldsymbol{W}_{ab}} \mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t) \\ &\approx \boldsymbol{W}_{ab}^t \frac{\sum_j V_{aj} \boldsymbol{H}_{bj} / (\boldsymbol{W}\boldsymbol{H})_{aj}}{\frac{1}{2}\lambda_{\boldsymbol{W}} P_{row}(\boldsymbol{W}_{ab}) + \sum_j \boldsymbol{H}_{bj}}. \end{aligned} \tag{3.40}$$

Notice that $\boldsymbol{W}$ also exists in the penalty regularisation terms, and it is difficult to calculate their corresponding derivatives with respect to $\boldsymbol{W}$. For simplicity and efficiency, $\boldsymbol{W}$ is substituted in penalties with $\boldsymbol{W}^t$ so that they become irrelevant to $\boldsymbol{W}$. This approximation of the local optima of $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$ is numerically proved feasible through the experiments.

As $\mathcal{G}_{ab}(\boldsymbol{W}, \boldsymbol{W}^t)$ is an auxiliary function, $\mathcal{F}_{ab}(\boldsymbol{W})$ is non-increasing under this update rule. □

The complete algorithms of RPR-NMF are demonstrated in Algorithm 3.1 and Algorithm 3.2.

3.5 Experiments

In this section, experiments are conducted on both synthetic datasets and real datasets to demonstrate the performance of the proposed algorithm RPR-NMF. The baseline algorithms chose for comparison are:

- Fast and robust recursive algorithms for separable non-negative matrix factorisation (FSNMF) [14]. It was proposed to solve NMF with separability assumption.

- Alternating Non-negative Least Squared (ANLS) method for non-negative matrix factorisation [65]. This is a generic approach for NMF by solving multiple non-negative least squared tasks when fixing one of the factorised matrices.

Algorithm 3.1 Relative Pairwise Relationship constrained Non-negative Matrix Factorisation (RPR-NMF) using squared Euclidean distance

INPUT: $\boldsymbol{V} \in \mathbb{R}_{\geqslant 0}^{N \times M}$, $K \in \mathbb{N}^{+}$, $\boldsymbol{L_W} \in \mathbb{N}^{l_{\boldsymbol{W}} \times 3}$, $\boldsymbol{L_H} \in \mathbb{N}^{l_{\boldsymbol{H}} \times 3}$,$\lambda_{\boldsymbol{W}} \in \mathbb{R}^{+}$, $\lambda_{\boldsymbol{H}} \in \mathbb{R}^{+}$

OUTPUT: $\boldsymbol{W} \in \mathbb{R}_{\geqslant 0}^{N \times K}$, $\boldsymbol{H} \in \mathbb{R}_{\geqslant 0}^{K \times M}$

Initialise $\boldsymbol{W}$ and $\boldsymbol{H}$ as non-negative random matrices
while Terminal conditions not satisfied **do**
  for $k \leftarrow 1$ **to** K **do**
    for $a \leftarrow 1$ **to** N **do** ▷ Update $\boldsymbol{W}$
      $C_{row}^{+}(\boldsymbol{W}_{ab}), C_{row}^{-}(\boldsymbol{W}_{ab}) \leftarrow$ Eq. (3.8)
      $\boldsymbol{W}_{ak} \leftarrow \boldsymbol{W}_{ak} \frac{(\boldsymbol{VH})_{ak} + \lambda_{\boldsymbol{W}} C_{row}^{-}(\boldsymbol{W}_{ab})}{(\boldsymbol{WHH}^{T})_{ak} + \lambda_{\boldsymbol{W}} C_{row}^{+}(\boldsymbol{W}_{ab})}$
    for $b \leftarrow 1$ **to** M **do** ▷ Update $\boldsymbol{H}$
      $C_{col}^{+}(\boldsymbol{H}_{ab}), C_{col}^{-}(\boldsymbol{H}_{ab}) \leftarrow$ Eq. (3.8)
      $\boldsymbol{H}_{kb} \leftarrow \boldsymbol{H}_{kb} \frac{(\boldsymbol{W}^{T}\boldsymbol{V})_{kb} + \lambda_{\boldsymbol{H}} C_{col}^{-}(\boldsymbol{H}_{ab})}{(\boldsymbol{W}^{T}\boldsymbol{WH})_{kb} + \lambda_{\boldsymbol{H}} C_{col}^{-}(\boldsymbol{H}_{ab})}$

- Non-negative matrix factorisation using multiplicative updating rules (NMF) [1]. It is the basic framework solving NMF by multiplicative rules.
- Graph regularised non-negative matrix factorisation (GNMF) [22]. It imposes similarity constraints as penalty regularisation to maintain the proximity among data.
- Label constrained non-negative matrix factorisation (LCNMF) [23]. This method uses partial labels of data to ensure the corresponding columns in the right factorised matrix to be identical.

FSNMF and ANLS only use Euclidean measure, while each of the other algorithms have two versions: one with Euclidean measure (denoted by suffix "_euc" in figures and tables), the other with Divergence measure (denoted by suffix "_div").

Algorithm 3.2 Relative Pairwise Relationship constrained Non-negative Matrix Factorisation (RPR-NMF) using Divergence measure

INPUT: $\boldsymbol{V} \in \mathbb{R}_{\geqslant 0}^{N \times M}$, $K \in \mathbb{N}$, $\boldsymbol{L_W} \in \mathbb{N}^{l_{\boldsymbol{W}} \times 3}$, $\boldsymbol{L_H} \in \mathbb{N}^{l_{\boldsymbol{H}} \times 3}$, $\lambda_{\boldsymbol{W}} \in \mathbb{R}^{+}, \lambda_{\boldsymbol{H}} \in \mathbb{R}^{+}$

OUTPUT: $\boldsymbol{W} \in \mathbb{R}_{\geqslant 0}^{N \times K}$, $\boldsymbol{H} \in \mathbb{R}_{\geqslant 0}^{K \times M}$

Initialise $\boldsymbol{W}$ and $\boldsymbol{H}$ as non-negative random matrices
while Terminal conditions not satisfied **do**
  for $k \leftarrow 1$ **to** K **do**
    for $a \leftarrow 1$ **to** N **do** ▷ Update $\boldsymbol{W}$
      $P_{row}(\boldsymbol{W}_{ab}) \leftarrow$ Eq. (3.14)
      if $\frac{1}{2}\lambda_{\boldsymbol{W}} P_{row}(\boldsymbol{W}_{ab}) + \sum_j \boldsymbol{H}_{kj} < 0$ **then**
        $\boldsymbol{W}_{ak} \leftarrow \boldsymbol{W}_{ak} \frac{\sum_j \boldsymbol{V}_{aj} \boldsymbol{H}_{kj} / (\boldsymbol{WH})_{aj}}{\sum_j \boldsymbol{H}_{kj}}$
      else
        $\boldsymbol{W}_{ak} \leftarrow \boldsymbol{W}_{ak} \frac{\sum_j \boldsymbol{V}_{aj} \boldsymbol{H}_{kj} / (\boldsymbol{WH})_{aj}}{\frac{1}{2}\lambda_{\boldsymbol{W}} P_{row}(\boldsymbol{W}_{ab}) + \sum_j \boldsymbol{H}_{kj}}$
    for $b \leftarrow 1$ **to** M **do** ▷ Update $\boldsymbol{H}$
      $P_{col}(\boldsymbol{H}_{ab}) \leftarrow$ Eq. (3.14)
      if $\frac{1}{2}\lambda_{\boldsymbol{H}} P_{col}(\boldsymbol{H}_{ab}) + \sum_i \boldsymbol{W}_{ik} < 0$ **then**
        $\boldsymbol{H}_{kb} \leftarrow \boldsymbol{H}_{kb} \frac{\sum_i \boldsymbol{V}_{ib} \boldsymbol{W}_{ik} / (\boldsymbol{WH})_{ib}}{\sum_i \boldsymbol{W}_{ik}}$
      else
        $\boldsymbol{H}_{kb} \leftarrow \boldsymbol{H}_{kb} \frac{\sum_i \boldsymbol{V}_{ib} \boldsymbol{W}_{ik} / (\boldsymbol{WH})_{ib}}{\frac{1}{2}\lambda_{\boldsymbol{H}} P_{col}(\boldsymbol{H}_{ab}) + \sum_i \boldsymbol{W}_{ik}}$
  if Objective function value increases **then**
    $\lambda_{\boldsymbol{W}} \leftarrow 1/2 \cdot \lambda_{\boldsymbol{W}}, \quad \lambda_{\boldsymbol{H}} \leftarrow 1/2 \cdot \lambda_{\boldsymbol{H}}$
    Roll back
  else
    $\lambda_{\boldsymbol{W}} \leftarrow 1.01 \cdot \lambda_{\boldsymbol{W}}, \quad \lambda_{\boldsymbol{H}} \leftarrow 1.01 \cdot \lambda_{\boldsymbol{H}}$

Note that GNMF and LCNMF only have one regularisation term (they are both designed as dimensionality reduction methods), the weight matrix for GNMF only performs as constraints imposed on the right factorised matrix and the label matrix for LCNMF also only affects the right factorised matrix. Thus for fair comparison, RPR-NMF only has RPR constraints on the right factorised matrix in the experiments on synthetic datasets as well as datasets for image clustering. However, when it comes down to recommender systems, constraints are imposed on both factorised matrices which are considered as users' and items' features.

The metrics used to evaluate the performance of algorithms are:

1. *Mean Squared Loss (MSL)* for algorithms using Euclidean measure and *Mean Divergence (MD)* for algorithms using Divergence measure, which evaluate the approximation performance and are defined as

$$\text{MSL} = \frac{1}{NM}\|\boldsymbol{V} - \boldsymbol{W}^*\boldsymbol{H}^*\|_F^2, \tag{3.41}$$

$$\text{MD} = \frac{1}{NM}D(\boldsymbol{V}||\boldsymbol{W}^*\boldsymbol{H}^*), \tag{3.42}$$

where $\boldsymbol{W}^*$ and $\boldsymbol{H}^*$ are the final factorised matrices.

2. *Constraint Satisfied Rate (CSR)*, which presents how well the RPR constraints are satisfied and is defined as

$$\text{CSR} = \frac{1}{2}(l^*_{\boldsymbol{W}}/l_{\boldsymbol{W}} + l^*_{\boldsymbol{H}}/l_{\boldsymbol{H}}), \tag{3.43}$$

where $l^*_{\boldsymbol{W}}$ and $l^*_{\boldsymbol{H}}$ are the numbers of satisfied constraints.

3. *Clustering Accuracy (ACC)*, which is calculated by constructing a cost matrix, solving the matrix by Munkres Assign Algorithm [66], and mapping one clustering to the other; and *Normalised Mutual Information (NMI)* [67], which is to calculate the entropy information correlation of clusterings. They are used to evaluate the performance of the clustering results in image clustering experiments.

4. *Root Mean Squared Error (RMSE)* [68], which evaluates the difference between

recovering ratings and original ratings (the latter are removed before factorisation in cross validation); and *F1 score* [69], which tells how much proportion of correct recommendations the recovered rating matrix gives. They are used to evaluate the performance on recommendation systems and defined as

$$\text{RMSE} = (\sum_{ij} \boldsymbol{M}_{ij})^{-1/2} \|\boldsymbol{M} * (\boldsymbol{V} - \boldsymbol{W}^* \boldsymbol{H}^*)\|_F, \tag{3.44}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3.45}$$

where $\boldsymbol{M}$ is a binary matrix in which ones denote the chosen entries and zeros denote not chosen entries in a cross validation experiment. Each user's average rating is used as a threshold. Ratings greater than the threshold suggest the corresponding items are recommended. Thus two recommendation binary vectors before and after factorisation can then be used to calculate the value of Precision and Recall.

First, two algorithms are introduced to convert the additional information. Then the effectiveness of RPR-NMF is validated and its computing complexity is analysed through two synthetic experiments, followed by experiments on real datasets for applications of image clustering and recommender systems. The statistics of the datasets used are presented in Table 3.1.

Table 3.1 : Statistics of Datasets used for Experiments

dataset	rows	columns	range	density
Synthetic 1	500	500	[0,10]	1
Synthetic 2	50-10,000	50-10,000	[0,10]	1
AT&T ORL	1,024	400	[0,255]	1
CMU PIE	1,024	2,856	[0,255]	1
Movielens 1M	6,040	3,706	[1,5]	0.0447

All the code of RPR-NMF as well as preprocessed datasets can be downloaded from: https://github.com/shawn-jiang/RPRNMF.

3.5.1 Additional Information Conversion

Since GNMF, LCNMF and RPR-NMF all need additional information besides the factorising matrix itself, it is necessary to make it fair for all the three algorithms to equally access available additional information.

Algorithm 3.3 Constructing Weight Matrix for GNMF from Pairwise Relationship Constraints

---

INPUT: $M \in \mathbb{N}^+$ (the column dimension of $\boldsymbol{H}$),

$\boldsymbol{L_H} \in \mathbb{N}^{l_{\boldsymbol{H}} \times 3}$ (pairwise relationship constraints on $\boldsymbol{H}$),

$mins, maxs \in \mathbb{R}$ (minimal and maximal weight to set)

OUTPUT: $\boldsymbol{S_H} \in \mathbb{R}_{\geqslant 0}^{M \times M}$ (weight matrix for GNMF)

$\boldsymbol{S_H} \leftarrow \boldsymbol{I}_M$

Construct a graph G initialised with zero node

for $l \leftarrow 1$ **to** $l_{\boldsymbol{H}}$ **do**

    $n_1 \leftarrow (\boldsymbol{L_H}(l,1), \boldsymbol{L_H}(l,2)), \quad n_2 \leftarrow (\boldsymbol{L_H}(l,1), \boldsymbol{L_H}(l,3))$ ▷ node 1 and 2

    $e_{12} \leftarrow (n_1, n_2)$ ▷ edge from node 1 to 2

    $G \leftarrow \{n_1, n_2, e_{12}\}$

Recursively calculate the depth of each node (nodes with no outgoing edges are set depth 1, their direct parents are set depth 2, and so on).

$t \leftarrow (maxs - mins)/(\max(depth) - 1)$

for each node n **do**

    $i, j \leftarrow$ two points of n

    $\boldsymbol{S_H}(i,j) \leftarrow mins + (depth(n) - 1) * t$

GNMF needs a weight matrix which describes how close the data vectors should

Algorithm 3.4 Constructing Label Matrix for LCNMF from Pairwise Relationship Constraints

INPUT: $M \in \mathbb{N}^+$ (the column dimension of $\boldsymbol{H}$),

$\boldsymbol{L_H} \in \mathbb{N}^{l_{\boldsymbol{H}} \times 3}$ (pairwise relationship constraints on $\boldsymbol{H}$)

OUTPUT: $\boldsymbol{B}$ (binary matrix whose size is at most $M \times M$ but not determined)

$\boldsymbol{B} \leftarrow \boldsymbol{0}_M$
for $l \leftarrow 1$ **to** $l_{\boldsymbol{H}}$ **do**
    $q \leftarrow \boldsymbol{L_H}(l,1) \quad r \leftarrow \boldsymbol{L_H}(l,2)$
    if $\boldsymbol{B}_{:,q} \neq \boldsymbol{0} \wedge \boldsymbol{B}_{:,r} \neq \boldsymbol{0}$ **then**
        $\boldsymbol{B}_{:,q} \leftarrow \boldsymbol{B}_{:,q} \vee \boldsymbol{B}_{:,r}$
        $\boldsymbol{B}_{:,r} \leftarrow \boldsymbol{0}$
    else if $\boldsymbol{B}_{:,q} \neq \boldsymbol{0}$ **then**
        $label(r) \leftarrow label(q)$
    else if $\boldsymbol{B}_{:,r} \neq \boldsymbol{0}$ **then**
        $label(q) \leftarrow label(r)$
    else
        $q, r \leftarrow$ new label
Delete rows containing all zeros from $\boldsymbol{B}$

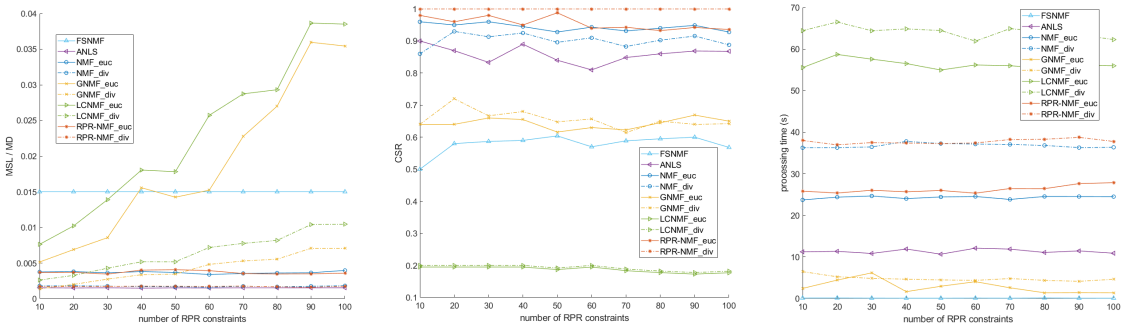
be, LCNMF requires which data vectors should have the same label, while RPR-NMF demands a list of RPR constraints among factorised vectors. These three additional information can be converted from each to the others. However, their intensities are different: labels of LCNMF are the strongest, followed by the similarities of GNMF, and the RPR constraints of RPR-NMF are the weakest. It will cause loss of information if one converts stronger constraints to weaker ones, but not if it is done the other way around. Thus here two algorithms are introduced to convert the RPR constraints for RPR-NMF into the weight matrix for GNMF

(Algorithm 3.3) and the label matrix for LCNMF (Algorithm 3.4) respectively.

3.5.2 Effectiveness Validation & Complexity Analysis

Two experiments have been conducted on synthetic datasets to verify the effectiveness and the computing complexity of RPR-NMF comparing to the baseline algorithms has been studied. These experiments are conducted on a computer with i7-6900K CPU, 16G RAM and Windows 10.

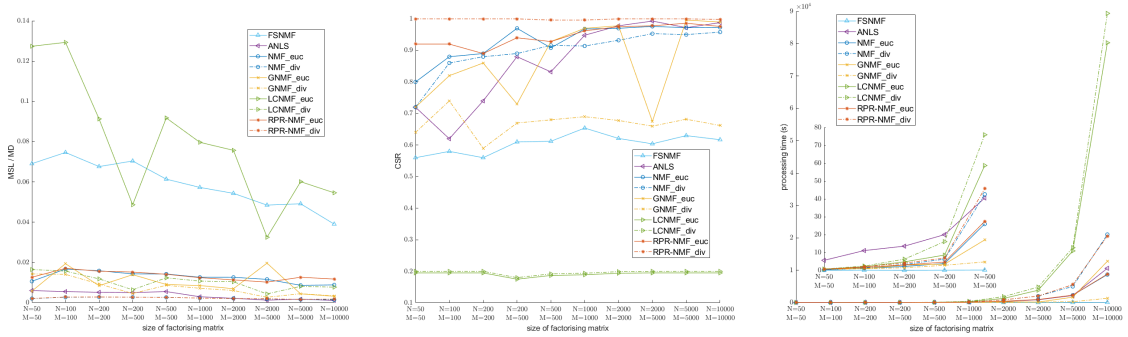
It is worth noting that, the RPR constraints in the experiments are randomly generated but there are chains among them. A chain of constraints is like $dis(\boldsymbol{a}, \boldsymbol{b}) < dis(\boldsymbol{b}, \boldsymbol{c}) < dis(\boldsymbol{c}, \boldsymbol{d})$, which is a 3-chain of constraints. As it is going to show in the results, RPR-NMF can satisfy chain constraints while others are incapable.



(a) (b) (c)

Figure 3.2 : Performance with respect to the number of pairwise relationship constraints. (a) MSL/MD, (b) CSR, (c) Processing time.

In the first experiment, the influence of the number of RPR constraints with respect to MSL/MD, CSR and processing time is explored. First, a 500×20 matrix $\boldsymbol{W}_0$ and a 20×500 matrix $\boldsymbol{H}_0$ are randomly generated, which yields the factorising matrix $\boldsymbol{V}$ by rescaling their product to be within $[0, 10]$. Then 2 to 20 (with step 2) groups of 5-chain constraints (i.e. 10 to 100 constraints) are generated from $\boldsymbol{H}_0$. K is set 20 and $\lambda_{\boldsymbol{H}}$ is set 1. For each experiment with different number of constraints,



(a) (b) (c)

Figure 3.3 : Performance with respect to the size of factorising matrix. (a) MSL/MD, (b) CSR, (c) Processing time.

repeat 5 times and the average results is presented in Figure 3.2.

As the number of constraints increases, (a) ANLS, NMF and RPR-NMF algorithms obtain very close and low errors, while the errors of GNMF and LCNMF are much higher and increasing as well, and the performance of FSNMF is quite stable; (b) RPR-NMF using Divergence measure achieves the highest CSRs (average 100.00%) followed by its Euclidean version (average 95.52%), ANLS and FSNMF obtain 85.89% and 57.82% respectively, while NMF, GNMF, LCNMF obtain 94.35% / 90.23%, 64.28% / 65.58%, 19.23% / 19.23% average CSR for Euclidean and Divergence measures respectively. (c) the processing time of FSNMF is the lowest, while both LCNMF algorithms spend the longest time, and the time used by RPR-NMF is close to that used by NMF with slight increases. This demonstrates that comparing with GNMF and LCNMF, the proposed RPR-NMF not only maintains as low error and low time complexity as NMF, but also satisfies the most expected relationship constraints after factorisation.

The second experiment focused on how the size of factorising matrix affects the performance of different algorithms. The size of the factorising matrix $\boldsymbol{V}$ varies from

50×50 to $10,000 \times 10,000$. Each $\boldsymbol{V}$ is also obtained by the rescaled ($[0, 10]$) product of randomly generated $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$. For each matrix size, the latent dimension K is set 20, and the number of constraints is set $N/10$. The constraints are also of 5-chains. $\lambda_{\boldsymbol{H}}$ is set 1. The average results of 5 repeated experiments are presented in Figure 3.3.

As the size of factorising matrix increases, (a) NMF and RPR-NMF algorithms obtain close and low errors again, while the errors of FSNMF and LCNMF using Euclidean measure are decreasing but still much higher than those of others; (b) RPR-NMF using Divergence measure achieves the highest score on CSR (100.00% on all cases), while FSNMF, ANLS, NMF and GNMF obtain higher CSR; (c) the processing time of all algorithms increases, and FSNMF is the fastest while LCNMF algorithms are the slowest. RPR-NMF spent nearly the same time as unconstrained NMF, despite the imposed RPR constraints. It is noticed that FSNMF is significantly faster than all other algorithms because of its separability assumption. However, this assumption is too strong in many settings and does not generate good factorisation (see algorithm performance in Section 4.4).

The results on CSR also suggest that the weight matrix of GNMF and the labels of LCNMF are not helpful on satisfying such RPR constraints. When there are only independent constraints, GNMF and LCNMF can satisfy them by setting proper weights and labels (e.g. for a constraint $dis(\boldsymbol{a}, \boldsymbol{b}) < dis(\boldsymbol{b}, \boldsymbol{c})$, GNMF sets weight 1 for $(\boldsymbol{a}, \boldsymbol{b})$ and 0 for $(\boldsymbol{b}, \boldsymbol{c})$, and LCNMF sets $(\boldsymbol{a}, \boldsymbol{b})$ the same label). However, when the constraints are not independent (such as in chains), GNMF cannot simply set the weights 1s and 0s, and LCNMF can only guarantee satisfying one constraint of a group of chain constraints (e.g. for a 3-chain of constraints $dis(\boldsymbol{a}, \boldsymbol{b}) < dis(\boldsymbol{b}, \boldsymbol{c}) < dis(\boldsymbol{c}, \boldsymbol{d})$, the weight of $(\boldsymbol{b}, \boldsymbol{c})$ in GNMF has to be greater than 0 and less than 1, while in LCNMF, setting $(\boldsymbol{a}, \boldsymbol{b})$ the same label can satisfy the first constraint, but the only way to satisfy the second constraint is to set $(\boldsymbol{b}, \boldsymbol{c})$ the same label, which will

contradict with the first constraint). Thus, GNMF and LCNMF cannot guarantee most of the constraints are satisfied after factorisation (recall the example in Figure 2.2).

The above two synthetic experiments both demonstrate that RPR-NMF algorithms have the advantages of getting accurate factorisation and satisfying relative constraints comparing with other algorithms.

3.5.3 Parameter Selection

The parameters introduced by RPR-NMF are the penalty coefficients. Experiments with varying values of these parameters are conducted under the following settings: $N = 100, M = 100, K = 20, l_{\boldsymbol{W}} = l_{\boldsymbol{H}} = N/10$. $\lambda_{\boldsymbol{W}}$ and $\lambda_{\boldsymbol{H}}$ vary from 0.4 to 4 with step 0.4 and from 20 to 100 with step 20. The average results of 10 repetitions are presented in Figure 3.4.

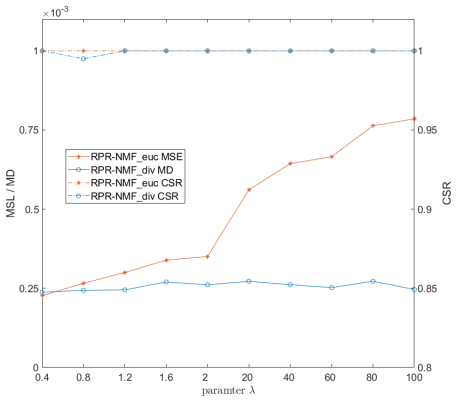


Figure 3.4 : MSL/MD & CSR of RPR-NMF algorithms with penalty coefficients varying from 0.4 to 2 and from 20 to 100.

As showed in the figure, the coefficients have nearly no influence on the CSR for both algorithms. However, the MSL of RPR-NMF using Euclidean measure increases when its parameters become larger, while the MD of the Divergence version

is stable all the time. This suggests that RPR-NMF using Divergence measure is not sensitive to its parameters which can be set without much effort, but using smaller parameters is better when working with Euclidean measure.

3.5.4 Performance Analysis in Image Clustering

The performance of algorithms is validated in image clustering on two public datasets with ground truth: AT&T ORL and CMU PIE.

AT&T ORL Dataset

The AT&T ORL database consists of 400 images for 40 classes with 10 different facial images in each class [70]. The images were taken at different times, lighting and facial expressions. The faces are in an upright position in frontal view, with a slight left-right rotation. Each image is preprocessed into a 32×32 matrix with 256 grey levels [23]. Thus the size of factorising matrix is $1,024 \times 400$.

The following steps are adopted in this experiment:

i. Randomly choose K classes and mix up images from these classes to form the factorising matrix;

ii. In the K classes, randomly select 2 images in each class and set them more similar to images chosen in other classes, which forms the list of RPR constraints for RPR-NMF; transform the constraints into a weight matrix and a label matrix; the latent dimension is set as the number of chosen classes K;

iii. Run algorithms to obtain the right factorised matrix $\boldsymbol{H}$; utilise K-means method on $\boldsymbol{H}$ to get the clustering results;

iv. Calculate the ACC and NMI for each algorithm.

The number of class K varies from 2 to 10 with step 2. The penalty coefficients for RPR-NMF using Euclidean measure are set 20 and for its Divergence version

Table 3.2 : Clustering Performance on AT&T ORL (Euclidean / Divergence)

	K	$l_{\boldsymbol{H}}$	FSNMF	ANLS	NMF	GNMF	LCNMF	RPR-NMF
MSL / MD	5	10	561.3	**335.2**	339.9 / **1.604**	397.9 / 1.980	355.1 / 1.677	340.7 / 1.606
	10	20	432.3	**252.2**	263.2 / 1.200	304.5 / 1.589	288.8 / 1.294	261.7 / **1.197**
	20	40	422.0	**232.8**	253.7 / 1.155	267.2 / 1.610	284.1 / 1.293	252.6 / **1.147**
	30	60	400.2	**202.8**	228.6 / **1.039**	233.4 / 1.494	265.5 / 1.207	228.1 / 1.041
	40	80	375.9	**180.5**	211.0 / 0.960	209.2 / 1.400	247.1 / 1.116	210.9 / **0.957**
	Avg.		438.3	**240.7**	259.3 / 1.192	282.4 / 1.615	288.1 / 1.317	258.8 / **1.190**
CSR (%)	5	10	94.00	92.00	90.00 / 86.00	98.00 / **100.0**	**100.0** / **100.0**	98.00 / 96.00
	10	20	81.00	89.00	88.00 / 87.00	98.00 / **100.0**	**100.0** / **100.0**	94.00 / **100.00**
	20	40	84.50	90.50	92.00 / 83.50	91.00 / **100.0**	**100.0** / **100.0**	91.50 / **100.00**
	30	60	86.00	89.33	91.00 / 89.00	92.33 / **100.0**	**100.0** / **100.0**	97.00 / **100.00**
	40	80	85.25	90.50	91.25 / 89.25	93.50 / **100.0**	**100.0** / **100.0**	97.00 / **100.00**
	Avg.		86.15	90.27	90.45 / 86.95	94.57 / **100.0**	**100.0** / **100.0**	95.50 / 99.20
ACC (%)	5	10	74.00	82.80	78.40 / 76.40	67.20 / 65.60	89.20 / 85.20	**90.40** / **88.40**
	10	20	58.60	65.40	64.20 / 68.00	63.40 / 55.60	75.00 / 71.20	**78.80** / **71.80**
	20	40	52.50	63.40	64.10 / 63.50	62.10 / 53.30	64.80 / **70.40**	**72.20** / 68.00
	30	60	50.53	56.00	62.00 / 60.93	59.33 / 48.47	66.20 / 63.73	**67.93** / **64.67**
	40	80	46.00	53.65	61.25 / 58.25	56.95 / 46.85	62.95 / **59.20**	**63.50** / 58.10
	Avg.		56.33	64.25	65.99 / 65.42	61.80 / 53.96	71.63 / 69.95	**74.57** / **70.19**
NMI (%)	5	10	70.12	80.87	73.37 / 79.29	59.01 / 51.76	85.33 / **83.03**	**85.35** / 82.05
	10	20	67.55	72.66	73.28 / 74.10	67.69 / 58.00	81.24 / 77.89	**82.12** / **78.40**
	20	40	67.83	75.88	77.11 / 76.33	73.57 / 60.59	79.06 / **81.79**	**82.44** / 78.19
	30	60	68.28	73.47	77.92 / 76.80	74.92 / 59.82	80.12 / 78.85	**80.34** / **78.89**
	40	80	68.07	72.89	79.01 / 77.65	75.62 / 60.39	79.22 / **78.01**	**79.29** / 75.71
	Avg.		68.37	75.15	76.14 / 76.83	70.16 / 58.11	80.99 / **79.91**	**81.91** / 78.65

are set 2. The results are showed in Table 3.2.

From the table, RPR-NMF using Euclidean measure achieves the best average ACC (74.57%) and the best average NMI (81.91%) as well. Among algorithms using Euclidean measure, RPR-NMF outperforms FSNMF, ANLS, NMF, GNMF and LCNMF by 18.24%, 10.32%, 8.58%, 12.77%, 2.94% on ACC and by 13.54%, 6.76%,

5.77%, 11.75%, 0.92% on NMI respectively; as for algorithms using Divergence measure, RPR-NMF outperforms NMF and GNMF by 4.77%, 16.23% on ACC and by 1.82%, 20.54% on NMI respectively, and is slightly better than LCNMF on ACC but has a bit lower NMI score than LCNMF. ANLS obtains the lowest factorisation errors, followed by RPR-NMF algorithms. All of the algorithms obtain very high CSR, because the images in this dataset are quite distinguishing. Besides, there is no chain among RPR constraints, thus LCNMF can satisfy all of them by setting proper labels.

CMU PIE Dataset

CMU PIE database was collected at Carnegie Mellon University in 2000, and it has been very influential in advancing research in face recognition across pose and illumination [71]. Following the pre-processed PIE dataset used in [22] which contains $2,856$ images for 68 different people with 42 images for each person, the images are processed into 32×32 matrices denoting the grey level of pixels. Thus the size of factorising matrix is $1,024 \times 2,856$.

Similar experimental steps as those for AT&T ORL dataset are adopted, with a few changes on the extraction of RPR constraints. For this experiment, 4 images instead of 2 in each cluster are randomly selected. Moreover, considering that the similarity can exist not only among intra-class images, but also inter-class images (e.g. the image of a dog is more similar to that of another dog, rather than a cat), the constraints are extracted in both ways. The number of class K varies from 10 to 60 with step 10. The penalty coefficients for RPR-NMF using Euclidean measure are set 20. For its Divergence version, they are set 2. The results are presented in Table 3.3.

According to the table, we can see that ANLS and NMF using Divergence measure achieve the best average approximation (120.0 MSL and 1.242 MD). As for

Table 3.3 : Clustering Performance on CMU PIE (Euclidean / Divergence)

	K	$l_{\boldsymbol{H}}$	FSNMF	ANLS	NMF	GNMF	LCNMF	RPR-NMF
MSL / MD	10	120	548.4	**191.2**	203.1 / **1.788**	203.8 / 2.916	429.8 / 3.234	210.1 / 1.852
	20	240	589.6	**147.4**	171.6 / **1.385**	163.2 / 2.689	425.3 / 3.063	161.4 / 1.478
	30	360	510.6	**124.1**	151.4 / **1.292**	140.6 / 2.447	397.4 / 2.944	141.5 / 1.370
	40	480	578.8	**106.5**	137.3 / **1.147**	122.9 / 2.282	382.4 / 2.793	125.2 / 1.219
	50	600	592.2	**98.11**	130.0 / **1.088**	113.7 / 2.217	385.8 / 2.796	118.1 / 1.139
	60	720	610.6	**89.88**	122.3 / **1.013**	105.5 / 2.124	378.2 / 2.731	110.4 / 1.102
	68	816	599.1	**82.74**	116.9 / **0.978**	98.83 / 2.037	370.8 / 2.693	103.5 / 0.998
	Avg.		575.6	**120.0**	147.5 / **1.242**	135.5 / 2.387	395.7 / 2.893	138.6 / 1.308
CSR (%)	10	120	72.83	78.17	84.00 / 78.67	80.50 / 68.83	00.00 / 00.00	**84.17** / **82.83**
	20	240	72.92	78.83	82.83 / **73.25**	82.25 / 70.58	00.00 / 00.00	**85.42** / 71.75
	30	360	73.06	78.72	82.17 / 72.89	83.22 / 68.28	00.00 / 00.00	**86.78** / **76.83**
	40	480	74.96	79.71	83.13 / 76.04	84.46 / 70.58	00.00 / 00.00	**88.08** / **77.29**
	50	600	74.40	78.47	82.50 / **76.83**	84.27 / 67.53	00.00 / 00.00	**88.43** / 75.60
	60	720	72.78	79.14	81.47 / 74.92	83.69 / 67.67	00.00 / 00.00	**87.83** / **75.50**
	68	816	74.26	78.92	81.30 / **74.34**	84.14 / 68.70	00.00 / 00.00	**88.51** / 71.57
	Avg.		73.60	78.85	82.49 / 75.28	83.22 / 68.88	00.00 / 00.00	**87.03** / **75.91**
ACC (%)	10	120	32.48	55.86	64.86 / 63.19	51.76 / 53.71	60.29 / 57.52	**66.43** / **66.76**
	20	240	27.21	56.83	58.00 / 62.57	60.79 / 51.10	56.88 / 57.05	**67.36** / **66.64**
	30	360	27.06	59.41	62.49 / 61.89	58.21 / 53.51	57.87 / 57.16	**66.32** / **66.73**
	40	480	26.13	59.12	61.73 / 59.35	60.23 / 54.20	56.25 / 54.13	**64.51** / **66.17**
	50	600	25.53	53.22	60.63 / 61.62	58.98 / 52.15	56.77 / 56.17	**64.49** / **66.24**
	60	720	24.44	56.37	61.36 / 60.09	57.06 / 53.97	55.23 / 53.38	**62.31** / **66.48**
	68	816	23.82	53.03	57.75 / 60.78	58.60 / 51.51	52.72 / 53.79	**62.40** / **63.10**
	Avg.		26.67	56.26	60.97 / 61.36	57.95 / 52.88	56.57 / 55.60	**64.83** / **66.02**
NMI (%)	10	120	28.57	56.02	**67.13** / **67.10**	57.73 / 52.70	55.78 / 55.98	67.01 / 64.38
	20	240	34.24	67.04	68.54 / **72.07**	68.63 / 59.70	60.88 / 61.87	**73.61** / 71.00
	30	360	38.93	71.48	71.87 / **72.60**	70.37 / 63.14	64.71 / 63.66	**75.06** / 72.55
	40	480	39.19	72.44	73.15 / 73.01	72.66 / 63.26	64.40 / 65.35	**73.94** / **74.63**
	50	600	41.80	70.52	73.85 / 74.40	73.41 / 64.53	66.48 / 65.06	**75.67** / **74.96**
	60	720	43.07	73.19	**75.47** / 74.49	73.35 / 66.32	66.68 / 66.12	75.33 / **75.05**
	68	816	43.48	72.01	73.32 / **75.25**	74.18 / 66.31	66.38 / 65.65	**75.72** / 74.21
	Avg.		38.47	68.96	71.90 / **72.70**	70.05 / 62.28	63.62 / 63.38	**73.76** / 72.40

the other three evaluation metrics, RPR-NMF using Euclidean measure achieves the best average CSR (87.03%) and NMI (73.76%) while RPR-NMF using Divergence measure achieves the best average ACC (66.02%). Notice that the CSR of LCNMF are all zeros because inter-class and intra-class constraints lead to cyclic chain constraints.

Among the algorithms using Euclidean measure, RPR-NMF outperforms FS-NMF, ANLS, NMF, GNMF and LCNMF by 38.16%, 8.57%, 3.86%, 6.88%, 8.26% on ACC and by 35.29%, 4.80%, 1.86%, 3.71%, 10.14% on NMI respectively. For the algorithms using Divergence measure, RPR-NMF outperforms NMF, GNMF and LCNMF by 4.66%, 13.145%, 10.42% on ACC. However, its performance on NMI is a bit lower than NMF, while it outperforms GNMF and LCNMF by 10.14%, 9.02%.

3.5.5 Performance Analysis in Recommender Systems

As for the performance analysis in recommender systems, RPR-NMF is compared with NMF, GNMF, and LCNMF on Movielens 1M dataset. For the reason that the rating matrices in recommender systems have missing values which are usually denoted by zeros, all the algorithms have to be modified with a $MASK$ matrix for incomplete factorising matrix as proposed in [72]. The modified versions for all algorithms have been implemented except for the GNMF using Divergence measure. In the Appendix of [22], the authors only mentioned how to deal with incomplete factorising matrix for GNMF using Euclidean measure. Thus GNMF using Divergence measure is not compared in this part.

Specifically, the pairwise relationship constraints in this part are extracted from meta information and are imposed on both factorised matrices: users' gender, age and occupation as well as movies' genre are used to obtain RPRs for the Movielens dataset.

Movielens 1M Dataset

The Movielens 1M dataset is a well-known stable baseline dataset in recommender systems [73]. It has $1,000,209$ ratings (1 to 5) from $6,040$ users on $3,883$ movies. Indeed some movies have no ratings, thus after removing these movies, a pre-processed $6,040 \times 3,706$ rating matrix is derived.

Three groups of cross validation experiments are conducted on this dataset: (1) 300 constraints, $K = 20$; (2) 500 constraints and $K = 50$; (3) 1000 constraints and $K = 100$. The penalty coefficients for RPR-NMF using Euclidean measure are set 200, 10 and 1 respectively, and the coefficients for the Divergence version of RPR-NMF are set 0.1, 0.01, 0.001 respectively. For each group, conduct 5 cross validation experiments and the average results are showed in Table 3.4.

As presented in the table, NMF achieves the lowest MSL and RPR-NMF achieves the lowest MD; both of the algorithms obtain close approximation errors while the other two methods, GNMF and LCNMF, have a higher rate of errors. RPR-NMF also achieves the highest CSR on both its versions. As for the recommendation evaluation, GNMF using Euclidean measure achieves the lowest RMSE while RPR-NMF using Euclidean measure achieves the highest F1 score. It is worth noting that, in this experiment, the RPRs are randomly selected through meta information, and some of the constraints may not represent the true ratings' pattern. Thus the RPR-NMF algorithms seem not greatly outperform existing methods. However, their overall performance is still better compared to other alternatives.

3.6 Summary

In this chapter, a novel MF algorithm called RPR-NMF is introduced to effectively utilise the relative pairwise relationship among rows or columns of factorised matrices. Both of the Euclidean and Divergence measures are used in the objective

Table 3.4 : Cross Validation Results on Movielens 1M (Euclidean / Divergence)

	K	$l_{\boldsymbol{W}}$ & $l_{\boldsymbol{H}}$	NMF	GNMF_euc	LCNMF	RPR-NMF
MSL / MD	20	300	**0.513** / 0.083	0.526	0.532/ 0.086	0.514 / **0.082**
	50	600	**0.373** / **0.060**	0.409	0.422 / 0.068	0.374 / **0.060**
	100	900	**0.249** / **0.039**	0.317	0.334 / 0.053	0.253 / **0.039**
		Avg.	**0.378** / 0.061	0.417	0.429 / 0.069	0.380 / **0.060**
CSR	20	300	49.07 / 49.87	87.37	50.00 / 50.00	**96.17** / **95.77**
	50	600	49.23 / 51.03	89.25	50.00 / 50.00	**95.13** / **98.83**
	100	900	50.17 / 51.51	89.72	49.94 / 49.94	**93.49** / **99.39**
		Avg.	49.38 / 50.80	88.78	49.98 / 49.98	**94.93** / **98.00**
RMSE	20	300	0.959 / 0.975	0.929	0.991 / 1.004	**0.928** / **0.974**
	50	600	1.027 / 1.045	**0.961**	1.044 / 1.068	0.979 / **1.030**
	100	900	1.100 / **1.147**	**1.010**	1.133 / 1.165	1.055 / 1.157
		Avg.	1.028 / **1.055**	**0.966**	1.056 / 1.079	0.987 / 1.056
F1 Score	20	300	68.75 / 68.67	68.29	66.39 / 66.23	**69.25** / **68.71**
	50	600	67.34 / **67.20**	67.45	64.78 / 64.80	**67.51** / 67.19
	100	900	65.10 / 65.01	65.31	62.20 / 62.26	**65.46** / **65.03**
		Avg.	67.06 / 66.96	67.02	64.46 / 64.43	**67.41** / **66.98**

function. RPR-NMF imposes penalties for each relative pairwise relationship constraint in a form of addition of natural exponential functions for Euclidean measure and hinge loss for Divergence measure. Complete and sufficient proofs of convergence are also provided to ensure that RPR-NMF conforms to the "multiplicative update rules". Numerical analysis shows that the proposed algorithm achieves superior performance on both the overall loss and the accuracy of satisfied constraints compared with the other algorithms. Experiments on synthetic and real datasets for image clustering and recommender systems demonstrate the effectiveness of RPR-NMF algorithms which outperform baseline methods on several evaluation criteria.

The experimental results presented in this work show that the selection of RPRs also plays an important role in an overall recommender system. Although outside of the scope of this chapter, in the future work, approaches to select application-oriented RPR constraints will be explored.

Chapter 4

Relative Pairwise Relationship Constrained Non-negative Tensor Factorisation

In this chapter, a novel low-rank TF algorithm called RPR-NTF is introduced, which places RPR constraints in the objectives and conforms to MUL. It considers two TF schemes, the CP decomposition and the Tucker decomposition, and two distance measures, the squared Euclidean distance and the Divergence.

The RPR constraints placed on tensors are firstly introduced, followed by four objective functions using the squared Euclidean distance and the Divergence distance for both the CP decomposition and Tucker decomposition, respectively. After that, the update rules for each of the objective functions as well as their concise convergence proofs under MUL framework are presented. Equivalent updating formulas with respect to factor matrices and core tensor are also provided to increase algorithm efficiency. Last, RPR-NTF variations are proposed to deal with incomplete data input.

4.1 RPRs and Objectives

The RPR constraints are placed on the factor matrices, and can be defined as a set of integer indexed triplets for each factor matrix $\boldsymbol{U}^{(n)}$:

$$\boldsymbol{L}_{\boldsymbol{U}}^{(n)} \subseteq \{(q,r,s)|q,r,s \in \mathbb{N}^{+}, q,r,s \leqslant I_n, q \neq r \neq s\}. \tag{4.1}$$

The expected relationship $dis(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)}) < dis(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)})$ is represented by the lth triplet in $\boldsymbol{L}_{\boldsymbol{U}}^{(n)}$, where $dis(\cdot,\cdot)$ is a distance measure. The squared Euclidean distance(Eq. (2.2)) and Divergence (Eq. (2.3)) are used as objective functions.

To model RPRs, the squared Euclidean distance is used in the former and the Symmetric Divergence (Eq. (3.3)) in the latter.

Now the RPR constraints can be incorporate into the NTF objective functions as penalty terms. For the squared Euclidean distance measure, RPRs are modelled by natural exponential functions, while for the Divergence distance measure, they are in hinge loss functions. Therefore, the four objective functions are defined as following (the subscripts c, t, e, d stand for CP, Tucker, Euclidean and Divergence respectively):

i) CP decomposition using squared Euclidean distance

$$\begin{aligned}\mathcal{F}_{\text{ce}} = \|\boldsymbol{\mathcal{X}} - \sum_{k=1}^{K}\bigotimes_{n=1}^{N}\boldsymbol{u}_k^{(n)}\|_F^2 + \sum_{n=1}^{N}\lambda_n\sum_{l=1}^{l_U^{(n)}} \\ [\exp(E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)})) + \exp(-E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)}))] \\ s.t. \quad \lambda_n \geqslant 0, \quad \boldsymbol{U}_{i_n j}^{(n)} \geqslant 0, \\ i_n = 1, \cdots, I_n, \quad j = 1, \cdots, K,\end{aligned} \tag{4.2}$$

ii) CP decomposition using Divergence

$$\begin{aligned}\mathcal{F}_{\text{cd}} = D(\boldsymbol{\mathcal{X}}||\sum_{k=1}^{K}\bigotimes_{n=1}^{N}\boldsymbol{u}_k^{(n)}) + \sum_{n=1}^{N}\lambda_n\sum_{l=1}^{l_U^{(n)}} \\ \max(0, SD(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)}) - SD(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)})) \\ s.t. \quad \lambda_n \geqslant 0, \quad \boldsymbol{U}_{i_n j}^{(n)} \geqslant 0, \\ i_n = 1, \cdots, I_n, \quad j = 1, \cdots, K,\end{aligned} \tag{4.3}$$

iii) Tucker decomposition using squared Euclidean distance

$$\begin{aligned}\mathcal{F}_{\text{te}} = \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \cdots \times_N \boldsymbol{U}^{(N)}\|_F^2 + \sum_{n=1}^{N}\lambda_n \\ \sum_{l=1}^{l_U^{(n)}}[\exp(E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)})) + \exp(-E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)}))] \\ s.t. \quad \lambda_n \geqslant 0, \quad \boldsymbol{U}_{i_n j}^{(n)} \geqslant 0, \\ i_n = 1, \cdots, I_n, \quad j = 1, \cdots, K,\end{aligned} \tag{4.4}$$

iv) Tucker decomposition using Divergence

$$\begin{aligned}\mathcal{F}_{\text{td}} &= D(\boldsymbol{\mathcal{X}}||\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \cdots \times_N \boldsymbol{U}^{(N)}) + \sum_{n=1}^{N} \lambda_n \\ &\sum_{l=1}^{l_{\boldsymbol{U}}^{(n)}} \max(0, SD(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)}) - SD(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)})) \\ &s.t. \quad \lambda_n \geqslant 0, \quad \boldsymbol{U}_{i_n j}^{(n)} \geqslant 0, \\ &\quad i_n = 1, \cdots, I_n, \quad j = 1, \cdots, K,\end{aligned} \tag{4.5}$$

where $l_{\boldsymbol{U}}^{(n)}$ is the number of RPRs for factor matrix $\boldsymbol{U}^{(n)}$ and λ_n is the penalty coefficient.

4.2 Updating Rules of RPR-NTF

The above four objective functions are much more complex than those in RPR-NMF due to the complicated outer products and n-mode production introduced by CP decomposition and Tucker decomposition. Take the deduction of update rules for RPR-NTF using CP decomposition with squared Euclidean distance as an example. Update rules for other objective functions can be obtained in a similar fashion.

As for Eq. (4.2), with non-negative constraints $\boldsymbol{U}_{i_n k}^{(n)} \geqslant 0$, construct a Lagrange function as follows:

$$\begin{aligned}\mathcal{L} = \|\boldsymbol{\mathcal{X}} - \sum_{k=1}^{K} \bigotimes_{n=1}^{N} \boldsymbol{u}_k^{(n)}\|_F^2 + \sum_{n=1}^{N} \big(\sum_{i_n k} \alpha_{i_n k} \boldsymbol{U}_{i_n k}^{(n)} + \lambda_n \sum_{l=1}^{l_{\boldsymbol{U}}^{(n)}} \\ [\exp(E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)})) + \exp(-E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)}))]\big).\end{aligned} \tag{4.6}$$

The partial derivative with respect to $\boldsymbol{U}_{i_n k}^{(n)}$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}_{i_n k}^{(n)}} = \alpha_{i_n k} + 2\big[- \sum_{\not\ni i_n} \boldsymbol{\mathcal{X}}_{i_1 \cdots i_N} \prod_{r \neq n} \boldsymbol{U}_{i_r k}^{(r)} \\ + \sum_{m=1}^{K} \boldsymbol{U}_{i_n m}^{(n)} \prod_{r \neq n} (\boldsymbol{u}_m^{(r)})^T (\boldsymbol{u}_k^{(r)}) + \lambda_n C(\boldsymbol{U}_{i_n k}^{(n)})\big],\end{aligned} \tag{4.7}$$

where the lower index $\not\ni i_n$ in the first summation denotes to loop over all dimensions but i_n, and

$$
\begin{aligned}
C(\boldsymbol{U}^{(n)}_{i_nk}) = &\sum_{l=1}^{l^{(n)}_{\boldsymbol{U}}} \Big\{ \\
&\exp(E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{r^l:}))[\sum_{q^l=i_n}(\boldsymbol{U}^{(n)}_{q^lk} - \boldsymbol{U}^{(n)}_{r^lk}) + \sum_{r^l=i_n}(\boldsymbol{U}^{(n)}_{r^lk} - \boldsymbol{U}^{(n)}_{q^lk})] - \\
&\exp(-E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{s^l:}))[\sum_{q^l=i_n}(\boldsymbol{U}^{(n)}_{q^lk} - \boldsymbol{U}^{(n)}_{s^lk}) + \sum_{s^l=i_n}(\boldsymbol{U}^{(n)}_{s^lk} - \boldsymbol{U}^{(n)}_{q^lk})] \Big\} \\
= &\sum_{l=1}^{l^{(n)}_{\boldsymbol{U}}} \Big(\exp(E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{r^l:}))(\sum_{q^l=i_n}\boldsymbol{U}^{(n)}_{q^lk} + \sum_{r^l=i_n}\boldsymbol{U}^{(n)}_{r^lk}) + \\
&\exp(-E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{s^l:}))(\sum_{q^l=i_n}\boldsymbol{U}^{(n)}_{s^lk} + \sum_{s^l=i_n}\boldsymbol{U}^{(n)}_{q^lk}) \Big) - \\
&\sum_{l=1}^{l^{(n)}_{\boldsymbol{U}}} \Big(\exp(E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{r^l:}))(\sum_{q^l=i_n}\boldsymbol{U}^{(n)}_{r^lk} + \sum_{r^l=i_n}\boldsymbol{U}^{(n)}_{q^lk}) + \\
&\exp(-E(\boldsymbol{U}^{(n)}_{q^l:}, \boldsymbol{U}^{(n)}_{s^l:}))(\sum_{q^l=i_n}\boldsymbol{U}^{(n)}_{q^lk} + \sum_{s^l=i_n}\boldsymbol{U}^{(n)}_{s^lk}) \Big) \\
= &\, C^+(\boldsymbol{U}^{(n)}_{i_nk}) - C^-(\boldsymbol{U}^{(n)}_{i_nk}).
\end{aligned}
\tag{4.8}
$$

Let the partial derivative vanish and considering the non-negative constraints ($\alpha_{i_nk}\boldsymbol{U}^{(n)}_{i_nk} = 0$ under K.K.T conditions), obtain:

$$
\begin{aligned}
&\{\sum_{\not\ni i_n} \boldsymbol{\mathcal{X}}_{i_1\cdots i_N} \prod_{r\neq n} \boldsymbol{U}^{(r)}_{i_rk} + \lambda_n C^-(\boldsymbol{U}^{(n)}_{i_nk})\}\boldsymbol{U}^{(n)}_{i_nk} - \\
&\{\sum_{m=1}^{K} \boldsymbol{U}^{(n)}_{i_nm} \prod_{r\neq n} (\boldsymbol{u}^{(r)}_m)^T(\boldsymbol{u}^{(r)}_k) + \lambda_n C^+(\boldsymbol{U}^{(n)}_{i_nk})\}\boldsymbol{U}^{(n)}_{i_nk} = 0,
\end{aligned}
\tag{4.9}
$$

thus the update rule for $\boldsymbol{U}^{(n)}_{i_nk}$ is:

$$
\boldsymbol{U}^{(n)}_{i_nk} \leftarrow \boldsymbol{U}^{(n)}_{i_nk} \frac{\sum_{\not\ni i_n} \boldsymbol{\mathcal{X}}_{i_1\cdots i_N} \prod_{r\neq n} \boldsymbol{U}^{(r)}_{i_rk} + \lambda_n C^-(\boldsymbol{U}^{(n)}_{i_nk})}{\sum_{m=1}^{K} \boldsymbol{U}^{(n)}_{i_nm} \prod_{r\neq n} (\boldsymbol{u}^{(r)}_m)^T(\boldsymbol{u}^{(r)}_k) + \lambda_n C^+(\boldsymbol{U}^{(n)}_{i_nk})}. \tag{4.10}
$$

The above update rule causes a large amount of redundant calculations if we follow it to update the factor matrices element-wisely. As pointed out in [41], this update rule can be written in an equivalent form with respect to factor matrices

instead of the elements inside them, which is much easier and faster to compute. However, because the study in [41] used different definitions for the Kronecker product and Khatri-Rao product to what researchers usually adopt, their formulas are somewhat misleading. For the common definitions of these two operations, please refer to Definition 2.2 and 2.3 in Chapter 2.

With the help of the Khatri-Rao product, the CP decomposition in Eq. (2.17) can also be written as

$$\boldsymbol{\mathcal{X}}_{(n)} \approx \boldsymbol{\mathcal{R}}_{(n)} = \boldsymbol{U}^{(n)}(\bigodot_{r=N,r\neq n}^{1} \boldsymbol{U}^{(r)})^T, \tag{4.11}$$

where $\boldsymbol{\mathcal{X}}_{(n)}$ is the mode-n matricisation of $\boldsymbol{\mathcal{X}}$. Let $\boldsymbol{V}^{(n)} = (\bigodot_{r=N;r\neq n}^{1} \boldsymbol{U}^{(r)})^T$, then

$$\boldsymbol{\mathcal{X}}_{(n)} \approx \boldsymbol{\mathcal{R}}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)}, \tag{4.12}$$

which can be regarded as an NMF problem. Thus, the update rule in Eq. (4.10) is equivalent to

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k} \frac{(\boldsymbol{\mathcal{X}}_{(n)}(\boldsymbol{V}^{(n)})^T)_{i_n k} + \lambda_n C^-(\boldsymbol{U}^{(n)}_{i_n k})}{(\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)}(\boldsymbol{V}^{(n)})^T)_{i_n k} + \lambda_n C^+(\boldsymbol{U}^{(n)}_{i_n k})}. \tag{4.13}$$

As for RPR-NTF using CP decomposition with Divergence, the update rule for $\boldsymbol{U}^{(n)}_{i_n k}$ is

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k} \frac{[(\boldsymbol{\mathcal{X}}_{(n)}/(\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)}))(\boldsymbol{V}^{(n)})^T]_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}^{(n)}_{i_n k}) + [\boldsymbol{\mathcal{E}}_{(n)}(\boldsymbol{V}^{(n)})^T]_{i_n k}}, \tag{4.14}$$

where $\boldsymbol{\mathcal{E}}$ is a tensor of the same size as $\boldsymbol{\mathcal{X}}$ with all ones and

$$\begin{aligned} P(\boldsymbol{U}^{(n)}_{i_n k}) = \sum_{l=1}^{l_U^{(n)}} \Bigg\{ & \sum_{q^l=i_n} [g(\boldsymbol{U}^{(n)+}_{q^l k}, \boldsymbol{U}^{(n)+}_{r^l k}) + g(\boldsymbol{U}^{(n)+}_{q^l k}, \boldsymbol{U}^{(n)+}_{s^l k})] \\ & + \sum_{r^l=i_n} g(\boldsymbol{U}^{(n)+}_{r^l k}, \boldsymbol{U}^{(n)+}_{q^l k}) - \sum_{s^l=i_n} g(\boldsymbol{U}^{(n)+}_{s^l k}, \boldsymbol{U}^{(n)+}_{q^l k}) \Bigg\}, \end{aligned} \tag{4.15}$$

$$\boldsymbol{U}^{(n)+}_{q^l k} = \begin{cases} 0, & \text{if the constraint } l \text{ is satisfied} \\ \boldsymbol{U}^{(n)}_{q^l k} & \text{otherwise} \end{cases} \tag{4.16}$$

$$g(x,y) = log\frac{x}{y} + (x-y)\frac{1}{x}. \tag{4.17}$$

As for RPR-NTF using Tucker decomposition, one needs to update not only the factor matrices, but also the core tensor.

With the help of the Kronecker product, the Tucker decomposition in Eq. (2.18) can be written as

$$\boldsymbol{\mathcal{X}}_{(n)} \approx \boldsymbol{\mathcal{R}}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\mathcal{G}}_{(n)}(\bigotimes_{r=N,r\neq n}^{1} \boldsymbol{U}^{(r)})^T. \tag{4.18}$$

Let $\boldsymbol{D}^{(n)} = \boldsymbol{\mathcal{G}}_{(n)}(\bigotimes_{r=N,r\neq n}^{1} \boldsymbol{U}^{(r)})^T$, then

$$\boldsymbol{\mathcal{X}}_{(n)} \approx \boldsymbol{\mathcal{R}}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)}, \tag{4.19}$$

which can also be regarded as an NMF problem. Thus, the update rule for $\boldsymbol{U}^{(n)}_{i_n k}$ in RPR-NTF using the Tucker decomposition with squared Euclidean distance is

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k}\frac{(\boldsymbol{\mathcal{X}}_{(n)}(\boldsymbol{D}^{(n)})^T)_{i_n k} + \lambda_n C^{-}(\boldsymbol{U}^{(n)}_{i_n k})}{(\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)}(\boldsymbol{D}^{(n)})^T)_{i_n k} + \lambda_n C^{+}(\boldsymbol{U}^{(n)}_{i_n k})}. \tag{4.20}$$

To update the core tensor, one needs to represent the Tucker decomposition in another form

$$\mathrm{vec}(\boldsymbol{\mathcal{X}}) \approx \mathrm{vec}(\boldsymbol{\mathcal{R}}) = (\bigotimes_{n=N}^{1} \boldsymbol{U}^{(n)}) \cdot \mathrm{vec}(\boldsymbol{\mathcal{G}}), \tag{4.21}$$

where $\mathrm{vec}(\cdot)$ denotes vectorisation of a tensor. Then

$$\mathrm{vec}(\boldsymbol{\mathcal{G}}) \leftarrow \mathrm{vec}(\boldsymbol{\mathcal{G}})\frac{(\bigotimes_{n=N}^{1} \boldsymbol{U}^{(n)})^T \mathrm{vec}(\boldsymbol{\mathcal{X}})}{(\bigotimes_{n=N}^{1} \boldsymbol{U}^{(n)})^T(\bigotimes_{n=N}^{1} \boldsymbol{U}^{(n)})\,\mathrm{vec}(\boldsymbol{\mathcal{G}})}, \tag{4.22}$$

which is equivalent to

$$\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{G}} \cdot \frac{\boldsymbol{\mathcal{X}} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{\boldsymbol{\mathcal{R}} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}, \tag{4.23}$$

As for RPR-NTF using the Tucker decomposition with Divergence, the update rule for $\boldsymbol{U}^{(n)}_{i_n k}$ is

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k}\frac{[(\boldsymbol{\mathcal{X}}_{(n)}/(\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)}))(\boldsymbol{D}^{(n)})^T]_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}^{(n)}_{i_n k}) + [\boldsymbol{\mathcal{E}}_{(n)}(\boldsymbol{D}^{(n)})^T]_{i_n k}}, \tag{4.24}$$

and the update rule for the core tensor is

$$\mathcal{G} \leftarrow \mathcal{G} \cdot \frac{(\mathcal{X}/\mathcal{R}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{\mathcal{E} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}, \tag{4.25}$$

where $(\mathcal{X}/\mathcal{R})$ is the result of element-wise division between $\mathcal{X}$ and $\mathcal{R}$.

With respect to all these update rules, there is a theorem as follows:

Theorem 4.1. *The objective functions $\mathcal{F}_{ce}$ in Eq. (4.2), $\mathcal{F}_{cd}$ in Eq. (4.3), $\mathcal{F}_{te}$ in Eq. (4.4) and $\mathcal{F}_{td}$ in Eq. (4.5) are non-increasing under the update rules in Eq. (4.13), Eq. (4.14), Eq. (4.20) & (4.23) and Eq. (4.24) & (4.25) respectively with appropriate penalty coefficients λ_n.*

4.3 Convergence Proofs

To prove the convergence of the above update rules, i.e. the proposed algorithms conform to MUL, one needs to construct appropriate auxiliary functions. As described in Chapter 2, the auxiliary function $\mathcal{G}$ for an algorithm under MUL must have two properties: i) $\mathcal{G}(x, x') \geqslant \mathcal{F}(x)$, and ii) $\mathcal{G}(x, x) = \mathcal{F}(x)$. It guarantees $\mathcal{F}(x)$ to be non-increasing by using $x^{t+1} = \arg\min_x \mathcal{G}(x, x')$. Let $\mathcal{F}_{i_n k}(\boldsymbol{U}^{(n)})$ denote the part of $\mathcal{F}(\boldsymbol{U}^{(n)})$ concerning $\boldsymbol{U}^{(n)}_{i_n k}$, so as $\mathcal{G}_{i_n k}(\boldsymbol{U}^{(n)}, \boldsymbol{U}^{(n)t})$, and let $\boldsymbol{g} = \text{vec}(\mathcal{G})$, $\boldsymbol{x} = \text{vec}(\mathcal{X})$, $\boldsymbol{A} = \bigotimes_{n=N}^{1} \boldsymbol{U}^{(n)}$.

Lemma 4.1. *Function*

$$\begin{aligned} \mathcal{G}_{i_n k}(\boldsymbol{U}^{(n)}, \boldsymbol{U}^{(n)t}) &= \mathcal{F}_{i_n k}(\boldsymbol{U}^{(n)t}) \\ &+ (\boldsymbol{U}^{(n)}_{i_n k} - \boldsymbol{U}^{(n)t}_{i_n k}) \mathcal{F}'_{i_n k}(\boldsymbol{U}^{(n)t}) + (\boldsymbol{U}^{(n)}_{i_n k} - \boldsymbol{U}^{(n)t}_{i_n k})^2 \\ &\cdot \frac{(\boldsymbol{U}^{(n)t} \boldsymbol{V}^{(n)} (\boldsymbol{V}^{(n)})^T)_{i_n k} + \lambda_n C^+(\boldsymbol{U}^{(n)t}_{i_n k})}{\boldsymbol{U}^{(n)t}_{i_n k}} \end{aligned} \tag{4.26}$$

is an auxiliary function for $\mathcal{F}_{i_n k}(\boldsymbol{U}^{(n)})$ in Eq. (4.2).

Lemma 4.2. *Function*

$$\begin{aligned}\mathcal{G}_{i_nk}(\boldsymbol{U}^{(n)},\boldsymbol{U}^{(n)t}) &= \sum_j (\boldsymbol{\mathcal{X}}_{(n)})_{i_nj}\log(\boldsymbol{\mathcal{X}}_{(n)})_{i_nj} - (\boldsymbol{\mathcal{X}}_{(n)})_{i_nj} \\ &+ \sum_j (\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)})_{i_nj} - \sum_{jm}\big\{(\boldsymbol{\mathcal{X}}_{(n)})_{i_nj}\frac{\boldsymbol{U}^{(n)t}_{i_nm}\boldsymbol{V}^{(n)}_{mj}}{(\boldsymbol{U}^{(n)t}\boldsymbol{V}^{(n)})_{i_nj}} \\ &\cdot(\log(\boldsymbol{U}^{(n)}_{i_nm}\boldsymbol{V}^{(n)}_{mj}) - \log\frac{\boldsymbol{U}^{(n)t}_{i_nm}\boldsymbol{V}^{(n)}_{mj}}{(\boldsymbol{U}^{(n)t}\boldsymbol{V}^{(n)})_{i_nj}})\big\} \\ &+\lambda_n\sum_{l=1}^{l_{\boldsymbol{U}}^{(n)}}\max(0, SD(\boldsymbol{U}^{(n)}_{q^l:},\boldsymbol{U}^{(n)}_{r^l:}) - SD(\boldsymbol{U}^{(n)}_{q^l:},\boldsymbol{U}^{(n)}_{s^l:}))\end{aligned} \tag{4.27}$$

is an auxiliary function for $\mathcal{F}_{i_nk}(\boldsymbol{U}^{(n)})$ in Eq. (4.3).

Lemma 4.3. *Functions*

$$\begin{aligned}\mathcal{G}_{i_nk}(\boldsymbol{U}^{(n)},\boldsymbol{U}^{(n)t}) &= \mathcal{F}_{i_nk}(\boldsymbol{U}^{(n)t}) \\ &+(\boldsymbol{U}^{(n)}_{i_nk} - \boldsymbol{U}^{(n)t}_{i_nk})\mathcal{F}'_{i_nk}(\boldsymbol{U}^{(n)t}) + (\boldsymbol{U}^{(n)}_{i_nk} - \boldsymbol{U}^{(n)t}_{i_nk})^2 \\ &\cdot\frac{(\boldsymbol{U}^{(n)t}\boldsymbol{D}^{(n)}(\boldsymbol{D}^{(n)})^T)_{i_nk} + \lambda_n C^+(\boldsymbol{U}^{(n)t}_{i_nk})}{\boldsymbol{U}^{(n)t}_{i_nk}}\end{aligned} \tag{4.28}$$

and

$$\begin{aligned}\mathcal{K}_j(\boldsymbol{g},\boldsymbol{g}^t) &= \mathcal{F}_j(\boldsymbol{g}^t) + (\boldsymbol{g}_j - \boldsymbol{g}^t_j)\mathcal{F}'_j(\boldsymbol{g}^t) \\ &+ (\boldsymbol{g}_j - \boldsymbol{g}^t_j)^2\frac{(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{g}^t)_j}{\boldsymbol{g}^t_j}\end{aligned} \tag{4.29}$$

are auxiliary functions for $\mathcal{F}_{i_nk}(\boldsymbol{U}^{(n)})$ and $\mathcal{F}_j(\boldsymbol{g})$ in Eq. (4.4).

Lemma 4.4. *Functions*

$$\begin{aligned}\mathcal{G}_{i_nk}(\boldsymbol{U}^{(n)},\boldsymbol{U}^{(n)t}) &= \sum_j (\boldsymbol{\mathcal{X}}_{(n)})_{i_nj}\log(\boldsymbol{\mathcal{X}}_{(n)})_{i_nj} - (\boldsymbol{\mathcal{X}}_{(n)})_{i_nj} \\ &+ \sum_j (\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)})_{i_nj} - \sum_{jm}\big\{(\boldsymbol{\mathcal{X}}_{(n)})_{i_nj}\frac{\boldsymbol{U}^{(n)t}_{i_nm}\boldsymbol{D}^{(n)}_{mj}}{(\boldsymbol{U}^{(n)t}\boldsymbol{D}^{(n)})_{i_nj}} \\ &\cdot(\log(\boldsymbol{U}^{(n)}_{i_nm}\boldsymbol{D}^{(n)}_{mj}) - \log\frac{\boldsymbol{U}^{(n)t}_{i_nm}\boldsymbol{D}^{(n)}_{mj}}{(\boldsymbol{U}^{(n)t}\boldsymbol{D}^{(n)})_{i_nj}})\big\} \\ &+\lambda_n\sum_{l=1}^{l_{\boldsymbol{U}}^{(n)}}\max(0, SD(\boldsymbol{U}^{(n)}_{q^l:},\boldsymbol{U}^{(n)}_{r^l:}) - SD(\boldsymbol{U}^{(n)}_{q^l:},\boldsymbol{U}^{(n)}_{s^l:}))\end{aligned} \tag{4.30}$$

and

$$\begin{aligned}\mathcal{K}_j(\boldsymbol{g}, \boldsymbol{g}^t) = &\sum_i (\boldsymbol{x}_i \log \boldsymbol{x}_i - \boldsymbol{x}_i) + \sum_i (\boldsymbol{A}\boldsymbol{g})_i \\ &- \sum_{im} \boldsymbol{x}_i \frac{\boldsymbol{A}_{im}\boldsymbol{g}_m^t}{(\boldsymbol{A}\boldsymbol{g}^t)_i} [\log(\boldsymbol{A}\boldsymbol{g}_m) - \log \frac{\boldsymbol{A}_{im}\boldsymbol{g}_m^t}{(\boldsymbol{A}\boldsymbol{g}^t)_i}]\end{aligned} \tag{4.31}$$

are auxiliary functions for $\mathcal{F}_{i_n k}(\boldsymbol{U}^{(n)})$ *and* $\mathcal{F}_j(\boldsymbol{g})$ *in Eq. (4.5).*

The key idea to prove Lemma 4.1 and 4.3 is to use the Taylor series on the objective function and compare the part of the second derivatives with its counterpart in the auxiliary function. To prove Lemma 4.2 and 4.4, it is essential to adopt Jensen's inequality to compare the different parts in the objective and auxiliary functions. Theorem 4.1 can then be proved in a similar way as Theorem 3.1 and 3.2.

4.4 Factorisation of Incomplete Tensor

In some scenarios, the data tensor can be incomplete, i.e. there are unknown entries. Thus, the objective functions should only consider the error of elements on the known entries instead of all entries. Given an additional indicator tensor $\mathcal{Z}$ in which ones denote known entries while zeros denote unknowns, the objective function in Eq. (4.2) becomes

$$\begin{aligned}\mathcal{F}_{ce} = &\|\mathcal{Z} * (\mathcal{X} - \sum_{k=1}^{K} \bigotimes_{n=1}^{N} \boldsymbol{u}_k^{(n)})\|_F^2 + \sum_{n=1}^{N} \lambda_n \sum_{l=1}^{l_{\boldsymbol{U}}^{(n)}} \\ &[\exp(E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{r^l:}^{(n)})) + \exp(-E(\boldsymbol{U}_{q^l:}^{(n)}, \boldsymbol{U}_{s^l:}^{(n)}))] \\ &s.t. \quad \lambda_n \geqslant 0, \quad \boldsymbol{U}_{i_n j}^{(n)} \geqslant 0, \\ &\quad i_n = 1, \cdots, I_n, \quad j = 1, \cdots, K,\end{aligned} \tag{4.32}$$

where $*$ denotes the element-wise multiplication. The corresponding update rule for $\boldsymbol{U}_{i_n k}^{(n)}$ in Eq. (4.13) becomes

$$\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{((\mathcal{Z}_{(n)} * \mathcal{X}_{(n)})(\boldsymbol{V}^{(n)})^T)_{i_n k} + \lambda_n C^-(\boldsymbol{U}_{i_n k}^{(n)})}{(\mathcal{Z}_{(n)} * (\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)})(\boldsymbol{V}^{(n)})^T)_{i_n k} + \lambda_n C^+(\boldsymbol{U}_{i_n k}^{(n)})}. \tag{4.33}$$

Similarly, Eq. (4.14) becomes

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k} \frac{[((\mathcal{Z}_{(n)} * \mathcal{X}_{(n)})/(\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)}))(\boldsymbol{V}^{(n)})^T]_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}^{(n)}_{i_n k}) + [\mathcal{Z}_{(n)}(\boldsymbol{V}^{(n)})^T]_{i_n k}}, \tag{4.34}$$

Eq. (4.20) and (4.23) become

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k} \frac{((\mathcal{Z}_{(n)} * \mathcal{X}_{(n)})(\boldsymbol{D}^{(n)})^T)_{i_n k} + \lambda_n C^{-}(\boldsymbol{U}^{(n)}_{i_n k})}{(\mathcal{Z}_{(n)} * (\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)})(\boldsymbol{D}^{(n)})^T)_{i_n k} + \lambda_n C^{+}(\boldsymbol{U}^{(n)}_{i_n k})}, \tag{4.35}$$

$$\mathcal{G} \leftarrow \mathcal{G} \cdot \frac{(\mathcal{Z} * \mathcal{X}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{(\mathcal{Z} * \mathcal{R}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}. \tag{4.36}$$

Eq. (4.24) and (4.25) become

$$\boldsymbol{U}^{(n)}_{i_n k} \leftarrow \boldsymbol{U}^{(n)}_{i_n k} \frac{[((\mathcal{Z}_{(n)} * \mathcal{X}_{(n)})/(\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)}))(\boldsymbol{D}^{(n)})^T]_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}^{(n)}_{i_n k}) + [\mathcal{Z}_{(n)}(\boldsymbol{D}^{(n)})^T]_{i_n k}}, \tag{4.37}$$

$$\mathcal{G} \leftarrow \mathcal{G} \cdot \frac{(\mathcal{Z} * \mathcal{X}/\mathcal{R}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{\mathcal{Z} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}. \tag{4.38}$$

When the data tensor is complete, one can simply set the indicator tensor $\mathcal{Z}$ to a tensor with all ones. If the RPR constraints are only placed on part of the factor matrices, the $\boldsymbol{L}^{(n)}_{\boldsymbol{U}}$ corresponding to those with no constraints can be set to empty.

The complete algorithms of RPR-NTF are presented in Algorithm 4.1, 4.2, 4.3 and 4.4 (Notations like ZXV and $ZUVV$ in Algorithm 4.1 denote temporal variables).

4.5 Experiments

In this section, numerical experiments are conducted to verify the effectiveness and efficiency of the proposed RPR-NTF compare to other NTF and NMF algorithms. Two chosen NTF algorithms have four combinations of the decomposition schemes and distance measures: Non-negative Tensor Factorisation in MUL framework (MUL-NTF), and Projected Gradient Based Non-negative Tensor Factorisation (PGD-NTF). The Laplacian Regularised Non-negative Tensor Factorisation

Algorithm 4.1 Relative Pairwise Relationship constrained Non-negative Tensor Factorisation using CP decomposition with squared Euclidean distance (RPR-NTF-CE)

INPUT: $\boldsymbol{\mathcal{X}} \in \mathbb{R}_{\geqslant 0}^{I_1 \times \cdots \times I_N}$, $\boldsymbol{\mathcal{Z}} \in \mathbb{Z}_2^{I_1 \times \cdots \times I_N}$, $K \in \mathbb{N}^+$, $\boldsymbol{L}_{\boldsymbol{U}}^{(n)} \in \mathbb{N}^{l_{\boldsymbol{U}^{(n)}} \times 3}$, $\lambda_n \in \mathbb{R}^+$

OUTPUT: $\boldsymbol{U}^{(n)} \in \mathbb{R}_{\geqslant 0}^{I_n \times K}$

Initialise all $\boldsymbol{U}^{(n)}$ as non-negative random matrices
while Terminal conditions not satisfied **do**
  for $n \leftarrow 1$ **to** N **do**
    $\boldsymbol{\mathcal{X}}_{(n)}, \boldsymbol{\mathcal{Z}}_{(n)} \leftarrow$ Matricise $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}$
    $\boldsymbol{V}^{(n)} \leftarrow (\bigodot_{r=N;r\neq n}^{1} \boldsymbol{U}^{(r)})^T$
    $ZXV \leftarrow (\boldsymbol{\mathcal{Z}}_{(n)} * \boldsymbol{\mathcal{X}}_{(n)})(\boldsymbol{V}^{(n)})^T$
    for $k \leftarrow 1$ **to** K **do**
      $ZUVV \leftarrow \boldsymbol{\mathcal{Z}}_{(n)} * (\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)})(\boldsymbol{V}^{(n)})^T$
      for $i_n \leftarrow 1$ **to** I_n **do**
        $C^+(\boldsymbol{U}_{i_n k}^{(n)}), C^-(\boldsymbol{U}_{i_n k}^{(n)}) \leftarrow$ Eq. (4.8)
        $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZXV)_{i_n k} + \lambda_n C^-(\boldsymbol{U}_{i_n k}^{(n)})}{(ZUVV)_{i_n k} + \lambda_n C^+(\boldsymbol{U}_{i_n k}^{(n)})}$

(LRNTF) is also selected as one of the comparison algorithms, which is the extension of GNMF on CP decomposition using squared Euclidean distance. At the same time, four NMF algorithms each of which runs with two distance measures separately are compared as well: NMF-MUL, GNMF, LCNMF and RPR-NMF.

Several metrics are used to evaluate the performance of the above algorithms:

1. *Mean Squared Loss (MSL)* for algorithms using squared Euclidean distance and *Mean Divergence (MD)* for algorithms using Divergence, which evaluate how

Algorithm 4.2 Relative Pairwise Relationship constrained Non-negative Tensor Factorisation using CP decomposition with Divergence (RPR-NTF-CD)

INPUT: $\boldsymbol{\mathcal{X}} \in \mathbb{R}_{\geqslant 0}^{I_1 \times \cdots \times I_N}$, $\boldsymbol{\mathcal{Z}} \in \mathbb{Z}_2^{I_1 \times \cdots \times I_N}$, $K \in \mathbb{N}^+$, $\boldsymbol{L}_{\boldsymbol{U}}^{(n)} \in \mathbb{N}^{l_{\boldsymbol{U}^{(n)}} \times 3}$, $\lambda_n \in \mathbb{R}^+$

OUTPUT: $\boldsymbol{U}^{(n)} \in \mathbb{R}_{\geqslant 0}^{I_n \times K}$

Initialise all $\boldsymbol{U}^{(n)}$ as non-negative random matrices
while Terminal conditions not satisfied **do**
  for $n \leftarrow 1$ **to** N **do**
    $\boldsymbol{\mathcal{X}}_{(n)}, \boldsymbol{\mathcal{Z}}_{(n)} \leftarrow$ Matricise $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}$
    $\boldsymbol{V}^{(n)} \leftarrow (\bigodot_{r=N;r\neq n}^{1} \boldsymbol{U}^{(r)})^T$
    $ZV \leftarrow \boldsymbol{\mathcal{Z}}_{(n)}(\boldsymbol{V}^{(n)})^T, \quad ZX \leftarrow \boldsymbol{\mathcal{Z}}_{(n)} * \boldsymbol{\mathcal{X}}_{(n)}$
    for $k \leftarrow 1$ **to** K **do**
      $UVV \leftarrow (\boldsymbol{U}^{(n)}\boldsymbol{V}^{(n)}))(\boldsymbol{V}^{(n)})^T$
      for $i_n \leftarrow 1$ **to** I_n **do**
        Calculate $P(\boldsymbol{U}_{i_n k}^{(n)})$
        if $\frac{1}{2}\lambda_n P(\boldsymbol{U}_{i_n k}^{(n)}) + (ZV)_{i_n k} < 0$ **then**
          $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZX/UVV)_{i_n k}}{(ZV)_{i_n k}}$
        else
          $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZX/UVV)_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}_{i_n k}^{(n)}) + (ZV)_{i_n k}}$
  if $\mathcal{F}_{\text{cd}}$ increases **then**
    $\lambda_n \leftarrow 1/2 \cdot \lambda_n$, Roll back
  else
    $\lambda_n \leftarrow 1.01 \cdot \lambda_n$

close the restored tensor and the factorised tensor are:

$$\text{MSL} = \frac{1}{I_1 \cdots I_N} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{R}}\|_F^2, \tag{4.39}$$

$$\text{MD} = \frac{1}{I_1 \cdots I_N} D(\boldsymbol{\mathcal{X}} || \boldsymbol{\mathcal{R}}). \tag{4.40}$$

Algorithm 4.3 Relative Pairwise Relationship constrained Non-negative Tensor Factorisation using Tucker decomposition with squared Euclidean distance (RPR-NTF-TE)

INPUT: $\mathcal{X} \in \mathbb{R}_{\geqslant 0}^{I_1 \times \cdots \times I_N}$, $\mathcal{Z} \in \mathbb{Z}_2^{I_1 \times \cdots \times I_N}$, $K \in \mathbb{N}_{>0}^N$, $\boldsymbol{L}_{\boldsymbol{U}}^{(n)} \in \mathbb{N}^{l_{\boldsymbol{U}^{(n)}} \times 3}$, $\lambda_n \in \mathbb{R}^+$

OUTPUT: $\boldsymbol{U}^{(n)} \in \mathbb{R}_{\geqslant 0}^{I_n \times K}$, $\mathcal{G} \in \mathbb{R}_{\geqslant 0}^{K_1 \times \cdots \times K_N}$

Initialise all $\boldsymbol{U}^{(n)}$ as non-negative random matrices
Initialise $\mathcal{G}$ as a non-negative tensor
while Terminal conditions not satisfied **do**
  for $n \leftarrow 1$ **to** N **do**
    $\mathcal{X}_{(n)}, \mathcal{Z}_{(n)}, \mathcal{G}_{(n)} \leftarrow$ Matricise $\mathcal{X}, \mathcal{Z}, \mathcal{G}$
    $\boldsymbol{D}^{(n)} = \mathcal{G}_{(n)}(\bigotimes_{r=N, r\neq n}^{1} \boldsymbol{U}^{(r)})^T$
    $ZXD \leftarrow (\mathcal{Z}_{(n)} * \mathcal{X}_{(n)})(\boldsymbol{D}^{(n)})^T$
    for $k \leftarrow 1$ **to** K **do**
      $ZUDD \leftarrow \mathcal{Z}_{(n)} * (\boldsymbol{U}^{(n)}\boldsymbol{D}^{(n)})(\boldsymbol{D}^{(n)})^T$
      for $i_n \leftarrow 1$ **to** I_n **do**
        Calculate $C^+(\boldsymbol{U}_{i_n k}^{(n)})$ and $C^-(\boldsymbol{U}_{i_n k}^{(n)})$
        $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZXD)_{i_n k} + \lambda_n C^-(\boldsymbol{U}_{i_n k}^{(n)})}{(ZUDD)_{i_n k} + \lambda_n C^+(\boldsymbol{U}_{i_n k}^{(n)})}$
  $\mathcal{R} \leftarrow \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \cdots \times_N \boldsymbol{U}^{(N)}$
  $\mathcal{G} \leftarrow \mathcal{G} \cdot \frac{(\mathcal{Z} * \mathcal{X}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{(\mathcal{Z} * \mathcal{R}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}$

2. *Constraint Satisfied Rate (CSR)*, which is to demonstrate how much ratio of the RPR constraints is satisfied after factorisation:

$$\text{CSR} = \frac{1}{N} \sum_{n=1}^{N} l_{\boldsymbol{U}}^{(n)*} / l_{\boldsymbol{U}}^{(n)}, \tag{4.41}$$

where $l_{\boldsymbol{U}}^{(n)*}$ is the number of satisfied constraints.

3. *Clustering Accuracy (ACC)* [66] and *Normalised Mutual Information (NMI)* [67], which are to evaluate the clustering performance for image clustering experi-

Algorithm 4.4 Relative Pairwise Relationship constrained Non-negative Tensor Factorisation using Tucker decomposition with Divergence (RPR-NTF-TD)

INPUT: $\boldsymbol{\mathcal{X}} \in \mathbb{R}_{\geqslant 0}^{I_1 \times \cdots \times I_N}$, $\boldsymbol{\mathcal{Z}} \in \mathbb{Z}_2^{I_1 \times \cdots \times I_N}$, $K \in \mathbb{N}_{>0}^N$, $\boldsymbol{L}_{\boldsymbol{U}}^{(n)} \in \mathbb{N}^{l_{\boldsymbol{U}^{(n)}} \times 3}$, $\lambda_n \in \mathbb{R}^+$

OUTPUT: $\boldsymbol{U}^{(n)} \in \mathbb{R}_{\geqslant 0}^{I_n \times K}$, $\boldsymbol{\mathcal{G}} \in \mathbb{R}_{\geqslant 0}^{K_1 \times \cdots \times K_N}$

Initialise all $\boldsymbol{U}^{(n)}$ as non-negative random matrices
Initialise $\boldsymbol{\mathcal{G}}$ as a non-negative tensor
while Terminal conditions not satisfied **do**
    for $n \leftarrow 1$ **to** N **do**
        $\boldsymbol{\mathcal{X}}_{(n)}, \boldsymbol{\mathcal{Z}}_{(n)}, \boldsymbol{\mathcal{G}}_{(n)} \leftarrow$ Matricise $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{G}}$
        $\boldsymbol{D}^{(n)} = \boldsymbol{\mathcal{G}}_{(n)} (\bigotimes_{r=N, r \neq n}^{1} \boldsymbol{U}^{(r)})^T$
        $ZD \leftarrow \boldsymbol{\mathcal{Z}}_{(n)} (\boldsymbol{D}^{(n)})^T, \quad ZX \leftarrow \boldsymbol{\mathcal{Z}}_{(n)} * \boldsymbol{\mathcal{X}}_{(n)}$
        for $k \leftarrow 1$ **to** K **do**
            $UDD \leftarrow (\boldsymbol{U}^{(n)} \boldsymbol{D}^{(n)})(\boldsymbol{D}^{(n)})^T$
            for $i_n \leftarrow 1$ **to** I_n **do**
                Calculate $P(\boldsymbol{U}_{i_n k}^{(n)})$
                if $\frac{1}{2}\lambda_n P(\boldsymbol{U}_{i_n k}^{(n)}) + (ZD)_{i_n k} < 0$ **then**
                    $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZX/UDD)_{i_n k}}{(ZD)_{i_n k}}$
                else
                    $\boldsymbol{U}_{i_n k}^{(n)} \leftarrow \boldsymbol{U}_{i_n k}^{(n)} \frac{(ZX/UDD)_{i_n k}}{\frac{1}{2}\lambda_n P(\boldsymbol{U}_{i_n k}^{(n)}) + (ZD)_{i_n k}}$
    $\boldsymbol{\mathcal{R}} \leftarrow \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \cdots \times_N \boldsymbol{U}^{(N)}$
    $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{G}} \cdot \frac{(\boldsymbol{\mathcal{Z}} * \boldsymbol{\mathcal{X}} / \boldsymbol{\mathcal{R}}) \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}{\boldsymbol{\mathcal{Z}} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 \cdots \times_N (\boldsymbol{U}^{(N)})^T}$
    if $\mathcal{F}_{\text{td}}$ increases **then**
        $\lambda_n \leftarrow 1/2 \cdot \lambda_n$, Roll back
    else
        $\lambda_n \leftarrow 1.01 \cdot \lambda_n$

ments.

4. *Root Mean Squared Error (RMSE)* [68] and *F1 score* [69], which are to evaluate the recommendation performance in recommender systems.

Two synthetic datasets are generated to validate the effectiveness and conducting complexity as well as convergence analysis. Meanwhile, two real-world datasets are used to verify the performance of RPR-NTF against other algorithms regarding image clustering and recommender systems. The statistics of these datasets are presented in Table 4.1.

Table 4.1 : Statistics of Datasets

dataset	dimensions	range	density
Synthetic 1	50 x 50 x 50	[0,1]	1
Synthetic 2	100 x 100 x 100 x 100	[0,1]	0.001
AT&T ORL	400 x 64 x 64	[0,255]	1
Movielens 1M	35 x 6040 x 3706	[1,5]	0.0012

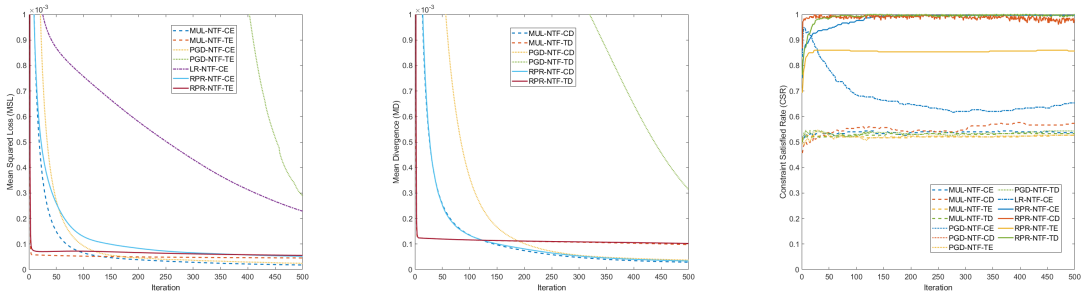
Note that the conversion algorithm proposed in Chapter 3 which converts RPRs to a weight matrix is also adopted. The latter is required in G-NMF and LR-NTF.

Since RPR-NMF has been proven to be an NMF algorithm with the lowest MSL/MD, the highest CSR and the most reasonable running time among all NMF algorithms, only the chosen NTF algorithms are compared in synthetic experiments. For real datasets, all NMF and NTF algorithms are compared.

4.5.1 Effectiveness and Convergence

An experiment is conducted on the first synthetic dataset (Synthetic 1 in Table 4.1) to demonstrate the effectiveness of the proposed RPR-NTF.

To generate the Synthetic 1 dataset as well as the RPRs, first, three random matrices are generated all with size 50×10 and a random tensor with size $10 \times 10 \times 10$. The values of their entries vary in $[0, 1]$. Then 20 RPRs are randomly extracted from each matrix. After that, the factorising tensor $\mathcal{X}$ is constructed by the reverse Tucker decomposition on the above three matrices and the tensor. For algorithms using CP decomposition, the number of components K is set to 10, while for those using Tucker decomposition, each of the three dimensions of the core tensor is set to 10 as well. The number of iterations is set 500. Other algorithm-specific parameters are set as follows: the learning rate $\lambda = 10^{-3}, 10^{-3}, 10^{-6}, 10^{-6}$ for PGD-NTF-CE, PGD-NTF-CD, PGD-NTF-TE and PGD-NTF-TD respectively, the regularisation parameter $\lambda = 100$ for LR-NTF-CE, the penalty coefficient $\lambda_n = 100, 50, 100, 50$ for RPR-NTF-CE, RPR-NTF-CD, RPR-NTF-TE and RPR-NTF-TD respectively. The above generation procedure is repeated 5 times and all NTF algorithms have been run on each of the generated tensor along with the constraints. The average results on MSL, MD and CSR are reported in Figure 4.1.



(a) (b) (c)

Figure 4.1 : Performance of NTF algorithms with respect to iterations. (a) MSL, (b) MD, (c) CSR.

From the results, several observations are worth-noting. (1) the approximation errors of RPR-NTF algorithms are very close to those of MUL-NTF algorithms.

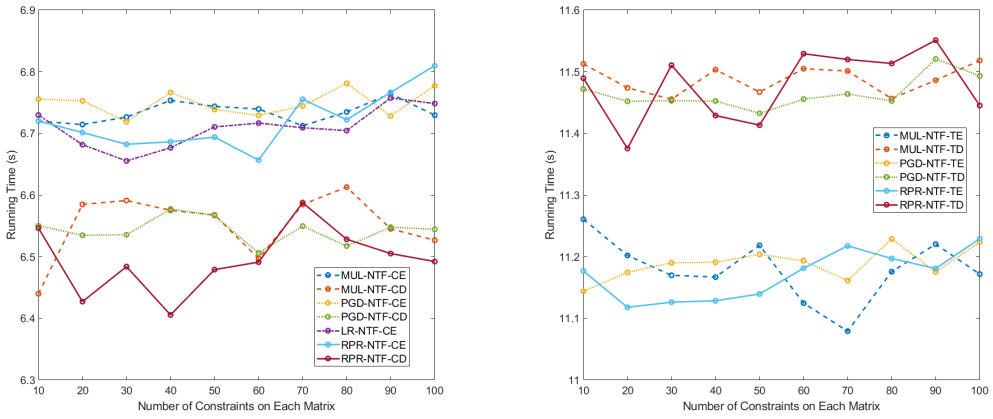
The lines of RPR-NTF-TE and MUL-NTF-TE are almost overlapping everywhere, as are those of RPR-NTF-TD and MUL-NTF-TD. There is a small gap between RPR-NTF-CE and MUL-NTF-CE, while the gap between RPR-NTF-CD and MUL-NTF-CD is quite small. (2) RPR-NTF-TE and MUL-NTF-TE converge the fastest in (a), and so do RPR-NTF-TD and MUL-NTF-TD in (b). PGD-NTF-TE and PGD-NTF-TD converge the slowest. (3) RPR-NTF-CE, MUL-NTF-CE and PGD-NTF-CE obtain the lowest MSL, while their corresponding Divergence versions obtain the lowest MD. (4) All four RPR-NTF algorithms improve CSR as the iteration increases, and achieve the best CSR performance (100%) except for RPR-NTF-TE. LR-NTF-CE satisfies a high rate of constraints at the beginning but the rate drops sharply and stabilises at around 65%. MUL-NTF and PGD-NTF algorithms have flat CSR curves between 50% and 60% which is reasonable since they consider no constraints in their objectives. This experiment demonstrates that RPR-NTF algorithms can achieve low approximation error, fast convergence and high constraint satisfied rate simultaneously.

4.5.2 Time Complexity

In each iteration, all RPR-NTF algorithms require $O\big((I_1 \cdots I_N + c\sum_n l_{\boldsymbol{U}}^{(n)})K\big)$ operations to update all factor matrices, where c is a scalar depending on the exponential and hinge calculations brought by RPRs. Experiments are conducted to test the extra consuming time on the Synthetic 2 dataset. The idea is to compare the running time of NTF algorithms in one iteration as the number of RPRs increases. The computer used in these experiments has 2x 2.60GHz Intel Xeon Gold 6126 12 cores CPUs, 192G RAM and an operating system of Ubuntu 18.04.

The generation process is similar to that adopted in the first experiment, but the factorising tensor $\boldsymbol{\mathcal{X}}$ is just a randomly generated sparse tensor instead of the resulting tensor of reverse Tucker decomposition. The number of constraints varies

from 10 to 100 with step 10 on each factor matrices (thus 40 to 400 constraints in total). 5 different $\mathcal{X}$s are generated, and for each $\mathcal{X}$, 10 groups of constraints are generated. All NTF algorithms are conducted 50 times on different $\mathcal{X}$s and on different groups of constraints. Other algorithm-specific parameters are almost the same as before. Only the learning rate λ are 10^{-8} instead of 10^{-6} for both PGD-NTF-TE and PGD-NTF-TD. The average results are reported in Figure 4.2.



(a) (b)

Figure 4.2 : Running time of NTF algorithms with respect to number of constraints. (a) CP algorithms, (b) Tucker algorithms.

The running time of algorithms using CP and Tucker decomposition are presented separately in Figure 4.2 (a) and (b). When using CP decomposition, we see from (a) that the running time of algorithms using Divergence is faster than that of algorithms using squared Euclidean distance. However, when using Tucker decomposition, the case is just opposite, i.e. algorithms using squared Euclidean distance are faster. The curves of constrained algorithms go up in general as the number of constraints becomes larger, but the increase in running time is quite small compared to the increase in the number of constraints. For example, the running time of RPR-NTF-CE in (a) is 6.72s with 10 constraints and 6.81s with 100 constraints. As

the number of constraints becomes 10 times larger, the running time only increases by 0.09s which is about 1.3% of the whole running time of an iteration. Meanwhile, we notice that the curves of non-constrained algorithms fluctuate as well, meaning that the influence in running time brought by extra constraints is occluded by even the variations of calculation time with different initiations.

4.5.3 Image Clustering on AT & T ORL

The AT&T ORL dataset [70] is a widely used image dataset for validation of algorithms in many applications. It contains 400 images taken from 40 people. Each person has 10 different facial images. Instead of using the 32×32 version of this dataset in RPR-NMF, the 64×64 version is chosen here, i.e. each image is represented by a 64×64 matrix with entries varying in $[0, 255]$. For NTF algorithms, the input data is a $400 \times 64 \times 64$ tensor, and for NMF algorithms, it is a 400×4096 matrix where each image is vectorised as a row.

First, random K people's images are chosen as the input data. Then 3 images for each person are randomly picked to construct the RPRs. There are two types of constraints: intro-class and inter-class RPRs. For each intro-class RPR, the first two elements come from the images of the same person, and the third element comes from an arbitrary image of the other person. For each inter-class RPR, the first element is picked randomly from a person, and the second element is the first one's nearest neighbour in other groups, while the third one is the second nearest neighbour. The number of RPRs is $9K$ in which $6K$ are the intro-class and $3K$ are the inter-class. Note that the constraints are only placed on one of the factor matrices corresponding to the dimension 400. The number of class K is set to $5, 10, 20, 30, 40$. The learning rate λ of PGD-NTF algorithms are tuned between 10^{-3} and 10^{-9} for their best performance, the regularisation parameter λ is set to 100 for G-NMF-E, G-NMF-D and LR-NTF-CE, the penalty coefficient λ is set to $100, 10$ for RPR-NMF-E,

RPR-NMF-D and λ_n is set to $500, 10, 100, 10$ RPR-NTF-CE, RPR-NTF-CD, RPR-NTF-TE and RPR-NTF-TD respectively. The experiments are repeated 5 times and the average results are presented in Table 4.2. The highest ACC, NMI under each settings (column) are highlighted with bold numbers.

Table 4.2 : Clustering Performance on AT&T ORL of NMF and NTF Algorithms

	ACC (%)						NMI (%)					
K	5	10	20	30	40	Avg.	5	10	20	30	40	Avg.
l	45	90	180	270	360		45	90	180	270	360	
MUL-NMF-E	75.92	69.68	61.88	57.49	56.47	64.29	76.65	75.95	75.53	74.84	74.90	75.57
MUL-NMF-D	80.72	68.76	61.20	57.45	55.12	64.65	80.50	75.44	75.20	74.19	74.22	75.91
G-NMF-E	57.84	51.28	43.82	39.37	39.04	46.27	44.52	48.57	48.53	48.10	48.89	47.72
G-NMF-D	62.16	49.40	44.06	39.07	39.19	46.78	48.66	47.50	48.84	48.05	49.51	48.51
LC-NMF-E	59.28	50.24	45.86	40.27	39.02	46.93	47.11	48.89	52.95	50.82	51.85	50.32
LC-NMF-D	62.08	50.40	44.90	39.67	38.81	47.17	47.77	48.85	52.06	49.99	51.01	49.94
RPR-NMF-E	80.88	69.00	65.50	58.05	56.17	65.92	82.08	76.44	78.90	75.29	74.98	77.54
RPR-NMF-D	77.20	69.72	62.84	56.89	56.02	64.53	78.16	76.16	76.23	74.11	74.35	75.80
MUL-NTF-CE	71.20	63.88	62.62	54.40	53.73	61.17	72.34	72.57	74.84	71.87	72.63	72.85
MUL-NTF-CD	71.20	64.80	59.20	53.51	52.41	60.22	69.50	71.24	72.57	70.96	71.53	71.16
MUL-NTF-TE	68.40	50.68	40.40	31.87	29.72	44.21	69.23	56.80	55.30	51.63	52.85	57.16
MUL-NTF-TD	70.96	57.44	42.08	37.11	32.04	47.93	71.58	62.48	56.14	56.29	54.83	60.26
PGD-NTF-CE	64.48	62.56	49.18	51.89	44.76	54.57	64.44	68.88	62.56	69.54	64.48	65.98
PGD-NTF-CD	69.68	66.32	57.00	35.52	19.72	49.65	71.46	71.83	70.62	53.66	41.81	61.88
PGD-NTF-TE	39.44	24.92	19.24	17.16	15.99	23.35	20.21	21.51	30.01	35.16	38.95	29.17
PGD-NTF-TD	64.16	26.24	19.48	17.11	16.21	28.64	60.93	23.27	30.40	34.60	37.91	37.42
LR-NTF-CE	75.36	63.00	56.70	46.93	44.47	57.29	76.72	69.31	68.56	63.32	63.78	68.34
RPR-NTF-CE	**84.40**	**71.96**	**67.62**	**60.55**	**57.27**	**68.36**	**86.96**	**77.16**	**79.02**	**75.80**	**75.26**	**78.84**
RPR-NTF-CD	71.04	63.48	62.36	54.35	51.67	60.58	70.53	72.08	74.86	71.25	70.61	71.87
RPR-NTF-TE	65.92	56.00	42.02	33.16	30.87	45.59	68.68	61.42	56.04	52.66	53.79	58.52
RPR-NTF-TD	70.80	52.48	45.90	37.00	32.62	47.76	71.35	57.83	59.63	56.19	55.46	60.09

We can see from Table 4.2 that: RPR-NTF-CE achieves the best performance among all tested algorithms under all settings. Its average ACC and NMI outperform the second highest one RPR-NMF-E by 2.44% and 1.30% respectively. The greatest improvement is 3.52% on ACC and 4.88% on NMI which happens when

$K = 5$. Overall, this experiment verifies the improvements on the task of image clustering from RPR-NTF algorithms which can achieve low approximation errors, high constraint satisfied rate and better clustering performance at the same time.

4.5.4 Recommendations on Movielens 1M

The Movielens datasets are typically used to validate the performance of algorithms on recommender systems [73]. The Movielens 1M dataset is chosen here, because the datasets released afterwards do not have any meta information about users which is needed to extract constraints. The dataset was collected from 6040 users on 3883 movies over 35 months. $1,000,209$ ratings were recorded, each varying from 1 to 5. After removing movies with no ratings, rating tensor with size $35 \times 6040 \times 3706$ is obtained.

The RPRs are extracted from the meta information of users and movies provided by this dataset. Each user is encoded by a vector of length 23 (1 for gender, 1 for age and 21 for occupation), and each movie is encoded by a vector of length 18 (18 different genres). The RPRs are extracted randomly based on the calculation of distance between pairwise users/movies. Note that NMF algorithms are conducted on each month's data (each slice of the input tensor over the dimension 35) and their restored tensor is the integration of all resultant matrices. The CSR of NMF algorithms is the average value of all CSRs on slices, while their RMSE and F1 Score are calculated using the integrated tensors. Furthermore, G-NMF-D is not available in this experiment because it has no implementation for incomplete data [22].

The latent dimension K is set to $5, 10, 20$. The number of constraints on user and movie dimensions are correspondingly set to $300, 600, 900$. For PGD-NTF algorithms, λ is tuned between 10^{-3} and 10^{-9}. For G-NMF-E and LR-NTF-CE, λ is set 100. For RPR-NMFs, λ is set to $100, 10$. For RPR-NTFs, λ_n is set to $500, 10, 100, 10$. The cross validation results of all constrained algorithms are showed in Table 4.3.

Table 4.3 : Recommendation Performance on Movielens 1M of Constrained NMF and NTF Algorithms

	RMSE				F1 Score			
K	5	10	20	Avg.	5	10	20	Avg.
l	300	600	900		300	600	900	
G-NMF-E	1.0790	1.1140	1.2610	1.1510	64.83	63.90	62.99	63.91
LC-NMF-E	1.1730	1.1810	1.1900	1.1810	64.79	63.70	62.69	63.73
LC-NMF-D	1.3430	1.1750	1.2220	1.2470	64.74	63.44	62.62	63.60
RPR-NMF-E	1.8130	1.3490	1.3500	1.5040	66.33	65.30	64.79	65.47
RPR-NMF-D	1.9980	2.7710	7.0940	3.9540	65.78	64.81	63.54	64.71
LR-NTF-CE	1.3440	1.1970	1.2300	1.2570	41.58	48.08	45.42	45.03
RPR-NTF-CE	0.8831	**0.8769**	0.8950	**0.8850**	70.00	70.08	**69.75**	**69.94**
RPR-NTF-CD	**0.8785**	0.8840	0.9263	0.8963	**70.15**	**70.22**	69.34	69.90
RPR-NTF-TE	0.8953	0.8901	**0.8888**	0.8914	69.06	69.23	69.71	69.33
RPR-NTF-TD	0.8886	0.8975	0.9000	0.8954	69.45	68.97	69.34	69.25

From the results we can see that RPR-NTF algorithms have better recommendation performance than others. Specifically, RPR-NTF-CE achieves the lowest average RMSE with 23.11% improvement $((1.1510 - 0.8850)/1.1510)$ and the highest average F1 Score with 4.47% improvement $(69.94 - 65.47)$ compared to the second excluding other RPR-NTF algorithms. The best RMSE is obtained by RPR-NTF-CE and the best F1 Score is obtained by RPR-NTF-CD when $K = 10$. This experiment shows the superiority of the proposed RPR-NTF on recommender system applications.

4.6 Summary

In this chapter, a novel TF algorithm RPR-NTF is proposed to keep expected relationship unchanged after decomposition by placing RPR constraints in objectives. It considers two TF schemes, CP decomposition and Tucker decomposition, and two commonly used distance measures, squared Euclidean distance and Divergence. Apart from conforming to MUL, a conversion approach is proposed to convert the updating formulas regarding each entry to equivalent ones with respect to factor matrices (and core tensor), which involve only matrix operations and make the computation much more efficient. Furthermore, a variation of RPR-NTF is proposed to cater for incomplete input data with an additional indicator tensor provided. Synthetic experiments demonstrate the effectiveness of imposing RPR constraints in NTF which causes little increase in running time. Finally, numerous comparison experiments of thirteen NTF algorithms and eight NMF algorithms on real-world datasets show that the proposed RPR-NTF algorithms are superior to other factorisation algorithms, and are able to achieve simultaneously low approximation error, high constraint satisfied rate and better application-specific performance.

Chapter 5

Magnitude Bounded Matrix Factorisation

In this chapter, a novel MF algorithm called MBMF is introduced to deal with the prediction fluctuation for users/item in recommender systems. First, the magnitude bounded matrix factorisation problem is defined, followed by conversions from the constrained task to an unconstrained one. After that, a stochastic gradient descent method is applied to solve the unconstrained task. The content of this Chapter is from a published IEEE journal paper [74] and the copyright/credit notice is placed in the reference.

5.1 MBMF Problem Definition

Given an observation matrix $\boldsymbol{V} \in \mathbb{R}^{N\times M}$, the original low rank matrix factorisation task is to find two factor matrices $\boldsymbol{W} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{H} \in \mathbb{R}^{K\times M}$ which satisfy the approximation $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$, where K is commonly referred to as the latent dimension and usually set a smaller value than N and M. According to the definition of matrix multiplication and the geometric interpretation of vector inner product in Euclidean space, each element of $\boldsymbol{V}$, say $\boldsymbol{V}_{ij}$, is approximated by the inner product of the i^{th} row of $\boldsymbol{W}$ and the j^{th} column of $\boldsymbol{H}$:

$$\boldsymbol{V}_{ij} \approx \hat{\boldsymbol{V}}_{ij} = \boldsymbol{W}_{i:} \cdot \boldsymbol{H}_{:j} = |\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}| \cos \alpha_{ij} \tag{5.1}$$

where $\boldsymbol{W}_{i:}$ denotes the i^{th} row of $\boldsymbol{W}$ (so does $\boldsymbol{H}_{:j}$), $|\cdot|$ denotes the vector magnitude and α_{ij} is the angle between vectors $\boldsymbol{W}_{i:}$ and $\boldsymbol{H}_{:j}$. As $\cos \alpha_{ij} \in [-1, 1]$, the prediction entry $\hat{\boldsymbol{V}}_{ij}$ is bounded by the product of magnitudes:

$$-|\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}| \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant |\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}|. \tag{5.2}$$

Definition 5.1 (Magnitude Bounded Matrix Factorisation Problem)**.** *Given an observation matrix* $\boldsymbol{V} \in \mathbb{R}^{N\times M}$ *and two constant positive magnitude vectors* $\boldsymbol{r^W} \in \mathbb{R}_+^N$ *and* $\boldsymbol{r^H} \in \mathbb{R}_+^M$*, magnitude bounded matrix factorisation aims to find two factor matrices* $\boldsymbol{W}$ *and* $\boldsymbol{H}$ *such that*

$$\begin{aligned} \boldsymbol{V} &\approx \hat{\boldsymbol{V}} = \boldsymbol{W}\boldsymbol{H} \\ s.t. \quad |\boldsymbol{W}_{i:}| &= \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ |\boldsymbol{H}_{:j}| &= \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M. \end{aligned} \tag{5.3}$$

According to the above definition, the prediction entry $\hat{\boldsymbol{V}}_{ij}$ is now bounded by the product of its two corresponding magnitudes:

$$-\boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}} \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}}. \tag{5.4}$$

In fact, the two magnitude vectors can form a rank one range matrix $\boldsymbol{R} = \boldsymbol{r^W}(\boldsymbol{r^H})^T \in \mathbb{R}^{N\times M}$. Thus the prediction entry is bounded by its corresponding entry in the range matrix $\boldsymbol{R}$:

$$-\boldsymbol{R}_{ij} \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{R}_{ij}. \tag{5.5}$$

Notice that all of the entries in the product matrix can be bounded within individually different ranges which are determined by the give constant magnitude vectors, though the ranges are not totally independent.

According to the definition of MBMF, no matter how few observations a user/item has, its corresponding predictions are always bounded as long as the magnitude vectors are set properly. Assume that the magnitude vectors can reflect a relatively stable preference of each user or item for the current factorisation task. So the magnitudes can also be regarded as user/item preferences in recommender systems.

5.2 Conversions from Constrained to Unconstrained Task

The magnitude bounded matrix factorisation defined in Eq. (5.3) is equal to a constrained optimisation problem:

$$\begin{aligned} &\min_{\boldsymbol{W},\boldsymbol{H}} \|\boldsymbol{Z} * (\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H})\|_F^2 \\ s.t. \quad &|\boldsymbol{W}_{i:}| = \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ &|\boldsymbol{H}_{:j}| = \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M. \end{aligned} \tag{5.6}$$

where $\boldsymbol{Z}$ is an indicator matrix in which ones denote the existing observations and zeros otherwise. It is not straightforward to work it out because of the magnitude constraints.

Let us now imagine the geometric representation of the magnitude constraints first. Given a two-dimensional variable $\boldsymbol{x} = [x_1, x_2]^T$ in Euclidean space, setting its magnitude to a constant $|\boldsymbol{x}| = r$ limits the possible space of $\boldsymbol{x}$ onto a circle with radius r. By introducing an angle variable ϕ, the vector can then be converted into the Polar coordinate system as $\boldsymbol{x} = [r\cos\phi, r\sin\phi]^T$, which embeds the magnitude constraint into the Polar representation such that calculations on the converted variable do not need to consider the magnitude constraint. Similarly, for higher dimensional variables, the magnitude constraints limit their possible space onto a sphere or hypersphere in the Euclidean system, and conversion to spherical coordinates will incorporate the magnitudes into their new representations, which "removes" the magnitude constraints.

Thus here introduces two angle matrices $\boldsymbol{\Phi} \in \mathbb{R}^{N\times(K-1)}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{(K-1)\times M}$ to help represent each row of $\boldsymbol{W}$ and each column of $\boldsymbol{H}$ in spherical space with the radii set as corresponding magnitudes. A n-dimensional variable $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ can be represented by a radial coordinate r and $n-1$ angular coordinates $\boldsymbol{\phi} =$

$[\phi_1, \phi_2, \cdots, \phi_{n-1}]^T$ where $\phi_1, \phi_2, \cdots, \phi_{n-2} \in [0, \pi]$ and $\phi_{n-1} \in [0, 2\pi)$:

$$\begin{aligned} x_1 &= r\cos\phi_1 \\ x_2 &= r\sin\phi_1\cos\phi_2 \\ &\vdots \\ x_{n-1} &= r\sin\phi_1\cdots\sin\phi_{n-2}\cos\phi_{n-1} \\ x_n &= r\sin\phi_1\cdots\sin\phi_{n-2}\sin\phi_{n-1}. \end{aligned} \tag{5.7}$$

Note that the constraints on these angles are to ensure the uniqueness of conversion from cartesian to spherical, however, MBMF does the conversion the other way around, i.e. from spherical to cartesian which does not have any angle constraints.

Define auxiliary vectors of functions $\boldsymbol{s}(\boldsymbol{\phi}) = [s_1(\boldsymbol{\phi}), s_2(\boldsymbol{\phi}), \cdots, s_n(\boldsymbol{\phi})]^T$ and $\boldsymbol{c}(\boldsymbol{\phi}) = [c_1(\boldsymbol{\phi}), c_2(\boldsymbol{\phi}), \cdots, c_n(\boldsymbol{\phi})]^T$ such that

$$s_i(\boldsymbol{\phi}) = \begin{cases} 1, & i = 1 \\ \prod_{j=1}^{i-1}\sin\phi_j, & i > 1 \end{cases}, \tag{5.8}$$

$$c_i(\boldsymbol{\phi}) = \begin{cases} \cos\phi_i, & i < n \\ 1, & i = n \end{cases}, \tag{5.9}$$

then

$$x_i = r s_i(\boldsymbol{\phi}) c_i(\boldsymbol{\phi}), \tag{5.10}$$

thus

$$\boldsymbol{x} = r\boldsymbol{s}(\boldsymbol{\phi}) * \boldsymbol{c}(\boldsymbol{\phi}). \tag{5.11}$$

Then both of the factor matrices can be converted into the spherical space:

$$\boldsymbol{W}_{i:} = \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{i:}) * \boldsymbol{c}(\boldsymbol{\Phi}_{i:}), \tag{5.12}$$

$$\boldsymbol{H}_{:j} = \boldsymbol{r}_j^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:j}) * \boldsymbol{c}(\boldsymbol{\Theta}_{:j}), \tag{5.13}$$

Define auxiliary matrices of functions $\boldsymbol{S}(\boldsymbol{\Phi})$, $\boldsymbol{S}(\boldsymbol{\Theta})$ such that

$$\boldsymbol{S}_{i:}(\boldsymbol{\Phi}) = \boldsymbol{s}(\boldsymbol{\Phi}_{i:}), \quad \boldsymbol{S}_{:j}(\boldsymbol{\Theta}) = \boldsymbol{s}(\boldsymbol{\Theta}_{:j}), \tag{5.14}$$

and $\boldsymbol{C}(\boldsymbol{\Phi})$, $\boldsymbol{C}(\boldsymbol{\Theta})$ such that

$$\boldsymbol{C}_{i:}(\boldsymbol{\Phi}) = \boldsymbol{c}(\boldsymbol{\Phi}_{i:}), \quad \boldsymbol{C}_{:j}(\boldsymbol{\Theta}) = \boldsymbol{c}(\boldsymbol{\Theta}_{:j}), \tag{5.15}$$

then

$$\boldsymbol{W} = \boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}), \tag{5.16}$$

$$\boldsymbol{H} = \boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta}), \tag{5.17}$$

where $\odot$ denotes the Khatri-Rao product. Notice that some transpose signs are omitted in the formulas from Eq. (5.12) to (5.17) to make them concise and consistent. This kind of conversion is referred as *Spherical2Cartesian*.

Thus, the constrained optimisation problem in Eq. (5.6) is equivalent to the following unconstrained optimisation task:

$$\min_{\boldsymbol{\Phi},\boldsymbol{\Theta}} \|\boldsymbol{Z} * (\boldsymbol{V} - (\boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}))(\boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta})))\|_F^2. \tag{5.18}$$

5.3 Solving Unconstrained Optimisation

The objective functions defined in Eq. (5.6) and Eq. (5.18) both are non-convex. Fortunately, it is found feasible to be solved efficiently by the SGD method [75] experimentally. The convergence properties of SGD for non-convex objective functions are out of the scope of this chapter. According to Eq. (5.18), the objective function is defined as

$$\mathcal{F} = \|\boldsymbol{Z} * (\boldsymbol{V} - (\boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}))(\boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta})))\|_F^2. \tag{5.19}$$

First, calculate the partial derivatives of $\mathcal{F}$ with respect to a single element, say $\boldsymbol{\Phi}_{ab}$. Here $\boldsymbol{W}$ is regarded as a function of $\boldsymbol{\Phi}$ (so does $\boldsymbol{H}$ to $\boldsymbol{\Theta}$). Thus by chain rule,

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}_{ab}} = \sum_{j=1}^{K} \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{aj}} \frac{\partial \boldsymbol{W}_{aj}}{\partial \boldsymbol{\Phi}_{ab}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{a:}} (\frac{\partial \boldsymbol{W}_{a:}}{\partial \boldsymbol{\Phi}_{ab}})^T, \tag{5.20}$$

where the summation is due to that $\boldsymbol{\Phi}_{ab}$ participants in the representation of all elements of vector $\boldsymbol{W}_{a:}$.

The partial derivative of $\mathcal{F}$ with respect to $\boldsymbol{W}_{aj}$ is

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{aj}} = 2((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{aj}. \tag{5.21}$$

The partial derivative of $\boldsymbol{W}_{aj}$ with respect to $\boldsymbol{\Phi}_{ab}$ is

$$\frac{\partial \boldsymbol{W}_{aj}}{\partial \boldsymbol{\Phi}_{ab}} = \boldsymbol{r}_a^{\boldsymbol{W}} \cdot \begin{cases} 0, & j < b \\ -\prod_{p=1}^{j} \sin \boldsymbol{\Phi}_{ap}, & j = b \\ \cos \boldsymbol{\Phi}_{ab} \cos \boldsymbol{\Phi}_{aj} \prod_{p=1;p\neq b}^{j-1} \sin \boldsymbol{\Phi}_{ap}, & b < j < K \\ \cos \boldsymbol{\Phi}_{ab} \prod_{p=1;p\neq b}^{j-1} \sin \boldsymbol{\Phi}_{ap}, & j = K \end{cases}. \tag{5.22}$$

Define auxiliary vectors of functions $\boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) = [s_1(\boldsymbol{\Phi}_{a:}, b), \cdots, s_K(\boldsymbol{\Phi}_{a:}, b)]^T$ and $\boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b) = [c_1(\boldsymbol{\Phi}_{a:}, b), \quad \cdots, c_K(\boldsymbol{\Phi}_{a:}, b)]^T$ such that

$$s_j(\boldsymbol{\Phi}_{a:}, b) = \prod_{p=1}^{b-1} \sin \boldsymbol{\Phi}_{ap} \cdot \begin{cases} 0, & j < b \\ -\sin \boldsymbol{\Phi}_{aj}, & j = b \\ 1, & j = b+1 \\ \prod_{p=b+1}^{j-1} \sin \boldsymbol{\Phi}_{aj}, & b+1 < j \leqslant K \end{cases}, \tag{5.23}$$

$$c_j(\boldsymbol{\Phi}_{a:}, b) = \begin{cases} 0, & j < b \\ 1, & j = b \\ \cos \boldsymbol{\Phi}_{aj} \cos \boldsymbol{\Phi}_{ab}, & b < j < K \\ \cos \boldsymbol{\Phi}_{ab}, & j = K \end{cases}, \tag{5.24}$$

then

$$\frac{\partial \boldsymbol{W}_{a:}}{\partial \boldsymbol{\Phi}_{ab}} = (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b))^T, \tag{5.25}$$

thus

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}_{ab}} =& 2((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{a:} \\ & \cdot (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b)). \end{aligned} \tag{5.26}$$

Similarly the derivative of $\mathcal{F}$ with respective to $\boldsymbol{\Theta}_{ab}$ is

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Theta}_{ab}} =& 2((\boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V}))_{:b})^T \\ & \cdot (\boldsymbol{r}_b^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:b}, a) * \boldsymbol{c}(\boldsymbol{\Theta}_{:b}, a)).\end{aligned} \tag{5.27}$$

So the updating rules of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ using stochastic gradient descent are

$$\begin{aligned}\boldsymbol{\Phi}_{ab} \leftarrow \boldsymbol{\Phi}_{ab} - & \lambda^{\boldsymbol{\Phi}}((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{a:} \\ & \cdot (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b)),\end{aligned} \tag{5.28}$$

$$\begin{aligned}\boldsymbol{\Theta}_{ab} \leftarrow \boldsymbol{\Theta}_{ab} - & \lambda^{\boldsymbol{\Theta}}((\boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V}))_{:b})^T \\ & \cdot (\boldsymbol{r}_b^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:b}, a) * \boldsymbol{c}(\boldsymbol{\Theta}_{:b}, a)),\end{aligned} \tag{5.29}$$

where $\lambda^{\boldsymbol{\Phi}}$ and $\lambda^{\boldsymbol{\Theta}}$ are the learning rate parameters.

5.4 Acceleration for MBMF

According to the experiments, it is inefficient to individually update each element of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ by Eq. (5.28) and (5.29). A feasible solution is to update the whole matrices at the same time as long as the derivatives of $\mathcal{F}$ with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ can be computed directly.

Referring to Eq. (5.20), the derivative of $\mathcal{F}$ with respect to a certain element is a result of a dot product. That is, the derivative of $\mathcal{F}$ with respect to $\boldsymbol{\Phi}$ is

$$D^{\boldsymbol{\Phi}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{1:}}(\frac{\partial \boldsymbol{W}_{1:}}{\partial \boldsymbol{\Phi}_{11}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{1:}}(\frac{\partial \boldsymbol{W}_{1:}}{\partial \boldsymbol{\Phi}_{1(K-1)}})^T \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{2:}}(\frac{\partial \boldsymbol{W}_{2:}}{\partial \boldsymbol{\Phi}_{21}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{2:}}(\frac{\partial \boldsymbol{W}_{2:}}{\partial \boldsymbol{\Phi}_{2(K-1)}})^T \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{N:}}(\frac{\partial \boldsymbol{W}_{N:}}{\partial \boldsymbol{\Phi}_{n1}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{N:}}(\frac{\partial \boldsymbol{W}_{N:}}{\partial \boldsymbol{\Phi}_{n(K-1)}})^T \end{bmatrix}. \tag{5.30}$$

Notice that the k^{th} column of $D^{\boldsymbol{\Phi}}$ is a vector of dot products of two matrices—$\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}}$ and $\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\Phi}_{:k}}$ on the row dimension. Thus $D^{\boldsymbol{\Phi}}$ can be computed in the following way:

1) calculate the derivative of $\boldsymbol{W}$ with respect to $\boldsymbol{\Phi}$, which is a $N \times K \times (K-1)$ three-way tensor (indeed it is a four-way tensor with a singleton dimension);

2) duplicate $\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}}$, a $N \times K$ matrix $(K-1)$ times to form the other $N \times K \times (K-1)$ three-way tensor;

3) $D^{\boldsymbol{\Phi}}$ is then the resultant matrix of dot products of the two tensors on the second dimension (dot products of corresponding mode-2/row fibres).

The tensor in 2) is straightforward but not in 1). According to Eq. (5.22), the derivative of $\boldsymbol{W}$ with respect to the a^{th} row of $\boldsymbol{\Phi}$ is a $(K-1) \times K$ matrix which is nearly upper triangular, and the derivative of $\boldsymbol{W}$ with respect to the b^{th} column of $\boldsymbol{\Phi}$ is a $N \times K$ matrix where each row shares the same format. Thus, the tensor in 1) can be constructed by calculating the derivatives of $\boldsymbol{W}$ with respect to each column of $\boldsymbol{\Phi}$ and integrate them together.

In the algorithm of MBMF, the calculation of the derivative of $\boldsymbol{W}$ with respect to $\boldsymbol{\Phi}$ is referred as *GradWwrtPhi*.

The derivative of $\mathcal{F}$ with respect to $\boldsymbol{\Theta}$ can be obtained in a similar way, and the calculation of the derivative of $\boldsymbol{H}$ with respect to $\boldsymbol{\Theta}$ is referred as *GradHwrtTheta*. Now the updating rules of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ become

$$\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} D^{\boldsymbol{\Phi}}, \tag{5.31}$$

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} D^{\boldsymbol{\Theta}}. \tag{5.32}$$

According to the conversion from the spherical to cartesian coordinate systems (see *Spherical2Cartesian*), $\boldsymbol{W}$ and $\boldsymbol{H}$ then can be easily updated with the renewed $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$.

5.5 MBMF Algorithm and Its Variant

MBMF can be extended to versions catering for different applicant scenarios. In this section, apart from the original MBMF algorithm demonstrated in the previous section, a MBMF variant with additional non-negative constraints is introduced, followed by the discussion on parameter settings.

5.5.1 Original MBMF

The complete algorithm of the original MBMF is shown in Algorithm 5.1.

Algorithm 5.1 Magnitude Bounded Matrix Factorisation (MBMF)

INPUT: $\boldsymbol{V} \in \mathbb{R}^{N\times M}$, $\boldsymbol{Z} \in \mathbb{R}^{N\times M}$, $K \in \mathbb{N}^{+}$, $\boldsymbol{r^W}, \boldsymbol{r^H}$, $\lambda_{\boldsymbol{\Phi}} \in \mathbb{R}^{+}, \lambda_{\boldsymbol{\Theta}} \in \mathbb{R}^{+}$

OUTPUT: $\boldsymbol{W} \in \mathbb{R}^{N\times K}$, $\boldsymbol{H} \in \mathbb{R}^{K\times M}$

Initialise $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ as random angle matrices
$\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
$\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$
while Terminal conditions not satisfied **do**
    GradW $\leftarrow (\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T$
    GradW $\leftarrow$ duplicate(GradW, $K-1$)
    GradPhi $\leftarrow$ *GradWwrtPhi*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} \cdot$ dot(GradW, GradPhi, 2)
    GradH $\leftarrow \boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V})$
    GradH $\leftarrow$ duplicate(GradH, $K-1$)
    GradTheta $\leftarrow$ *GradHwrtTheta*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} \cdot$ dot(GradH, GradTheta, 1)
    $\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
    $\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$

5.5.2 Non-negative MBMF

MBMF can be easily modified to deal with non-negative data with additionally imposed non-negative constraints. Define non-negative MBMF problem as

$$\begin{aligned} \boldsymbol{V} &\approx \hat{\boldsymbol{V}} = \boldsymbol{W}\boldsymbol{H} \\ s.t. \quad |\boldsymbol{W}_{i:}| &= \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ |\boldsymbol{H}_{:j}| &= \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M, \\ \boldsymbol{W} &\geqslant 0, \quad \boldsymbol{H} \geqslant 0. \end{aligned} \tag{5.33}$$

Because of the non-negativity in both factor matrices, $\cos \alpha_{ij} > 0$ for the angle α_{ij} between any feature vectors $\boldsymbol{W}_{i:}$ and $\boldsymbol{H}_{:j}$. Thus,

$$0 \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}}. \tag{5.34}$$

The above problem can be solved by combining the coordinate conversions used in MBMF and projected gradient descent (PGD). PGD is a variation of SGD. It updates variables according to SGD formulas and projects updated results which fail to satisfy imposed constraints to the constraint boundary. After the conversion from cartesian to spherical coordinate system, the magnitude constraints are removed and only the non-negative constraints are left. Therefore, one can update Φ and Θ according to Eq. (5.31) and (5.32), convert them back to $\boldsymbol{W}$ and $\boldsymbol{H}$ by Spherical2Cartesian and then simply project all negative values in $\boldsymbol{W}$ and $\boldsymbol{H}$ to 0. MBMF with additional non-negative constraints is referred as Non-negative MBMF (MBMF-N) which is shown in Algorithm 5.2.

5.5.3 Parameter Settings

The parameters of MBMF are the learning coefficients $\lambda_{\boldsymbol{\Phi}}$ and $\lambda_{\boldsymbol{\Theta}}$ which are introduced by the SGD method. Among literature about SGD, the learning coefficients or the step sizes can be set in three different ways: (1) set the steps the value

Algorithm 5.2 Non-negative Magnitude Bounded Matrix Factorisation (MBMF-N)

INPUT: $\boldsymbol{V} \in \mathbb{R}_{\geqslant 0}^{N \times M}$, $\boldsymbol{Z} \in \mathbb{R}^{N \times M}$, $K \in \mathbb{N}^{+}$, $\boldsymbol{r}^{\boldsymbol{W}}, \boldsymbol{r}^{\boldsymbol{H}}$, $\lambda_{\boldsymbol{\Phi}} \in \mathbb{R}^{+}, \lambda_{\boldsymbol{\Theta}} \in \mathbb{R}^{+}$

OUTPUT: $\boldsymbol{W} \in \mathbb{R}_{\geqslant 0}^{N \times K}$, $\boldsymbol{H} \in \mathbb{R}_{\geqslant 0}^{K \times M}$

Initialise $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ as random angle matrices

$\boldsymbol{W} \leftarrow \textit{Spherical2Cartesian}(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$

$\boldsymbol{H} \leftarrow \textit{Spherical2Cartesian}(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$

while Terminal conditions not satisfied **do**

    $\text{GradW} \leftarrow (\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^{T} - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^{T}$

    $\text{GradW} \leftarrow \text{duplicate}(\text{GradW}, K - 1)$

    $\text{GradPhi} \leftarrow \textit{GradWwrtPhi}(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$

    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} \cdot \text{dot}(\text{GradW}, \text{GradPhi}, 2)$

    $\text{GradH} \leftarrow \boldsymbol{W}^{T}(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^{T}(\boldsymbol{Z} * \boldsymbol{V})$

    $\text{GradH} \leftarrow \text{duplicate}(\text{GradH}, K - 1)$

    $\text{GradTheta} \leftarrow \textit{GradHwrtTheta}(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$

    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} \cdot \text{dot}(\text{GradH}, \text{GradTheta}, 1)$

    $\boldsymbol{W} \leftarrow \textit{Spherical2Cartesian}(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$

    $\boldsymbol{H} \leftarrow \textit{Spherical2Cartesian}(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$

    $\boldsymbol{W}(\boldsymbol{W} < 0) \leftarrow 0, \quad \boldsymbol{H}(\boldsymbol{H} < 0) \leftarrow 0$

that produce the best performance after several tries; (2) change the steps dynamically during the updating procedure by checking the change of the value of objective function: if the function value decreases, make the steps a bit bigger (e.g. multiplied by 1.1), otherwise shrink them (e.g. divided by 2) and roll back; (3) apply a line search method (e.g. bisection search or backtracking search with Wolfe conditions) to determine more reasonable steps. The dynamic setting for parameters is used in the implementation, and the initial values for both parameters are set at 0.1.

5.6 Preliminaries of Data

5.6.1 Data Centring

Eq. (5.3) defines MBMF as an algorithm bounding each entry of the product matrix in a centrosymmetric range with respect to the origin point zero. It requires the input matrix to be centred. To centre a data matrix with a pair of pre-defined bounds $[r_{\min}, r_{\max}]$, one only needs to shift all the observed values by

$$\boldsymbol{V}_{ij}^{C} = \boldsymbol{V}_{ij} - \frac{r_{\min} + r_{\max}}{2}, \forall i, j, \boldsymbol{Z}_{ij} = 1. \tag{5.35}$$

For example, movie ratings in Netflix are one of $\{1, 2, 3, 4, 5\}$, and the centred ratings should be $\{-2, -1, 0, 1, 2\}$ correspondingly. In this case, one can simply set the magnitudes as any values satisfying the following equation:

$$\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} = \frac{r_{\max} - r_{\min}}{2}, \forall i, j. \tag{5.36}$$

The easiest way is to set $\boldsymbol{r}_i^{\boldsymbol{W}} = \boldsymbol{r}_j^{\boldsymbol{H}} = \sqrt{\frac{r_{\max} - r_{\min}}{2}}$. For example, when applying MBMF on Netflix dataset, after centring the observed matrix by Eq. (5.35) one can set all the magnitudes to be $\sqrt{2}$. The entries in the product matrix will be naturally bounded within $[-2, 2]$. MBMF using pre-defined bounds and setting identical magnitudes is referred as Fixed MBMF (MBMF-F).

However, since MBMF can bound each entry to be within different individual bounds, the centring approach for MBMF should also reflect and utilise this feature. Thus, a new centring approach with respect to the individual bounds determined by the magnitude constraints is proposed:

$$\boldsymbol{V}_{ij}^{C} = \boldsymbol{V}_{ij} - \min(\boldsymbol{V}) - \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}}, \forall i, j, \boldsymbol{Z}_{ij} = 1. \tag{5.37}$$

Also take the movie ratings in Netflix as an example where $\min(\boldsymbol{V}) = 1$. If the magnitude constraints $\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} = 2.5$, then $\boldsymbol{V}_{ij}$ will be mapped from $\{1, 2, 3, 4, 5\}$ to $\{-2.5, -1.5, -0.5, 0.5, 1.5\}$. Note that there might be contradictions when centring

data in this way. If $\boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}} = 1.5$ and $\boldsymbol{V}_{ij} = 5$, Eq. (5.37) gives $\boldsymbol{V}_{ij}^C = 2.5$ which is out of the magnitude bounds $[-1.5, 1.5]$. It happens when $\boldsymbol{V}_{ij} > 2\boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}} + \min(\boldsymbol{V})$ and one needs to modify the magnitude constraints. Otherwise, the corresponding approximation error will produce an unresolvable gap (in the above example, the gap is 1 because the best approximation for $\boldsymbol{V}_{ij}^C$ is 1.5). MBMF using this centring approach and magnitudes from historical data (see next section) is referred as Centred MBMF (MBMF-C).

By using the newly proposed centring approach, MBMF-C does not requires the awareness of any pre-defined systematic bounds once the magnitude constraints are properly set. In the Experiments Section, we can see that MBMF-C can work on not only datasets with pre-defined ranges, such as Jester ($[-10, 10]$) and BookCrossing ($[1, 10]$), but also datasets which do not have such pre-defined ranges like Ali Mob.

5.6.2 Choice of Magnitudes

The magnitude constraints imported in MBMF can be set according to Eq. (5.36) if pre-defined bounds $[r_{\min}, r_{\max}]$ are available. When the bounds are unavailable or to fully utilise the advantages of MBMF, the magnitudes can be obtained in other ways, such as from historical records, user/item profiles and/or social connections. Among them, using historical records is relatively feasible and practical since most of the real world recommender systems collect data in time streams. Here, a method in the implementation to obtain good magnitudes based on historical data is introduced.

Assume that $\boldsymbol{V}$ is the current factorising matrix and $\boldsymbol{V}'$ is the corresponding historical matrix. Consider the recommendation for user i. A possible way to calculate the magnitude of this user is

$$\boldsymbol{r}_i^{\boldsymbol{W}} = \sqrt{\mathbb{E}(\boldsymbol{V}'_{i:}) + \sigma(\boldsymbol{V}'_{i:})}, \tag{5.38}$$

where $\mathbb{E}(\cdot)$ is the mean value and $\sigma(\cdot)$ is the standard deviation. Note that this equation requires the historical matrix to be non-negative, because if there are negative values, the mean cannot reflect the degree of goodness. Although this magnitude calculation is based on the individual average rating and allows possible variations as well, there might be cases in which a user has very few historical records, and applying Eq. (5.38) to calculate the magnitude in these cases can cause over-fitting issues. For example, if user i only has rated a movie 1 out of 5 in the past, Eq. (5.38) gives $\boldsymbol{r}_i^{\boldsymbol{W}} = 1$ since the mean is 1 and the standard deviation is 0, which means this user will be super strict for all of its future ratings. Thus, introduce the global influence for individuals to mitigate such issues:

$$\boldsymbol{r}_i^{\boldsymbol{W}} = \omega_i \sqrt{\mathbb{E}(\boldsymbol{V}'_{i:}) + \sigma(\boldsymbol{V}'_{i:})} + (1 - \omega_i) \sqrt{\mathbb{E}(\boldsymbol{V}') + \sigma(\boldsymbol{V}')}, \tag{5.39}$$

where ω_i is the weight of how much impact in the magnitude is from the user itself. In the implementation, the following formula is used to determine the value of ω_i:

$$\omega_i = \begin{cases} 0, & \text{user } i \text{ has no historical data} \\ \min(\sum \boldsymbol{Z}'_{i:}/(\rho M), 1), & \text{otherwise} \end{cases} \tag{5.40}$$

where $\boldsymbol{Z}'$ is the indicator matrix for historical data, $\rho \in (0, 1]$ is a consistent percentage denoting how much historical data a user needs to completely represent its own preferences, and M is the number of items. More specifically, if a user has no records (i.e. $\omega_i = 0$), MBMF uses the global impact from all other users to define its magnitude. If a user has rated more than ρ of all the items in the past (i.e. $\omega_i = 1$), MBMF will fully use its historical ratings to determine its magnitude; the more historical data a user has, the more weight will be imposed on the impact from its own, and vice versa.

For the proposed MBMF-N, one can directly use Eq. (5.39) to compute the magnitudes required. For MBMF-C, the magnitudes $\boldsymbol{r}^C$ have a relationship with

those of MBMF-N ($\boldsymbol{r}^N$):

$$\boldsymbol{r}^C = \boldsymbol{r}^N/\sqrt{2} \tag{5.41}$$

which provides a feasible approach to obtain magnitudes for MBMF-C.

5.7 Complexity Analysis

The computation complexity of the proposed MBMF algorithm is analysed both theoretically and numerically.

For each iteration, MBMF first calculates the partial derivatives of $\mathcal{F}$ with respect to $\boldsymbol{W}$ and $\boldsymbol{H}$, which costs $2NMK + 2(N+M)K^2$ addition and $2NMK + 2(N+M)K^2 + (N+M)K$ multiplication operations; then it computes the partial derivatives of $\boldsymbol{W}$ and $\boldsymbol{H}$ with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$, and requires $2(N+M)(K-1)$ trigonometric operations plus $2(N+M)(K-2)(K-1)$ times multiplication; it also operates $2(N+M)(K-2)$ multiplication to transform $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ to $\boldsymbol{W}$ and $\boldsymbol{H}$. Thus the overall complexity of MBMF is $O(NMK)$, which is the same as the traditional MF approach.

Experiments are conducted to show the time consumed of MBMF on 7 different scales of matrices against other bounding algorithms BMF, BMC-ADMM and Bounded-SVD. The matrix scale varies from 10^3 to 10^{11}. The experiments are divided into two groups. One is conducted on small and full matrices with all 7 algorithms (repeated 5 times), and the other is conducted on large and sparse matrices (run once) without BMF and Bounded-SVD since they are extremely time consuming. All complexity experiments are conducted on a computer with i7-6900K CPU, 16G memory and Windows 10. The average results are presented in Table 5.1.

The results show that, (i) BMF and Bounded-SVD are quite fast when the matrix is small, but their running time increases when the matrix size becomes larger. Since the third case study, they start to fall far behind to the other bounding algorithms,

Table 5.1 : Running Time (s) on Different Scales of Matrix.

N	M	K	density	BMF	BMC-ADMM	Bounded-SVD	MBMF-N
20	50	10	1	**0.1497**	2.2388	0.2631	0.3018
100	100	10	1	**0.5007**	8.8932	2.6718	0.7857
200	500	10	1	5.7100	4.4400	23.291	**3.0845**
1,000	1,000	10	1	313.75	**14.908**	201.71	18.293
2,000	5,000	10	1	4,753.4	128.73	2,101.3	**91.020**
	Avg.			1,014.7	31.841	465.84	**22.697**
100,000	100,000	10	0.001	-	34,025	-	**5,728.3**
200,000	500,000	10	0.0002	-	442,250	-	**12,251**
	Avg.			-	238,138	-	**8989.7**

and for the group of large scale matrices, they are even incapable of converging in a reasonable time. (ii) BMF-ADMM is not the fastest algorithm on small matrices, but it can still be applied to large scale instances, despite its long running time. (iii) MBMF-N is the overall fastest algorithm not only on small matrices, but also on large scale ones. This demonstrates that MBMF is the most efficient bounding algorithm and has great advantages when applied on large-scale recommender systems.

5.8 Experiments

The prediction variation of algorithms is explored on synthetic datasets and their recommendation performance is compared against related algorithms on several real datasets including Jester [76], BookCrossing [77], and Tianchi Mobile Record datasets. The statistics of these datasets after preprocessing (i.e. removing invalid records) are presented in Table 5.2. The baseline algorithms chose for comparison with the proposed MBMF-F, MBMF-C and MBMF-N are:

- *Matrix Factorisation for recommender systems (MF)* [2]. The conventional approach used for recommender systems.
- *Non-negative Matrix Factorisation algorithms (NMF)* [1]. Using "multiplicative rules" instead of addition formula in the SGD, NMF becomes popular when dealing with non-negative observations.
- *Feature-wise updated Cyclic Coordinate Descent method (CCD++)* [78]. It is a new and fast non-bounding approach to handle large-scale datasets.
- *Bounded Matrix Factorisation (BMF)* [25]. This is the first algorithm studying bounding issues in matrix factorisation for recommender systems.
- *Bounded Matrix Completion in Alternating Direction of Multiplier Method (BMC-ADMM)* [26]. It solved the convergence issue in BMF with the help of ADMM framework.
- *Bounded Singular Value Decomposition (Bounded-SVD)* [27]. This approach tries to bound factorising entries by imposing exponential penalties.

Table 5.2 : Statistics of Datasets used for Experiments

dataset	users	items	density	range	ordinal
Synthetic	500	500	0.2	[0,10]	N
Jester	73,421	100	0.5634	[-10,10]	N
BookCrossing	92,184	270,169	0.00004	[1,10]	Y
Ali Mob	10,000	8,916	0.0101	[1,6853]	Y

Note that NMF and MBMF-N require the input data to be non-negative, MBMF-C requires the data to be centred with respect to magnitudes, MBMF-F requires

the data to be centred with respect to systematic bounds, while other algorithms have no additional requirements. Thus, each dataset has been pre-processed into a non-negative and two centred ones.

Three metrics are used to evaluate the performance of recommendations:

(1) *Root Mean Square Error (RMSE)* [68], which evaluates difference between the recovered ratings and their corresponding original ratings:

$$\text{RMSE} = (\sum_{ij} \boldsymbol{M}_{ij})^{-1/2} \|\boldsymbol{M} * (\boldsymbol{V} - \boldsymbol{W}^* \boldsymbol{H}^*)\|_F, \tag{5.42}$$

where $\boldsymbol{W}^*$ and $\boldsymbol{H}^*$ are the final factorised matrices, and $\boldsymbol{M}$ is a binary matrix in which one denotes the chosen entry for cross validation.

(2) *Mean Absolute Error (MAE)* [79], which evaluates the absolute difference between the recovered ratings and their corresponding original ratings:

$$\text{MAE} = \frac{1}{\sum_{ij} \boldsymbol{M}_{ij}} |\boldsymbol{M} * (\boldsymbol{V} - \boldsymbol{W}^* \boldsymbol{H}^*)|. \tag{5.43}$$

(3) *F1 score* [69], which is a percentage value that evaluates how much proportion of correct recommendations is with respect to users:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5.44}$$

In the experiments, each user's average rating is used as a threshold, and ratings greater than the threshold imply that the corresponding items are recommended. Two binary recommendation vectors before and after factorisation can then be used to calculate the F1 score.

For RMSE and MAE, lower is better, while for F1 Score, higher is better.

The maximal iteration times for all of the algorithms are set at 500. An early termination condition is also applied: if the decrease of the objective function value is smaller than 10^{-5} for 10 consecutive iterations, the algorithm is regarded as converged.

The source codes of MBMF as well as all the preprocessed datasets used for experiments can be downloaded at: https://github.com/shawn-jiang/MBMF.

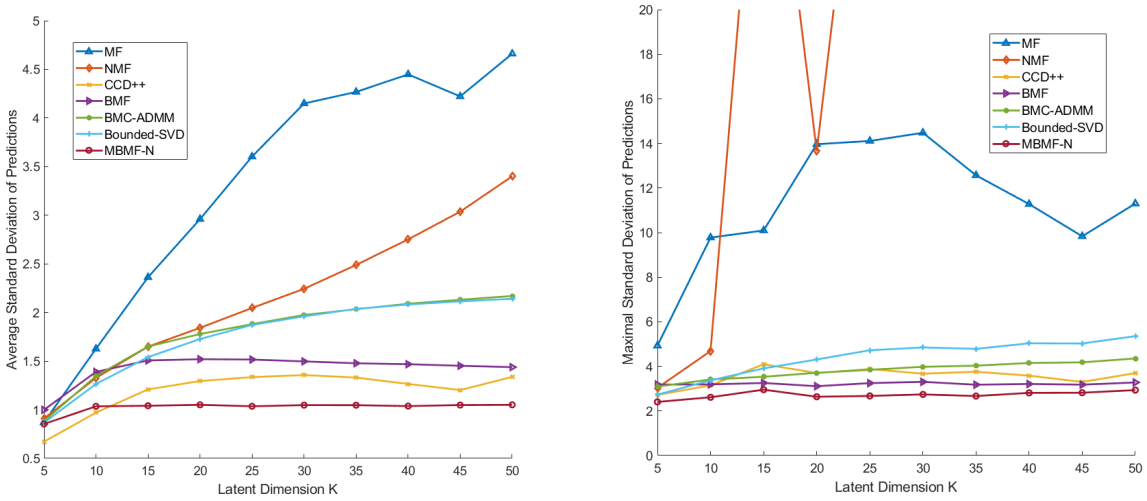
5.8.1 Prediction Variation

Experiments are conducted on synthetic datasets to explore the prediction variation of algorithms. In this experiment, K varies from 5 to 50 with step 5. For each K, the following experimental procedure is adopted:

- Randomly generate a factorising matrix $\boldsymbol{V}$ of size 500×500 and scale its entries to $[0, 10]$;
- Remove 80% of the values (set them to missing data) in $\boldsymbol{V}$ randomly while making sure the remaining part does not have zero rows/columns;
- Run each of the algorithms on the remaining matrix 10 times, and record their recovered matrices;
- Calculate the standard deviation (σ) on missing entries.

The average and maximal standard deviation over predictions is presented in Figure 5.1. The lower the standard deviation is, the more stable the predictions are. Algorithms with stable predictions are more operational on large datasets.

According to Figure 5.1 (note that in Figure 5.1 (b) the maximal σ of NMF is greater than the limit of the y-axis in many cases, since more details can be presented for other algorithms), we can see that four bounding algorithms (BMF, BMC-ADMM, Bounded-SVD and MBMF-N) all successfully reduced the prediction variation compared with the traditional MF and NMF which have no bounding conditions. MBMF-N achieves both the lowest average σ and the lowest maximal σ in all cases, followed by CCD++ (the performance of CCD++ is impressive although it has no bounding constraints), BMF and Bounded-SVD. This experiment



(a) ave(σ) of predictions.

(b) max(σ) of predictions.

Figure 5.1 : Prediction variation on Synthetic dataset.

demonstrates that MBMF is so far the best bounding algorithm in terms of reducing prediction variation, and can be easily applied on real recommender systems: it does not require repeated runs.

5.8.2 Performance on Real Recommender Systems

As for the experiments on real recommendation datasets, each of them is randomly divided into historical data and present data. The historical data can have zero rows/columns while the present data cannot. Then in each run 10% entries from the present data are randomly picked as a validation fold. The latent dimension K varies in $10, 20, 50$. 5 folds are run for each K and the results are averagely reported.

Jester Dataset

Jester is a free online dataset containing about 4.1 million continuous ratings ($[-10, 10]$) of 100 jokes from $73,421$ users collected between April 1999 and May 2003. Due to the high density, the continuity in ratings and the centrosymmetric

bounds, which are rarely seen in other recommender systems, this dataset becomes popular to test recommendation algorithms. Also because of this reason, 80% is used as historical data and 20% as present data. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 100, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 1$; MBMF, $\rho = 0.05, \lambda = 1$. The results are showed in Table 5.3.

Table 5.3 : Recommendation Performance on Jester Dataset

	K	MF	NMF	CCD++	BMF	BMC-ADMM	Bounded-SVD	MBMF-F	MBMF-C	MBMF-N
RMSE	10	5.6107	333.18	5.6643	5.4271	4.9162	6.3798	4.9785	4.6816	**4.6474**
	20	4.9242	98.349	5.5892	5.5027	4.9477	6.7069	5.0395	4.6695	**4.6399**
	50	4.6770	10.161	5.1609	5.3717	4.7541	5.7458	5.2365	4.6773	**4.6344**
	Avg.	4.2949	147.20	5.4710	5.4340	4.8730	6.2780	5.0840	4.6760	**4.6410**
MAE	10	4.2949	9.3592	4.2840	4.2851	3.8149	4.7758	3.9484	3.8045	**3.7467**
	20	3.8589	9.3339	4.2523	4.3581	3.8807	5.1434	4.0252	3.8014	**3.7473**
	50	3.7919	5.6652	3.9493	4.2744	3.7961	4.5270	4.2259	3.8039	**3.7404**
	Avg.	3.9820	8.1190	4.1620	4.3060	3.8310	4.8150	4.0670	3.8030	**3.7450**
F1 Score	10	60.54	57.96	60.40	58.62	60.52	59.90	58.79	64.08	**65.58**
	20	62.26	56.02	60.08	58.96	57.42	58.32	58.62	64.47	**65.71**
	50	64.35	58.07	60.46	58.79	55.22	58.82	56.48	64.16	**65.60**
	Avg.	62.38	57.35	60.31	58.79	57.72	59.01	57.96	64.24	**65.63**

From Table 5.3, we can see that among non-bounding algorithms, NMF fails and gets very high RMSE and MAE, and also the lowest F1 Score. MF is a bit better than CCD++, but overall their performance is close. For bounding algorithms, Bounded-SVD obtains the highest (worst) RMSE and MAE, while BMF gets the lowest F1 Score. The performance of MBMF-N on this dataset is the best over all three metrics. It achieves 4.6410 RMSE, 3.7450 MAE and 65.63% F1 Score, followed by MBMF-C with 4.6760 RMSE, 3.8030 MAE and 64.24% F1 Score. The performance of MBMF-F is better than BMF and Bounded-SVE, but worse than BMC-ADMM, MBMF-C and MBMF-N. This demonstrates that bounding the pre-

dicted values in product matrix in individually concise ranges can help improve the recommendation performance. Another observation is that, the improvement of MBMF to comparative algorithms is massive on this dataset. This is because the magnitudes obtained from the historical ratings can represent users and jokes very well. From Table 5.2 we know that this dataset is the smallest and most dense one compared to the other two datasets, which means the proposed MBMF is superior to all other existing algorithms on small and dense systems.

BookCrossing Dataset

Bookcrossing is a large online dataset which contains 278,858 users with 1,149,780 ratings (explicit / implicit) about 271,379 books. However, neither all users nor all books have corresponding ratings in this dataset. From the ratings file, a rating matrix with $92,184$ rows (users) and $270,169$ columns (books) is constructed. It has been randomly split into 50% historical and 50% present data. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 0.1, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 1$; MBMF, $\rho = 0.05, \lambda = 1$. Only one experiment using BMF for each K is done on this dataset since it is too time consuming (the experiment with K set 50 took about 30 days). The results are showed in Table 5.4.

From Table 5.4, BMC-ADMM gets the worst performance, with the highest RMSE and MAE and the lowest F1 Score. Except for the proposed MBMF-C, MBMF-F and MBMF-N, the best algorithms on this dataset are CCD++ (non-bounding) and BMF (bounding). Their performance is close and CCD++ also obtains the best F1 Score among all algorithms. However, when it comes to RMSE and MAE, MBMF-C is the best which achieves the lowest RMSE (1.1810) and MAE (0.7825) in all cases, followed by MBMF-N. Even over the F1 Score, MBMF-N is the second best, scoring 86.32% just after CCD++ (87.08%). Notice that this dataset is

Table 5.4 : Recommendation Performance on BookCrossing Dataset

	K	MF	NMF	CCD++	BMF	BMC-ADMM	Bounded-SVD	MBMF-F	MBMF-C	MBMF-N
RMSE	10	1.7566	1.8827	1.3137	1.3409	3.3757	1.6221	1.3898	**1.1783**	1.2598
	20	1.3539	1.6211	1.3191	1.2863	3.0368	1.3541	1.3929	**1.1846**	1.2639
	50	1.6726	1.4146	1.3131	1.2990	2.8886	1.6021	1.3951	**1.1813**	1.2628
	Avg.	1.5940	1.6390	1.3150	1.3090	3.1000	1.5260	1.3930	**1.1810**	1.2620
MAE	10	1.3164	1.2889	0.8713	0.9329	3.0033	1.1873	0.9909	**0.7810**	0.8503
	20	0.9979	1.1479	0.8753	0.8563	2.6781	0.9712	0.9893	**0.7849**	0.8534
	50	1.2578	0.9779	0.8699	0.8653	2.5543	1.2017	0.9943	**0.7817**	0.8526
	Avg.	1.1910	1.1380	0.8722	0.8848	2.7450	1.1200	0.9915	**0.7825**	0.8521
F1 Score	10	**87.25**	82.68	87.05	86.29	81.93	87.02	83.98	85.56	86.33
	20	86.85	83.98	87.07	86.29	78.93	**87.47**	84.01	85.64	86.30
	50	77.31	86.43	**87.12**	86.06	67.27	77.94	84.02	85.63	86.32
	Avg.	83.80	84.36	**87.08**	86.21	76.04	84.14	84.00	85.61	86.32

not only large-scale ($92,184 \times 270,169$) but very sparse (0.00004 density) as well (see Table 5.2). Thus, the results demonstrate that MBMF is also suitable to be applied on large-scale and sparse datasets. The improvements from MBMF-F to MBMF-C/MBMF-N verify again the effectiveness of individual bounding motivations.

Ali Mob Dataset

Ali Mob is a public recommendation dataset provided by Tianchi platform. It recorded the online shopping behaviours of users on Alibaba's M-Commerce platforms in 2014. The dataset consists of $10,000$ users and $2,876,947$ commodities in $8,916$ categories. $12,256,906$ valid records were collected in this dataset which contained users' behaviours (click, collect, add-to-cart and payment) towards commodities. This dataset is preprocessed as following:

- Count the number of each behaviour (b_1, b_2, b_3, b_4) for each user (u) on each category of commodities (c);

- Calculate the interest points (p) of user u on category c:

$$p = w_1 b_1 + w_2 b_2 + w_3 b_3 + w_4 b_4, \tag{5.45}$$

  where w_1, w_2, w_3, w_4 are behaviour weights. Set $w_1 = 1, w_2 = 2, w_3 = 3, w_4 = 5$ in the experiments according to the increase of interest from "click" to "payment".

- Construct the data matrix with interest points whose rows denote users and columns denote categories.

This dataset does not have a pre-defined pair of bounds and all values in the factorising matrix are non-negative. Since the range of observations varies among users/items, the maximal value and minimal value are used as the upper and lower bounds respectively for algorithms (i.e. BMF, BMC-ADMM, Bounded-SVD and MBMF-F) which can only handle one single pair of bounds. Half ratings are used as historical data while the other half is used as the present. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 1, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 0.01$; MBMF, $\rho = 0.05, \lambda = 1$. The results are showed in Table 5.5.

As shown in Table 5.5, among non-bounding algorithms, MF and NMF fail to get reasonable approximations and recommendations, while CCD++ works somewhat. Both MF and CCD++ obtain better results when K is set at 50. This implies that the performance of these algorithms is affected by the value of K. Among bounding algorithms, both BMF and BMC-ADMM fail. They are projection-based algorithms which project the product values that are out of range to the exact boundaries. So this bad performance implies that the projection approach cannot handle datasets without pre-defined bounds. As for Bounded-SVD, it is observed from the dataset that most of the interest points are small, and only a tiny part of users on specific

Table 5.5 : Recommendation Performance on Ali Mob Dataset

	K	MF	NMF	CCD++	BMF	BMC-ADMM	Bounded-SVD	MBMF-F	MBMF-C	MBMF-N
RMSE	10	115.93	235511	63.087	549.66	319.65	43.381	50.548	43.273	**42.540**
	20	95.286	4540.0	59.334	455.34	392.39	**42.810**	52.530	43.671	42.903
	50	61.138	11238	56.933	380.98	593.43	**41.766**	50.532	43.881	42.999
	Avg.	90.784	131098	59.785	461.99	435.16	**42.652**	51.200	43.608	42.814
MAE	10	29.930	2686.0	22.667	507.57	157.90	15.827	21.459	**14.405**	18.963
	20	34.871	959.17	25.103	428.79	205.80	15.953	20.910	**14.329**	18.948
	50	29.975	1223.1	26.128	362.07	358.47	16.438	21.757	**14.400**	18.896
	Avg.	31.592	1622.8	24.633	432.81	240.73	16.073	21.380	**14.378**	18.935
F1 Score	10	38.99	35.90	39.58	32.07	17.70	41.83	35.06	**43.47**	39.64
	20	38.22	34.29	38.66	32.28	18.39	42.41	34.51	**43.51**	39.70
	50	38.76	34.77	38.29	31.69	22.85	42.68	34.76	**43.56**	39.72
	Avg.	38.66	34.99	38.84	32.01	19.65	42.31	34.78	**43.51**	39.69

categories of commodities have ridiculously high points. Therefore, the way that Bound-SVD restricts all the predictions to be small mitigates the inaccurate recommendation for those extreme users and leads to good results. However, MBMF-C and MBMF-N considers all users' interest points over all categories and sets different magnitudes for them. From the results we can see that, MBMF-C achieves the best performance over MAE (14.378 in average) and F1 score (43.51% in average) in all cases. On the other hand, MBMF-F is still workable on this dataset, although its performance falls far behind the ones with magnitude constraints. This experiment demonstrates that the proposed MBMF can properly handle datasets which do not have pre-defined bounds.

From all the experiments on real-world datasets, the proposed MBMF achieves the best performance in most cases (29 out of 36 – three datasets, three metrics and four Ks), and comparable in the remaining ones.

5.8.3 Discussion

From the experimental results, a phenomenon is interesting and noteworthy: the performance of MBMF does not change much when K varies, while other methods obtain their best results with different K when applied on different datasets. This observation reveals the high stability, a unique feature of MBMF which other methods do not have. It implies that the selection of latent dimension K has little impact on the performance of MBMF. This is very useful when applying it on large scale datasets because it means one does not have to repeat experiments to determine the best suitable K which is usually required by existing methods. The reason of the high stability in it is the magnitude constraint: no matter what K's value is, the magnitude of row/column in the factorised matrices is the same.

To some degree, however, the magnitude constraints are quite strict as well: since all the rows/columns in the factorised matrices have to conform to these magnitude constraints, there is not much space for the SGD method to approximate the product of factorised matrices to be as close to the factorising matrix as in other algorithms. In the future work, the magnitude constraints can be relaxed from equalities to inequalities (i.e. $|\boldsymbol{W}_{i:}| \leqslant \boldsymbol{r}_i^{\boldsymbol{W}}$), which will make MBMF more flexible but harder to solve as well. It is believed that imposing magnitude constraints is an effective way to solve bounded MF.

5.9 Summary

In this chapter, a novel MF algorithm for recommender systems called MBMF is proposed. It is a bounded algorithm which reduces the fluctuation of predictions especially for large and sparse datasets. The magnitude constraints allow MBMF to have individually different bounds for each entry and make the algorithm capable of dealing with data that has no fixed bounds. The conversions of coordinates used in the method of MBMF turn the magnitude constrained optimisation into an equiv-

alent unconstrained one by setting the radii as corresponding magnitudes. As for the unconstrained task, a stochastic gradient descent method is applied to solve it efficiently. In order to update the variables efficiently, an acceleration approach is introduced which involves tensor construction and multiplication based on the derivative patterns. A MBMF variant, Non-negative MBMF is also proposed to deal with non-negative data. Compared with the unconstrained MF, NMF and CCD++ as well as the-state-of-art bounding methods BMF, BMC-ADMM and Bounded-SVD, MBMF achieved superior performance in most cases over all evaluation metrics on both synthetic datasets and real recommendation datasets. Moreover, the improvements from MBMF-F to MBMF-C/MBMF-N verify the effectiveness and benefits of bounding product entries in different individual ranges.

Chapter 6

Conclusion

As one of the most popular mathematical techniques in machine learning area, low-rank MF/TF has been widely applied in real-world applications for its simple implementation, great effectiveness, high efficiency and strong interpretability. Imposing constraints in the vanilla MF/TF has been proven effective to cater for various scenarios. In this thesis, by the adoption of different constraints/regularisations, three constrained MF/TF algorithms have been studied and demonstrated in detail.

The relative relationships among features in factorised matrices of NMF characterise the relations in real world data. This thesis introduces a novel algorithm, RPR-NMF to incorporate such relative relationships in a way of imposing RPR constraints to retain the expected relationships after factorisation. Different from the pairwise weights and partial labels, RPRs are much more flexible and applicant. Meanwhile, it conforms to the MUL framework which guarantees its efficiency and convergence with detailed proofs presented.

It faces more difficulties to study the relative relationships among features in high dimensional tensor data, requiring the consideration of two different decomposition schemes, CP and Tucker. This thesis introduces RPR-NTF for both schemes. To avoid complicated tensor calculations, Kronecker and Khatri-Rao products are utilised in the matricisation of update rules which significantly improves computation efficiency.

For the bounding issues of low-rank MF when applied on recommender systems, this thesis introduces a novel bounding algorithm, MBMF. It provides a totally new

perspective on the issue which bounds predictions by incorporating magnitude constraints, according to the fundamental bounding natures of vector inner product. The conversion from cartesian to spherical coordinate systems significantly simplifies the original quadratically constrained quadratic programming and enables the efficient application of SGD solver. Furthermore, the acceleration and choice of magnitudes provided make MBMF the state-of-the-art.

Numerous experiments have been conducted to verify not only the effectiveness of proposed algorithms, but also their performance on real-world datasets. The results have demonstrated that the proposed RPR-NMF, RPR-NTF and MBMF are superior to existing algorithms in terms of both accuracy and efficiency.

To better understand the implications of these algorithms, future studies may consider addressing these research problems in other ways:

- The RPRs used in RPR-NMF and RPR-NTF are denoted by triplets and modelled by hinge and exponential functions, so the effectiveness of other different forms of RPRs and functions for modelling them can be further explored.
- To impose the RPRs, one may try other optimisation approaches rather than the penalty method used in this thesis.
- As discussed in the chapter of MBMF, the equality magnitude constraints restrict the matrix approximation, thus inequalities may be considered to take the place.
- SGD is used to solve the unconstrained MBMF problem, however, the convergence of SGD for solving non-convex functions are remained undiscovered.
- A popular direction in this area is to combine low-rank MF/TF algorithms with deep neural networks, then the RPRs and magnitude constraints can be considered to incorporate with them in some way.

Bibliography

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[4] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.

[5] C. H. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 45–55, 2008.

[6] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. ACM, 2013, pp. 587–596.

[7] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2140–2151, Oct 2013.

[8] E. Esser, M. Moller, S. Osher, G. Sapiro, and J. Xin, "A convex model for non-negative matrix factorization and dimensionality reduction on physical space," *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3239–3252, July 2012.

[9] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint non-negative matrix factorization," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, vol. 13, May 2013, pp. 252–260.

[10] H. Kim, J. Choo, J. Kim, C. K. Reddy, and H. Park, "Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. ACM, 2015, pp. 567–576.

[11] E. F. Gonzalez and Y. Zhang, "Accelerating the lee-seung algorithm for non-negative matrix factorization," *Dept. Comput. & Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR-05-02*, 2005.

[12] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[13] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, vol. 5, 2005, pp. 606–610.

[14] N. Gillis and S. A. Vavasis, "Fast and robust recursive algorithmsfor separable nonnegative matrix factorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 4, pp. 698–714, 2014.

[15] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Transactions on Signal Processing*, vol. 62, no. 1, pp. 211–224, 2014.

[16] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research*, vol. 5, pp. 1457–1469, Nov 2004.

[17] 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*, vol. 28, no. 3, pp. 403–415, March 2006.

[18] K. Kimura, M. Kudo, and Y. Tanaka, "A column-wise update algorithm for nonnegative matrix factorization in bregman divergence with an orthogonal constraint," *Machine Learning*, vol. 103, no. 2, pp. 285–306, 2016.

[19] A. Ozerov and C. Fevotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, March 2010.

[20] R. Sandler and M. Lindenbaum, "Nonnegative matrix factorization with earth mover's distance metric for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1590–1602, Aug 2011.

[21] V. P. Pauca, J. Piper, and R. J. Plemmons, "Nonnegative matrix factorization for spectral data analysis," *Linear algebra and its applications*, vol. 416, no. 1, pp. 29–47, 2006.

[22] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1548–1560, 2010.

[23] H. Liu, Z. Wu, X. Li, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1299–1311, 2011.

[24] C. Wang, X. He, J. Bu, Z. Chen, C. Chen, and Z. Guan, "Image representation using laplacian regularized nonnegative tensor factorization," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2516–2526, 2011.

[25] R. Kannan, M. Ishteva, and H. Park, "Bounded matrix factorization for recommender system," *Knowledge and Information Systems*, vol. 39, no. 3, pp. 491–511, 2014.

[26] H. Fang, Z. Zhang, Y. Shao, and C.-J. Hsieh, "Improved bounded matrix completion for large-scale recommender systems," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1654–1660.

[27] B. H. Le, K. Q. Nguyen, and R. Thawonmas, "Bounded-svd: A matrix factorization method with bound constraints for recommender systems," in *2015 International Conference on Emerging Information Technology and Engineering Solutions*, Feb 2015, pp. 23–26.

[28] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012, pp. 17–23.

[29] S. Jia and Y. Qian, "Constrained nonnegative matrix factorization for hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 1, pp. 161–173, 2008.

[30] S. Z. Li, X. Hou, H. Zhang, and Q. Cheng, "Learning spatially localized, parts-based representation," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, Dec 2001, pp. I–207–I–212.

[31] B. Shen and L. Si, "Non-negative matrix factorization clustering on multiple manifolds," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, ser. AAAI-10, Jul 2010, pp. 575–580.

[32] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-10)*, vol. 10, 2010, pp. 236–241.

[33] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010, pp. 199–210.

[34] Y. Koren, "Collaborative filtering with temporal dynamics," *Commun. ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[35] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. ACM, 2010, pp. 1029–1038.

[36] H. A. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 3, pp. 105–122, 2000.

[37] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[38] Y.-D. Kim and S. Choi, "Nonnegative tucker decomposition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[39] M. Mørup, L. K. Hansen, and S. M. Arnfred, "Algorithms for sparse nonnegative tucker decompositions," *Neural computation*, vol. 20, no. 8, pp. 2112–2131, 2008.

[40] P. Jain and S. Oh, "Provable tensor factorization with missing data," in *Advances in Neural Information Processing Systems*, 2014, pp. 1431–1439.

[41] S. Zafeiriou, "Algorithms for nonnegative tensor factorization," in *Tensors in Image Processing and Computer Vision*. Springer, 2009, pp. 105–124.

[42] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 81–90.

[43] A. Cichocki, R. Zdunek, and S.-i. Amari, "Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 169–176.

[44] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari, "Non-negative tensor factorization using alpha and beta divergences," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 3. IEEE, 2007, pp. III–1393.

[45] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 3, pp. 708–721, 2009.

[46] G. Zhou, A. Cichocki, and S. Xie, "Fast nonnegative matrix/tensor factorization based on low-rank approximation," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2928–2940, 2012.

[47] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[48] A. Cichocki and R. Zdunek, "Regularized alternating least squares algorithms for non-negative matrix/tensor factorization," in *International Symposium on Neural Networks*. Springer, 2007, pp. 793–802.

[49] B. Hidasi and D. Tikk, "Fast als-based tensor factorization for context-aware recommendation from implicit feedback," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 67–82.

[50] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[51] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal pattern-aware qos prediction via biased non-negative latent factorization of tensors," *IEEE transactions on cybernetics*, 2019.

[52] H. Lee, Y.-D. Kim, A. Cichocki, and S. Choi, "Nonnegative tensor factorization for continuous eeg classification," *International journal of neural systems*, vol. 17, no. 04, pp. 305–317, 2007.

[53] J. C. Ho, J. Ghosh, and J. Sun, "Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 115–124.

[54] T. Van de Cruys, "A non-negative tensor factorization model for selectional preference induction," *Natural Language Engineering*, vol. 16, no. 4, pp. 417–437, 2010.

[55] E. Benetos and C. Kotropoulos, "Non-negative tensor factorization applied to music genre classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 1955–1967, 2010.

[56] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 792–799.

[57] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, "Facewarehouse: A 3d facial expression database for visual computing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, pp. 413–425, 2013.

[58] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 727–736.

[59] K. Takeuchi and N. Ueda, "Graph regularized non-negative tensor completion for spatio-temporal data analysis," in *Proceedings of the 2nd International Workshop on Smart*, 2016, pp. 1–6.

[60] X. Li, M. K. Ng, G. Cong, Y. Ye, and Q. Wu, "Mr-ntd: Manifold regularization nonnegative tucker decomposition for tensor data dimension reduction and representation," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 8, pp. 1787–1800, 2016.

[61] Y. Qiu, G. Zhou, Y. Zhang, and S. Xie, "Graph regularized nonnegative tucker decomposition for tensor data representation," in *ICASSP 2019-2019 IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 8613–8617.

[62] H. Maki, H. Tanaka, S. Sakti, and S. Nakamura, "Graph regularized tensor factorization for single-trial eeg analysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 846–850.

[63] Y.-X. Wang, L.-Y. Gui, and Y.-J. Zhang, "Neighborhood preserving non-negative tensor factorization for image representation," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3389–3392.

[64] © 2018 IEEE. Reprinted, with permission, from S. Jiang, K. Li, and Y. D. R. Xu, "Relative Pairwise Relationship Constrained Non-negative Matrix Factorisation", IEEE Transactions on Knowledge and Data Engineering, 2018.

[65] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM journal on matrix analysis and applications*, vol. 30, no. 2, pp. 713–730, 2008.

[66] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[67] 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*. ACM, 2003, pp. 267–273.

[68] A. G. Barnston, "Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score," *Weather and*

Forecasting, vol. 7, no. 4, pp. 699–709, 1992.

[69] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[70] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*. IEEE, 1994, pp. 138–142.

[71] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image and Vision Computing*, vol. 28, no. 5, pp. 807–813, 2010.

[72] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, vol. 6, April 2006, pp. 549–553.

[73] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, Dec 2015.

[74] © 2020 IEEE. Reprinted, with permission, from S. Jiang, K. Li, and Y. D. R. Xu, "Magnitude Bounded Matrix Factorisation for Recommender Systems", IEEE Transactions on Knowledge and Data Engineering, 2020.

[75] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Physica-Verlag HD, 2010, pp. 177–186.

[76] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, Jul 2001.

[77] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. ACM, 2005, pp. 22–32.

[78] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems," in *2012 IEEE 12th International Conference on Data Mining*, Dec 2012, pp. 765–774.

[79] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.