Tensor Decomposition Method Applied to Recommendation Systems

by

Sheng Chai

A thesis submitted to the

School of Computing

in conformity with the requirements for

the degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

September 2021

Abstract

With the growth of network web services, web service recommendations based on the quality of service (QoS) attribute have become a research interest topic. Service recommendation technologies can help users discover new web services and make their online experience better. Further, by providing users with recommendations for high-quality web services, these technologies ultimately benefit both users and service providers. In this thesis, we study the tensor decomposition in web service recommendations. In particular, we propose new tensor computational methods and algorithms for QoS attribute prediction to improve the recommendation accuracy. Our methods follow the machine learning techniques.

First, to remedy the shortage of low prediction accuracy rates caused by the lack of initial data samples, a traversal-tensor method (TTM) is proposed to enhance the sampling scheme. The new method integrates the feature factor matrices to construct more data samples for tensor decomposition. We analyze and validate the new algorithm in comparison with the traditional tensor decomposition applied to service recommendations. Empirical studies with multiple datasets show that the TTM effectively improves the prediction performance.

Second, a modified regularization term is designed and applied with the TTM to overcome the overfitting problem. This is done by using a linear combination of two commonly applied regularization models. It is shown that the updated term can increase the accuracy rate of predicting QoS attributes and better support the TTM method.

Third, a two-step strategy approach involving a K-means clustering with TTM is introduced to deal with the initial unorganized data. The pre-clustered data are used as input to the TTM to complete the QoS attribute prediction. This process is evaluated between our methods and the clustering method.

The thesis describes a framework of tensor-based web service recommendation by synthesizing the above methods. This framework is centered on TTM, with a modified regularization term to support TTM and a method to handle the initial unorganized data.

Co-Authorship

Journal Article

Sheng Chai, Wenying Feng, and Hossam S. Hassanein. Tensor decomposition-based web service QoS prediction. *Journal of Coupled Systems and Multiscale Dynamics*, vol. 4, no. 2, pp.113-118, June. 2016.

Posters

1. **Sheng Chai**, Wenying Feng, and Hossam S. Hassanein. A Tensor Decomposition Approach Based on User Trust for QoS Web Service Recommendation. Poster presented at *AMMCS-CAIMS Congress*, Waterloo Canada, June 2015.
2. **Sheng Chai**, Wenying Feng, and Hossam S. Hassanein. Fuzzy Control Based on Local Path Planning for Mobile Robot. Poster presented at *5th Annual Queen's Graduate Computing Society Conference*, Kingston, Ontario, Canada, May 2014.

Presentation

1. **Sheng Chai**, Wenying Feng, and Hossam S. Hassanein. QoS Prediction New Strategy with Clustering and Tensor Decomposition. Presentation at *CAIMS annual meeting 2021*, Waterloo Canada, June 2021.
2. **Sheng Chai**, Wenying Feng, and Hossam S. Hassanein. QoS Recommendation Approach in Web Services. Presentation at *IV International AMMCS Interdisciplinary Conference*, Waterloo Canada, August 2017.
3. **Sheng Chai**, Wenying Feng, and Hossam S. Hassanein. Multi-View Recommendation Model Using Adaptive Resonance Theory and Canonical Correlation Analysis. Presentation at *ICML 2016 Workshop on MVRL*, New York USA, June 2016.

Acknowledgments

Please allow me to express my gratitude to the kindest people.

Firstly, my deep gratitude goes to Dr. Wenying Feng and Dr. Hossam S, Hassanein. Thank my professors very much for always being so helpful to me. I could not have done it without you.

I also want to say how I would greatly appreciate the thesis committee: Prof. Nick Graham and Prof. Abd-Elhamid Taha. Thank you many for your comments which provided me many insightful suggestions on my proposal and thesis work.

I would like to seriously thank Debby Robertson for being helpful throughout the program process. In particular, she gave me clear guidance and encouragement when I was going through the process of medical leaving.

I would also like to give a special thanks to Basia. Thanks to her help in review working, I can have the confidence to continue the thesis writing.

Thanks also to Dr. Yaser Al Mtawa and Dr. Hesham Farahat from the Telecommunications Research Lab (TRL). I appreciate your encouragement during my studies.

Finally, I would like to thank my family and parents. I hope I can repay all of you for all you have done.

Statement of Originality

I hereby certify that this Ph.D. thesis is original and that all ideas and inventions attributed to others have been properly referenced.

Table of Contents

List of Figures

List of Tables

Glossary

ALS	Alternating least squares algorithm
CANDECOMP	Canonical decomposition
CP decomposition	CANDECOMP and PARAFAC decomposition
EEG	Electroencephalogram
PARAFAC	Parallel factor analysis
PCA	Principal components analysis
QoS	Quality of service
R	Rank of the tensor. Its default value is 1
RTD	Regular tensor decomposition model
TTM	Traversal-tensor method
a, b, c	Vectors that belong to $\mathbb{R}^2$. $\bar{\boldsymbol{\mathcal{X}}} = a \circ b \circ c$
$a_1, a_2, b_1, b_2, c_1, c_2$	Element of the corresponding vector
α, β	Angle variables are in a range from -π to π
T, U, X	Matrix
$\mathbb{R}$	Set of real numbers
$\mathrm{I}_1 \times \mathrm{I}_2 \times \mathrm{I}_3$	Dimensionality. I_1, I_2, and I_3 is the dimensionality of first, second, and third-dimensional data, respectively
N-way	$\boldsymbol{N}$-dimensions
$\boldsymbol{\mathcal{X}}$	Original tensor
$\bar{\bar{\boldsymbol{\mathcal{X}}}}$	Aggregated tensor of $\boldsymbol{\mathcal{X}}$

$\bar{\boldsymbol{\mathcal{X}}}, \bar{\boldsymbol{\mathcal{X}}}'$	Approximation tensor of $\boldsymbol{\mathcal{X}}$
x_{ijk}	$(\mathrm{i}, \mathrm{j}, \mathrm{k}) - th$ entry of 3-way tensor
$x_{i_1 i_2 \dots i_N}$	Elements of the $\boldsymbol{\mathcal{X}}$ tensors
$i_1, i_2, \dots, i_N$	Each entry of the $\boldsymbol{\mathcal{X}}$ tensor's elements.
$\mathrm{X}_{(\mathrm{n})}, \overline{\mathrm{X}_{(\mathrm{n})}}, \overline{\mathrm{X}_{(\mathrm{n})}}'$	Mode-n matricization of the tensor $\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}$, and $\bar{\boldsymbol{\mathcal{X}}}'$ (Or n-th frontal slice of $\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}$, and $\bar{\boldsymbol{\mathcal{X}}}'$) $n = 1,2, and\ 3$
$U_{new}^{(n)}$	New factor matrices. $n = 1,2, and\ 3$
$U^{(\mathrm{n})}$	Regular factor matrices. $n = 1,2, and\ 3$
$\overline{U^{(\mathrm{n})}}$	Enhanced factor matrices. $n = 1,2, and\ 3$
$\Delta U^{(\mathrm{n})}$	Feature factor matrices. $n = 1,2, and\ 3$
$\odot$	Khatri-Rao product
*	Hadamard product
T	Matrix transpose
$+$	Pseudoinverse
$\circ$	Vector outer product
$L(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$	Objective function
$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$	Loss function
$\Omega(\boldsymbol{\mathcal{X}})$	Regular regularization term

Chapter 1

Introduction

This thesis presents a tensor decomposition method with QoS attribute prediction to improve the recommendation accuracy. We provide an overview of the research motivation and problems in Sections 1.1 and 1.2. We highlight the main contributions of this thesis in Section 1.3. Section 1.4 outlines the thesis structure.

1.1 Motivation

As more and more developers, enterprises, and organizations worldwide are becoming service providers, developing and delivering web services of varying functionality on various web service platforms has led to a dramatic increase in the number of services on each platform. As of June 26, 2021, more than 24,237 Web APIs are published on Programmable Web, a well-known web services and API provisioning platform, including over 1117 mapping services [Programmableweb, 2021]. The rapid development of web services has brought greater convenience to application developers. Web services have been increasingly used in e-commerce, multimedia services, and automation systems. Web service recommendation refers to the ability to predict the QoS attribute value using data analytics. The prediction results provide the most appropriate web service for users based on the QoS attributes of the service. QoS attribute prediction has become a hot research topic in service computing in recent years. Web services with the same or similar functional attributes can be directly ranked and recommended based on the magnitude of QoS attribute values. Web service recommendation systems usually use the collaborative filtering method to improve the accuracy of recommendation results. At present, with the dimensionality of the dataset increased, Web recommendation systems also apply the tensor decomposition model as an essential data analysis tool for the QoS attribute

prediction. These methods or models are based on a large sample and have been successfully applied to the web service recommendation system.

However, these web recommendation methods are not satisfactory when the user-service-time web recommendation data is sparse. The reason is that the samples in the initial recommendation dataset are only the corresponding values of the existing web services used by the existing users. The QoS attribute prediction is performed based on available historical data. When facing an increase of new users or new web services, no new invocation records are recorded due to the limitations of some conditions. For example, as a result, when the growing service dataset reaches four hundred thousand of service records totally, the corresponding number of samples is only thousands. The prediction values of the newly added services are unknown to the users. This issue will lead to a small sample problem [Stork, Duda, Hart, & Stork, 2001].

Although the traditional tensor decomposition model is a powerful prediction tool for extracting valuable information from sample data, it also does not avoid the shortcoming of low accuracy prediction rate due to small sample data. It is challenged to construct a suitable method to address the small sample problem based on the traditional tensor decomposition model. To the best of our knowledge, there are few methods in the context of the tensor decomposition for QoS attribute prediction. Thus, this thesis focuses on the tensor decomposition and its application as the research object for web service recommendations.

1.2 Problem statement

In general, we study the tensor decomposition to enhance web service recommendation performance. In this area, our research following several open issues: constructing more sample records from the currently limited dataset, applying the regularization term to reduce the possibility of overfitting, and the initial data preprocessing.

We observed the following shortcomings through data analysis:

(1) The number of samples of user-service-time QoS attribute is insufficient. With this limited sample data, the factor matrix iteration remains unchanged. If factor matrices can be reconstructed to speed up the iterative process, the QoS attribute prediction performance can be improved.

(2) Appropriate regularization techniques are needed to solve the overfitting problem for a sparse web service dataset. The ridge regression is popular for tensor decomposition, while the lasso regression has higher efficiency in a sparse dataset.

(3) The preprocessing of initial data affects the quality of web service recommendations. Research has shown that clustering methods are used in the initial unorganized preprocessing step.

In the context of this thesis, we focus on the following details,

Lack of sample data: Constructing the factor matrices from the limited response-time value of existing services that the existing users in the time slice have invoked.

Decision issue of appropriate regularization techniques: Designing a suitable regularization term that applies ridge regression, lasso regression, or both.

Initial unorganized data processing: Clustering the initial unorganized data at preprocessing step to evaluate the efficiency of the recommendation system.

1.3 Thesis contributions

In this thesis, we propose an improved tensor decomposition method, and apply the regularization term and clustering method to the QoS attribute prediction of the Web service recommendation system. The proposed method improves the QoS attribute prediction accuracy to obtain more reasonable and effective web service recommendation results.

The significant contributions of this thesis and corresponding descriptions are listed below.

(1) The traversal-tensor method (TTM), a tensor decomposition algorithm for enhancing sample schemes, is proposed. According to the dimension size of tensor data, this algorithm traversals all features by merging them into tensor decomposition steps. The algorithm aims to address the shortcomings of traditional recommendation methods with low accuracy due to insufficient initial recommendation data samples. The feature factor matrices are based on the current feature sample data to compensate for the missing items in the factor matrix of different time slices. Then the new matrices are applied to the tensor decomposition as a way to improve the prediction rate. The experiments are conducted on the web service WSDream and the traffic prediction datasets. Experimental results validate the effectiveness of the proposed TTM recommendation method.

(2) A modified regularization term incorporating two regular models is proposed to support the TTM method further. The major updates of the modified regularization term are the integration of both the Lasso and ridge expressions. Experimental validation is conducted on the web service WSDream dataset to discuss and evaluate the weights of the different regularization models. Experimental results show that the modified regularization term can increase the correct rate of predicting QoS attributes and better support the TTM web service recommendation method.

(3) Regarding clustering and TTM a two-step strategy is proposed for the initial data preprocessing. Since the K-means algorithm is currently the most widely used clustering algorithm, it is suitable for classification applications with tensor data. We chose a K-means algorithm as preprocessing of TTM. The first step involves a preprocessing process based on a K-means algorithm designed to cluster the initial data to discover the implicit relationships between the data. In the second step, the clustered data objects are used as input and applied to the TTM method to complete the QoS attribute prediction. Experimental results on the relevant dataset show the evaluation between our methods and the clustering method.

In summary, tensor-based modeling for web service recommendations is given to improve recommendation performance. By reconstructing the tensor decomposition model, regularization terms, and updating the clustering method, the high-dimensional data can be expressed more effectively, and the accuracy of QoS attribute prediction is improved.

1.4 Thesis organization

The remainder of this thesis is organized as follows.

Chapter 2 presents a background and an overview of web service recommendations. Basic ideas and applications of tensor representation are discussed. The development process from vector to tensor data processing is introduced. Tensor decomposition and its application in the recommendation system are described.

Chapter 3 introduces the notations and operations of tensor algebra. Then, we describe the tensor decomposition process in the web service recommendation. Moreover, an overview of our research is given.

Chapter 4 formally introduces a new tensor decomposition method, TTM, for QoS attribute prediction. Performance evaluation is given by validation of numerical examples and mathematical explanations. The results of the experiments conducted on real-world datasets and a thorough comparison with the benchmark methods are presented.

Chapter 5 proposes a modified regularization term for supporting TTM. We introduce the lasso and ridge regression separately and formulate the modified regularization term. The experimental results show that it effectively improves the QoS attribute prediction performance.

Chapter 6 proposes a two-step strategy based on the K-means algorithm and TTM to deal with the initial unorganized data. A series of experiments have been conducted to evaluate the effectiveness of the recommendation system combining the clustering technique and tensor decomposition algorithm. We discuss the impacts, such as selecting initial K values, computational performance, and distance calculation.

In Chapter 7, the conclusion and the future research directions are presented.

Chapter 2

Background and Literature Survey

In this thesis, the main technologies involved in web service recommendation research include web services, QoS attribute prediction, and tensor representation and decomposition. This chapter first briefly introduces web services and the QoS attribute prediction techniques of web services. Then, tensor representation and decomposition concepts are introduced.

2.1 Web service

In the service computing and foundation service units, web services are the subject of conceptual expression and operation [Rao & Su, 2005] [Thomas & Immanuel, 2017]. Web service recommendation includes service recommendation based on user functional preferences and service quality [Liu & Fulia, 2015] [Zheng, Ma, Lyu, & King, 2010]. The current research on web service recommendation focuses on accurately predicting the QoS attribute values of users accessing different web services. QoS attributes are used to describe the non-functional attributes of web services, including response time and throughput [O'sullivan, Edmond, & ter Hofstede, 2002]. These QoS attributes are the critical index to measure the quality of web services which can be ranked based on the attribute values.

2.2 QoS attribute prediction

QoS values represent the quantitative measure of the quality of a specific web service. Web service recommendation predicts the QoS attribute value using data analytics. The prediction results provide the most appropriate web service based on the non-functional attributes of the service. These properties accurately reflect the actual performance of the web service. In recent years, QoS attribute prediction has become a major research topic in service computing. For web services with the same or similar functional attributes, the services can be ranked and recommended based on the scale of QoS attribute values.

Predicting the missing QoS attribute is not an easy task. As a metric describing the non-functional attributes of a web service, the QoS attribute is easily influenced by other related information such as the relative user location and the access time. For example, if a user's location is relatively close to the service provider, the service's response time for this user is likely to be shorter. Likewise, if a user accesses a web service during a smooth network time, the network's throughput would be correspondingly higher.

QoS attribute prediction methods primarily utilize a user/service-based collaborative filtering algorithm and a model-based QoS prediction algorithm.

(1) User/service-based collaborative filtering algorithm

Collaborative filtering technology analyzes the access behavior of currently active users and users with similar interests to provide personalized recommendation solutions. The user/service-based collaborative filtering algorithm is the most used recommendation algorithm in personalized recommendation systems, such as Amazon's e-commerce recommendation and movie ranking systems.

User/service-based collaborative filtering algorithm uses the similarity calculation method to calculate the contribution of different users accessing the services to predict the QoS attribute values [Zheng, Ma, Lyu, & King, 2009]. The commonly used similarity calculation methods are listed below,

- User-based collaborative filtering method using Pearson Correlation Coefficient (UPCC): this method generates a prediction based on similar user behavior [Shao et al., 2007].
- Item-based collaborative filtering method using Pearson Correlation Coefficient (IPCC): this method generates a prediction based on similar item properties [Sarwar, Karypis, Konstan, & Riedl, 2001].
- User-based and Item-based Pearson Correlation Coefficient (UIPCC): this is a hybrid collaborative algorithm combining the UPCC and IPCC methods. The prediction is applied to similar users and similar web services.

User/service-based collaborative filtering algorithms are unsuitable for web service recommendation scenarios with large-scale datasets [Yu & Huang, 2017]. Large-scale web service data are often sparse, with many non-zero elements. User/service-based collaborative filtering algorithms depend on historical data to make predictions, and the more historical data there is, the higher the correctness of prediction. Therefore, when encountering large-scale web service data, the prediction performance of these algorithms degrades and is not suitable for web service recommendations.

(2) Model-based QoS attribute prediction algorithm

The model-based QoS attribute prediction algorithm follows machine learning approaches to train the learning model based on the user's historical preferences for items [Ghafouri, Hashemi, & Hung, 2020]. After several iterations of learning to obtain the user's predicted ratings for unrated items, the model-based collaborative filtering algorithm can generally obtain better prediction results. Model-based QoS attribute prediction algorithms include the cluster analysis, matrix factorization [Zheng, Ma, Lyu, & King, 2012][He, Zhu, Zheng, Xu, & Lyu, 2014], and tensor decomposition [Fan, Hu, Zhang, Chen, & Brézillon, 2015][Zhang, Sun, Liu, & Guo, 2014a]. There are two main methods as follows.,

- Probabilistic Matrix Factorization (PMF): a probabilistic method that uses Gaussian assumptions on the data matrices [Mnih & Salakhutdinov, 2007].
- Tensor decomposition: a user-service-time model based on regular tensor decomposition. It predicts the web service QoS attributes by considering the relationship between user, service, and time [Zhang, Sun, Liu, & Luo, 2014b].

Since the model-based QoS attribute prediction algorithms use the entire known evaluation data as information input and then iteratively trains to obtain a prediction model, using this method results in better accuracy [Koren, Bell, & Volinsky, 2009]. However, the direct use of matrix factorization leads to information loss and can affect prediction accuracy. As an extension of matrix factorization, tensor decomposition uses slices in different directions, which preserves the information in each dimension. Thus, tensor decomposition ensures

information integrity. Moreover, the tensor modeling and decomposition techniques can handle high-dimensional data [Karatzoglou, Amatriain, Baltrunas, & Oliver, 2010].

2.3 Tensor representation

Analyzing data requires a method of sorting the data according to a specific representation. The large-scale data generated by various sensors and human activities can be naturally expressed in a two-dimensional or high-dimensional array. For example, the grayscale image (row-column) is a two-dimensional array, whereas the web service data forms in a three-dimensional user-service-time array. The human face image sets consist of four-dimensional array data under different lighting and posture conditions as human-illumination-pose-pixel. The multi-channel Electroencephalograms (EEG) signals are in a six-dimensional array with channel-frequency-time-sample-condition-person features. This method of representing data with multidimensional arrays is very intuitive and convenient.

As an extension of vectors and matrices, tensor refers to a multidimensional array represented by multiple indicators. Mathematically, the strict definition of a tensor is described by a linear map, which refers to a set of ordered numbers that satisfy a specific coordinate conversion relationship when several coordinate systems are transformed [Kolda & Bader, 2009]. Tensor data can be folded into a lower-dimensional vector format in a certain way based on matrix analysis theory. The tensor data with a high-dimensional data structure is not only a simple promotion of adding dimensions based on vectors and matrices, but more importantly, it also has its unique properties and analysis methods. Some researchers conclude that the vectorized format of high-dimensional data presents the small sample problem [Shashua & Levin, 2001] [Wolf, Jhuang, & Hazan, 2007] [Tao, Li, Hu, Maybank, & Wu, 2005]. Reaching a learning performance often requires enough samples as a statistical point of view, but in practical applications, we obtain very few observation samples compared with the high-dimensional data. For example, a 128 × 128 × 3 color image needs to be described by high-dimensional data with a length of 49152

dimensions, but under certain observation conditions, smaller amounts of samples that have less than the data dimension are obtained.

- The high-dimensional data structure of the original data is destroyed.

For example, when a two-dimensional grayscale image is unfolded to a vector format, vectorization destroys the two-dimensional structure information of the image, ignoring the local spatial correlation information.

- The correlation between different modes of data is ignored.

For example, under different user/service locations and accessed times, the web service sample forms a three-dimensional array according to the modes of the user-service-timeline-response time. If we unfold one record into a large one-dimensional vector for processing, we ignore the correlation between different users, different services, and a different time.

When a vector format represents data, the collection of data and linear transformation can be represented by a matrix. Matrix decomposition is a powerful tool for vector data analysis and processing. The conventional methods include Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). The vectorization method is used for tensor data to rearrange the original high-dimensional data in a vector because it can be analyzed and processed by the matrix analysis theory. However, valuable structures and components are often sparsely distributed in the high-dimensional space, and this vectorization method destroys the high-dimensional structure of the original data. Thus, some studies extend the classical theoretical basis for processing vector to tensor algebra for processing tensors with a theoretical foundation for the tensor data analysis. Tensor and its representation can represent and handle high-dimensional data more naturally and intuitively. Google TensorFlow platform uses the tensor representation to store high-dimensional data, which has been widely used [Hao, Liang, Ye, & Xu, 2018]. In general, compared with vector representation methods, tensor representation methods are more benefits as follows

[Shashua & Levin, 2001] [Wolf, Jhuang, & Hazan, 2007] [Tao, Li, Hu, Maybank, & Wu, 2005],

- Tensor representation can maintain the high-dimensional structural characteristics of the data and make full use of the local spatial correlation of the data.
- The dimensionality can be effectively reduced through tensor decomposition, and a more effective data representation can be obtained.

2.4 Decomposition

High-dimensional data can be unfolded to have a low-dimensional structure in the decomposition model. Chandrasekaran et al. prove that high-dimensional data can be expressed as a linear combination of several low-dimensional components [Chandrasekaran, Recht, Parrilo, & Willsky, 2012]. Thus, the decomposition model aims to discover the low-dimensional components.

Matrix decomposition is a typical method for finding low-dimensional components of high-dimensional data, such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). PCA finds a set of projection vectors that project high-dimensional vectors into a low-dimensional space to maximize the squared errors. SVD generally achieves data centralization by decomposing the matrix [Stork, Duda, Hart, & Stork, 2001]. Tensor decomposition is extended from matrix decomposition and processes high-dimensional data based on tensor representation. In the tensor decomposition model, the high-dimensional data represents directly in a tensor format, and the tensor is decomposed into several lower-dimensional format data. Tensor decomposition has the main method: CANDECOMP/PARAFAC (CP) methods [Kolda & Bader, 2009]. Carroll et al. and Harshman studied the rank-one decomposition of tensors and almost independently proposed Canonical Decomposition (CANDECOMP) and Parallel Factor Analysis (PARAFAC) [Carroll, Pruzansky, & Kruskal, 1980] [Harshman, 1970]. These two equivalent decomposition models are called CP decomposition [Kiers, 2000].

Chapter 3

Tensor Decomposition for Web Service Recommendation

In this chapter, we discuss and describe the three-dimensional tensor and its decomposition in the Web service recommendation. An overview of our research will also be given.

3.1 Notations and operations

First, basic concepts and formulas of tensor algebra will be introduced as follows. We follow the notations from the literature of Kolda [Kolda & Bader, 2009].

The tensors are denoted by calligraphic bold capital letters $\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}, \boldsymbol{\mathcal{Y}}$, the capital letters A, B, T, X, U denotes the matrices, and the vectors are denoted by lower-case letters a, b, x, u.

3.1.1 Matrix products

A matrix can be defined as two-dimensional arrays of m rows and n columns. Matrices are denoted by capital letters A. The i_{th} row is denoted by $A_{\text{i}*}$ and the j_{th} column is denoted by $A_{*\text{j}}$. Thus, an m *by* n matrix is

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix}$$

The first subscript on an individual entry in a matrix designates the row that the entry occupies, and the second subscript denotes the column that the entry occupies.

(1) Adding

Each matrix can add other matrices by adding the corresponding entries.

(2) Product

(a) General Matrix Multiplication

Let

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \text{ and } B_{n\times k} = \begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{n1} & b_{n2} & b_{nk} \end{pmatrix}$$

be a $m \times n$ matrix and a $n \times k$ matrix, respectively. Each entry $(AB)_{\text{mk}}$ is given by the result of the scalar product of the m_{th} row of A and the k_{th} column of B, so that

$$(a_{m1}\ a_{m2}\ \ a_{mn})\ (b_{n1}\ b_{n2}\ \ b_{nk}) =< A_{m*}, B_{*k} >$$

$$A_{m\times n}\ B_{m\times n}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{n1} & b_{n2} & b_{nk} \end{pmatrix}$$

$$= \begin{pmatrix} < a_{1*}, b_{*1} > & < a_{1*}, b_{*2} > & < a_{1*}, b_{*k} > \\ < a_{2*}, b_{*1} > & < a_{2*}, b_{*2} > & < a_{2*}, b_{*k} > \\ < a_{m*}, b_{*1} > & < a_{m*}, b_{*2} > & < a_{m*}, b_{*k} > \end{pmatrix}.$$

(b) Hadamard Product: $*$

This matrix product is the elementwise matrix product defined by the French mathematician Jacques Hadamard. Let

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \text{ and } B_{m\times n} = \begin{pmatrix} b_{11} & b_{12} & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ b_{m1} & b_{m2} & b_{mn} \end{pmatrix}$$

be two $m \times n$ matrices. Hadamard product $A * B$ is defined as

$$A_{m\times n} * B_{m\times n}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ b_{m1} & b_{m2} & b_{mn} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & a_{2n}b_{2n} \\ a_{m1}b_{m1} & a_{m2}b_{m2} & a_{mn}b_{mn} \end{pmatrix}.$$

Hadamard product multiplies matrices of the same size, and the resulting matrix has the same size as the original matrices.

(c) Kronecker Product $\otimes$

The Kronecker product multiplies any two matrices of any given size. Let

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \text{ and } B_{j\times k} = \begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix}$$

be an $m \times n$ matrix and a $j \times n$ matrix, respectively. Then the Kronecker product $A \otimes B$ is defined as follows,

$$A_{m\times n} \otimes B_{j\times k} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{12}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{1n}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} \\ a_{21}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{22}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{2n}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} \\ a_{m1}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{m2}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} & a_{mn}\begin{pmatrix} b_{11} & b_{12} & b_{1k} \\ b_{21} & b_{22} & b_{2k} \\ b_{j1} & b_{j2} & b_{jk} \end{pmatrix} \end{pmatrix}.$$

The output product is a matrix of size $(mj) \times (nk)$.

(d) Khatri-Rao Product $\odot$

The Khatri-Rao product multiplies matrices with the same number of columns. Let

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \text{ and } B_{j\times n} = \begin{pmatrix} b_{11} & b_{12} & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ b_{j1} & b_{j2} & b_{jn} \end{pmatrix}$$

be an $m \times n$ matrix and a $j \times n$ matrix, respectively. Then the Khatri-Rao product $A \odot B$ is defined as follows,

$$A_{m\times n} \odot B_{j\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \odot \begin{pmatrix} b_{11} & b_{12} & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ b_{j1} & b_{j2} & b_{jn} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}\begin{pmatrix} b_{11} \\ b_{21} \\ b_{j1} \end{pmatrix} & a_{12}\begin{pmatrix} b_{12} \\ b_{22} \\ b_{j2} \end{pmatrix} & a_{1n}\begin{pmatrix} b_{1n} \\ b_{2n} \\ b_{jn} \end{pmatrix} \\ a_{21}\begin{pmatrix} b_{11} \\ b_{21} \\ b_{j1} \end{pmatrix} & a_{22}\begin{pmatrix} b_{12} \\ b_{22} \\ b_{j2} \end{pmatrix} & a_{2n}\begin{pmatrix} b_{1n} \\ b_{2n} \\ b_{jn} \end{pmatrix} \\ a_{m1}\begin{pmatrix} b_{11} \\ b_{21} \\ b_{j1} \end{pmatrix} & a_{m2}\begin{pmatrix} b_{12} \\ b_{22} \\ b_{j2} \end{pmatrix} & a_{mn}\begin{pmatrix} b_{1n} \\ b_{2n} \\ b_{jn} \end{pmatrix} \end{pmatrix}.$$

The output product is a matrix of size $(mj) \times n$.

Khatri-Rao product and the Kronecker product are identical when considering vectors, i.e., $a \odot b = a \otimes b$.

3.1.2 Definitions

The tensors are denoted by calligraphic bold capital letters $\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}, \boldsymbol{\mathcal{Y}}$, the capital letters T, X, U denote the matrices, and the vectors are denoted by lower-case letters a, b, x, u.

Definition 1 (**Tensor** [Kolda & Bader, 2009])

A tensor is a multidimensional array. More formally, an N-way tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system. An N-way tensor is denoted as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n \times \ldots \times I_N}, (n = 1,2,..,N)$, which has N indices $(i_1 i_2 \ldots i_n \ldots i_N)$ and its elements are denoted by $x_{i_1 i_2 \ldots i_n \ldots i_N}$.

For example, a 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m \times n \times f}$ has three features responded to the three indexes: user, service, and time, as shown in Figure 3.1. The element $(i, j, k) - th$ entry is denoted by x_{ijk}.

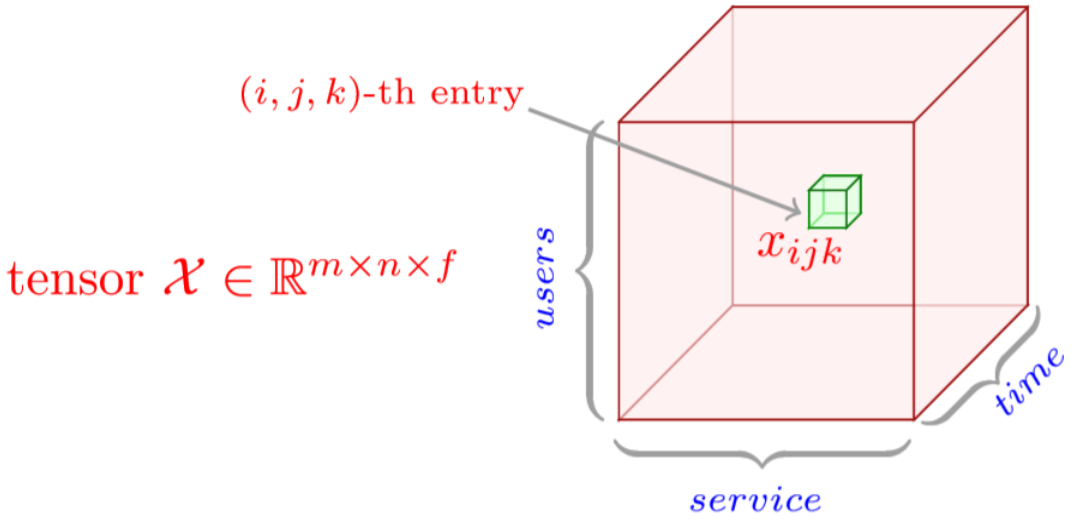


Figure 3.1 A 3-way tensor

Figure 3.2 shows a 4-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{m \times i \times j \times k}}$ which has two group elements $(1, i, j, k) - th$ and $(2, i, j, k) - th$ entries both are in four dimensions.

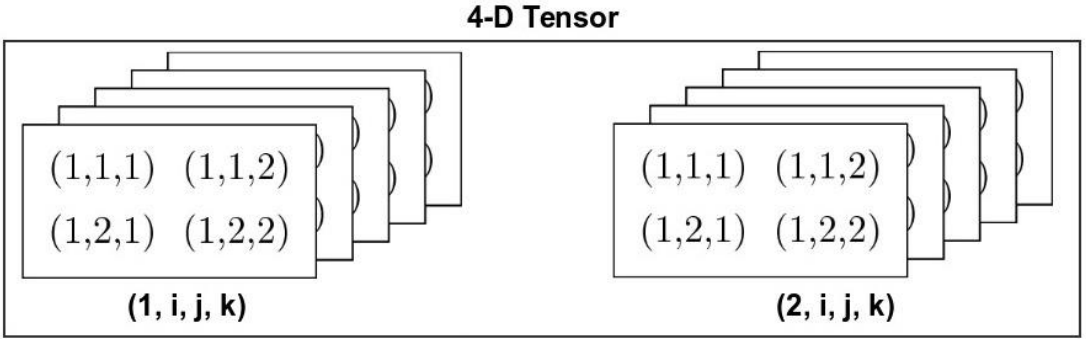


Figure 3.2 A 4-way tensor

A 3-way tensor can be identified by the vectors and matrices representations. A vector obtained by fixing the two of the three indexes of the entries of a tensor is a fiber of a tensor. A matrix obtained by fixing one of the three indexes of the entries of a tensor is a slice of a tensor [Ragnarsson & Van Loan, 2012].

For example, there are three vectors of a 3-way tensor $t_{:jk}, t_{i:k}, and\ t_{ij:}$, the indexes denote i, j, k and colon ":" to represent all other elements of the unfixed index in Figure 3.3.

- The vector $t_{:jk}$ denotes mode-1 fibers when fixing the $j\ and\ k$ indexes (green column).
- The vector $t_{i:k}$ denotes mode-2 fibers when fixing the $i\ and\ k$ indexes (orange row).
- The vector $t_{ij:}$ denotes mode-3 fibers when fixing the $i\ and\ j$ indexes (blue tube).

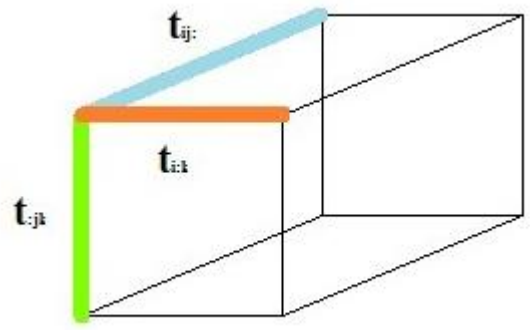
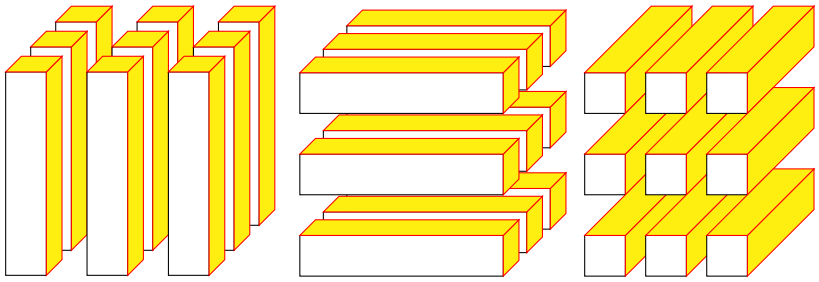


Figure 3.3 Three fibers of a 3-way tensor

All fibers of the 3-way tensor are shown in Figure 3.4 [Cichocki, Zdunek, Phan, & Amari, 2009].



(a) Mode-1 fibers: $t_{:jk}$ (b) Mode-2 fibers: $t_{i:k}$ (c) Mode-3 fibers: $t_{ij:}$

Figure 3.4 All fibers of a 3-way tensor

Furthermore, there are matrices examples of a 3-way tensor $T_{\mathrm{i}::}, T_{:\mathrm{j}:}, and\ T_{::\mathrm{k}}$, where the indexes denote $i, j, k = 1,2$ and full colon ":" to represent all other elements of the unfixed index.

- The matrix $T_{\mathrm{i}::}$ denotes mode-1 slices when fixing the i index (green horizontal slices in Figure 3.5).
- The matrix $T_{:\mathrm{j}:}$ denotes mode-2 slices when fixing the j index (orange lateral slices in Figure 3.6).
- The matrix $T_{::\mathrm{k}}$ denotes mode-3 slices when fixing the k index (blue frontal slices in Figure 3.7).

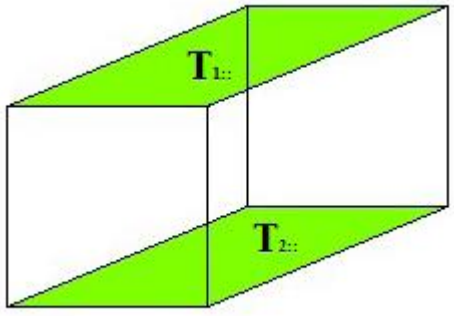


Figure 3.5 Mode-1 slices

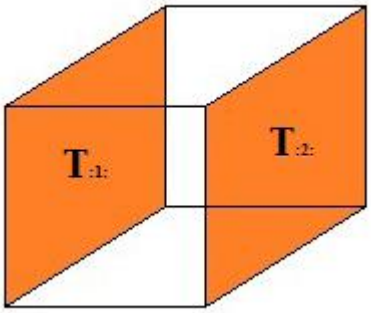


Figure 3.6 Mode-2 slices

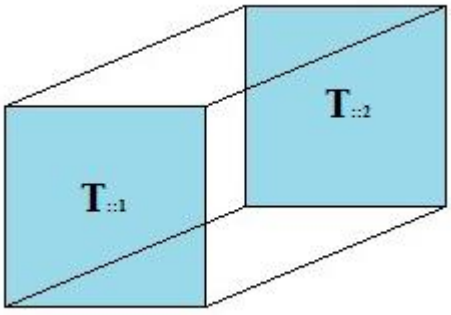
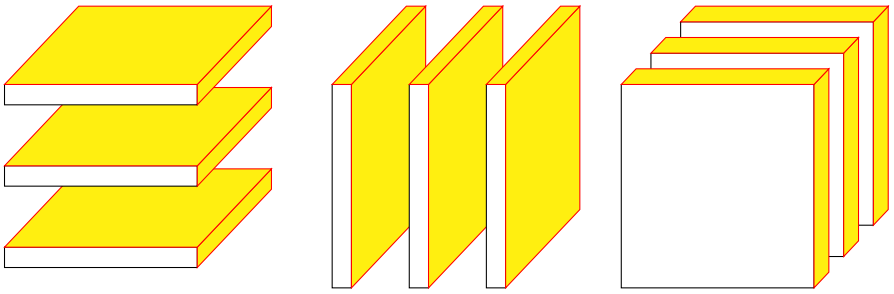


Figure 3.7 Mode-3 slices

Finally, Figure 3.8 shows all horizontal, lateral, and frontal slides of a 3-way tensor $\boldsymbol{\mathcal{X}}$ [Cichocki, Zdunek, Phan, & Amari, 2009].



(a) Horizontal slices: $T_{i::}$ (b) Lateral slices: $T_{:j:}$ (c) Frontal slices: $T_{::k}$

Figure 3.8 All slices of a 3-way tensor

Definition 2 (**Tensor Rank** [Kolda & Bader, 2009])

The tensor rank of a tensor $\boldsymbol{\mathcal{X}}$, denoted tensor rank ($\boldsymbol{\mathcal{X}}$), is defined as the smallest number

of rank-one tensors that generate $\boldsymbol{\mathcal{X}}$ as their sum.

For understanding easily, tensor description in mathematic context is also given as that a *rank-k* tensor in *N*-dimensional space is a mathematical object that has k indices and N^k components. Each component is presented by the vectors. Each index ranges over the number of dimensions. From the view of element, rank is how many indexes are needed to refer to a specific element within the tensor. The number of tensor components is shown in Table 3.1.

Table 3.1 Component numbers of the tensor

		Dimension *N*				
		***N*=1**	***N*=2**	***N*=3**	**...**	***N***
Rank *k*	***k*=1**	1	2	3	…	N^1
	***k*=2**	1	4	9	…	N^2
	***k*=3**	1	8	27	…	N^3
	…	…	…	…	…	…
	k	1^k	2^k	3^k	…	N^k

For example, a 3-way tensor with rank two has twenty-seven components, which means the tensor has three dimensions, and each component of the tensor lies on two indexes.

Definition 3 (**Rank-One Tensors** [Kolda & Bader, 2009])

An *N*-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is *rank-one* when it can be written as the outer product of *N* vectors,

$$\boldsymbol{\mathcal{X}} = u^{(1)} \circ u^{(2)} \circ \ldots \circ u^{(N)} \tag{3.1}$$

where $u^{(n)} \in \mathbb{R}^{I_n}$ $n = 1,2,\ldots,N$ is a vector. The symbol $\circ$ represents the vector outer product. (Equation 3.1 is a formula of tensor decomposition)

This thesis considers only a 3-way tensor with rank-one and its responding methods. If not specified, the default tensor is a rank-one tensor.

Definition 4 (**Norm of a Tensor** [Kolda & Bader, 2009])

The norm of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is the square root of the sum of the squares of all its elements,

$$\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{i_1=1}^{\mathrm{I}_1} \sum_{i_2=1}^{\mathrm{I}_2} \cdots \sum_{i_N=1}^{\mathrm{I}_N} x_{i_1 i_2 \ldots i_N}^2} = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle} \tag{3.2}$$

where $\langle \cdot \rangle$ denotes the inner product of the tensor. $x_{i_1 i_2 \ldots i_N}, all\ i_1, i_2 \ldots i_N = 1,2, \ldots \mathrm{I}_N$, denotes the elements, respectively.

Furthermore, the difference between two tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ is given by $\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\|$.

Definition 5 (**Matricization** [Kolda & Bader, 2009])

Matricization is the process of rearranging the entries of a tensor as a matrix, called unfolding or flattening. When unfolding or flattening the tensor, the mode-n matricization operation maps a tensor into a matrix. For an *N*-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \ldots \times \mathrm{I}_N}$, the mode-$n$ fibers become columns of the unfolding matrix, and the elements of tensor $\boldsymbol{\mathcal{X}}$ is mapped into the mode-n matrix $\mathrm{X}_{(\mathrm{n})} \in \mathbb{R}^{I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots \times I_N)}, n = 1,2,..,N$. The mode-$n$ matrix is the I_n-dimensional matrix obtained from tensor $\boldsymbol{\mathcal{X}}$ by varying the index i_{n} and keeping the other indices fixed.

For example, a $2 \times 2 \times 2$ 3-way tensor $\boldsymbol{\mathcal{X}}$ is shown as follows,

$$\boldsymbol{\mathcal{X}} = \begin{matrix} x_{111} & x_{121} \\ x_{211} & x_{221} \end{matrix} \quad \begin{matrix} x_{112} & x_{122} \\ x_{212} & x_{222} \end{matrix}$$

where $\boldsymbol{\mathcal{X}}$ denotes the tensor, $x_{i_1 i_2 i_3}$ denotes the tensor elements, and $all\ i_1, i_2, i_3 = 1,2$ denote each entry respectively of the tensor's elements.

The mode-n matricizaiton represented as the mode-n matrices $X_{(n)}, n = 1,2,3$ is given by the following,

$$\text{Mode-1 matricizaiton: } \mathrm{X}_{(1)} = \begin{bmatrix} x_{111} & x_{121} & x_{112} & x_{122} \\ x_{211} & x_{221} & x_{212} & x_{222} \end{bmatrix}$$

$$\text{Mode-2 matricizaiton: } \mathrm{X}_{(2)} = \begin{bmatrix} x_{111} & x_{211} & x_{112} & x_{212} \\ x_{121} & x_{221} & x_{122} & x_{222} \end{bmatrix}$$

$$\text{Mode-3 matricizaiton: } \mathrm{X}_{(3)} = \begin{bmatrix} x_{111} & x_{211} & x_{121} & x_{221} \\ x_{112} & x_{212} & x_{122} & x_{222} \end{bmatrix}.$$

3.1.3 Tensor operations

(1) Tensor inner product

The inner product of two same-sized tensors $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \ldots \times \mathrm{I}_N}$ is the sum of their entries' products as follows,

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sqrt{\sum_{i_1=1}^{\mathrm{I}_1} \sum_{i_2=1}^{\mathrm{I}_2} \ldots \sum_{i_N=1}^{\mathrm{I}_N} x_{i_1 i_2 \ldots i_N} \mathrm{y}_{i_1 i_2 \ldots i_N}} \tag{3.3}$$

where $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ denote the two tensors, symbol $\langle \cdot \rangle$denotes the inner product of the tensor, $x_{i_1 i_2 \ldots i_N} \mathrm{y}_{i_1 i_2 \ldots i_N}$ denotes the elements of the $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ tensors respectively, and $i_1 i_2 \ldots i_N$ denote each entry respectively of the $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ tensor's elements.

(2) Tensor outer product

An *N*-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \ldots \times \mathrm{I}_N}$ is rank-one tensor if it can be written as the outer product of *N* vectors, i.e.,

$$\boldsymbol{\mathcal{X}} = u^{(1)} \circ u^{(2)} \circ \ldots \circ u^{(N)} \tag{3.4}$$

where $\boldsymbol{\mathcal{X}}$ denote the tensor, $u^{(n)} \in \mathbb{R}^{\mathrm{I}_n}$ $n = 1,2, \ldots, N$ is a vector, and the symbol $\circ$ represents the vector outer product.

Each element of the tensor $\mathrm{x}_{i_1 i_2 \ldots i_N}$ is the product of the vector sets,

$$\mathrm{x}_{i_1 i_2 \ldots i_N} = u_{i_1}^{(1)} u_{i_2}^{(2)} \ldots u_{i_N}^{(N)} \tag{3.5}$$

where $u_{i_n}^{(n)}$ is the i_n-th element of the vector $u^{(n)}$.

3.1.4 Tensor decomposition

(1) Regular decomposition

CANDECOMP/PARAFAC (CP) decomposition is the primary tensor decomposition method [Kolda & Bader, 2009]. We reference CP decomposition as a regular tensor decomposition (RTD). Its definition is described as following.

For the *N*-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \ldots \times \mathrm{I}_N}$, a regular tensor decomposition is summarized as,

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= u_1^{(1)} \circ u_1^{(2)} \circ \ldots \circ u_1^{(N)} + u_2^{(1)} \circ u_2^{(2)} \circ \ldots \circ u_2^{(N)} + \cdots + u_{\mathrm{R}}^{(1)} \circ u_{\mathrm{R}}^{(2)} \circ \ldots \circ u_R^{(N)} \\ &= \sum_{r=1}^{R} u_{\mathrm{r}}^{(1)} \circ u_{\mathrm{r}}^{(2)} \circ \ldots \circ u_r^{(N)} \end{aligned} \tag{3.6}$$

where $\boldsymbol{\mathcal{X}}$ denote the tensor, $u_r^{(1)}, u_r^{(2)}, \dots, u_r^{(N)}$ are vector set and $u_r^{(1)} \in \mathbb{R}^{I_1}, u_r^{(2)} \in \mathbb{R}^{I_2}, \dots,$ $and\ u_r^{(N)} \in \mathbb{R}^{I_N}$, $r = 1, \dots, R$. R is a positive integer which means the number of vector sets that compose tensor $\boldsymbol{\mathcal{X}}$ when added up. The symbol ° denotes the vector outer product.

We notice that R is not exactly equal to rank value, but at least is the smallest number of rank-1 tensors.

Figure 3.9 illustrates a 3-way tensor decomposition $\boldsymbol{\mathcal{X}} = a \circ b \circ c$, where $a, b, and\ c$ are three vectors, and the symbol ° denotes the vector outer product,

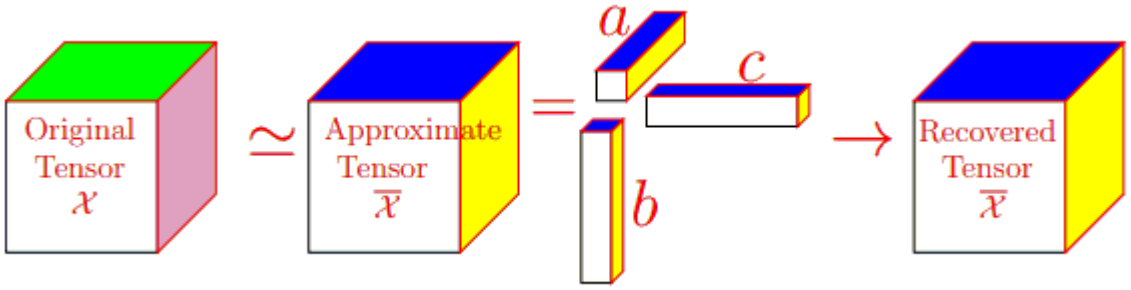


Figure 3.9 A 3-way tensor decomposition

(2) Factor matrices in decomposition

The factor matrix is the combination of the vectors that form the rank-one components. For the N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, the factor matrix is denoted by $U^{(n)} \in \mathbb{R}^{I_N \times R}$ as follows,

$$U^{(n)} = \left[u_1^{(n)}, u_2^{(n)}, \dots, u_R^{(n)}\right], n = 1,2, \dots, N. \tag{3.7}$$

where $\boldsymbol{\mathcal{X}}$ denote the tensor, $U^{(n)}$ denotes the factor matrix, and $u_r^{(n)}$ is the r-th element of the vector $u^{(n)}, n = 1,2, \dots, N\ and\ r = 1,2, \dots, R.$

Following [Kolda, 2006], the regular tensor decomposition model can be expressed as follows,

$$\boldsymbol{\mathcal{X}} = [\![U^{(1)}, U^{(2)}, \dots, U^{(N)}]\!] \tag{3.8}$$

where the symbol $[\![\bullet]\!]$ denotes the collection of factor matrices.

Based on the above equations (3.6) and (3.7), we illustrate how the regular tensor

decomposition converts to the new equation (3.8) as illustrated in Figure 3.10,

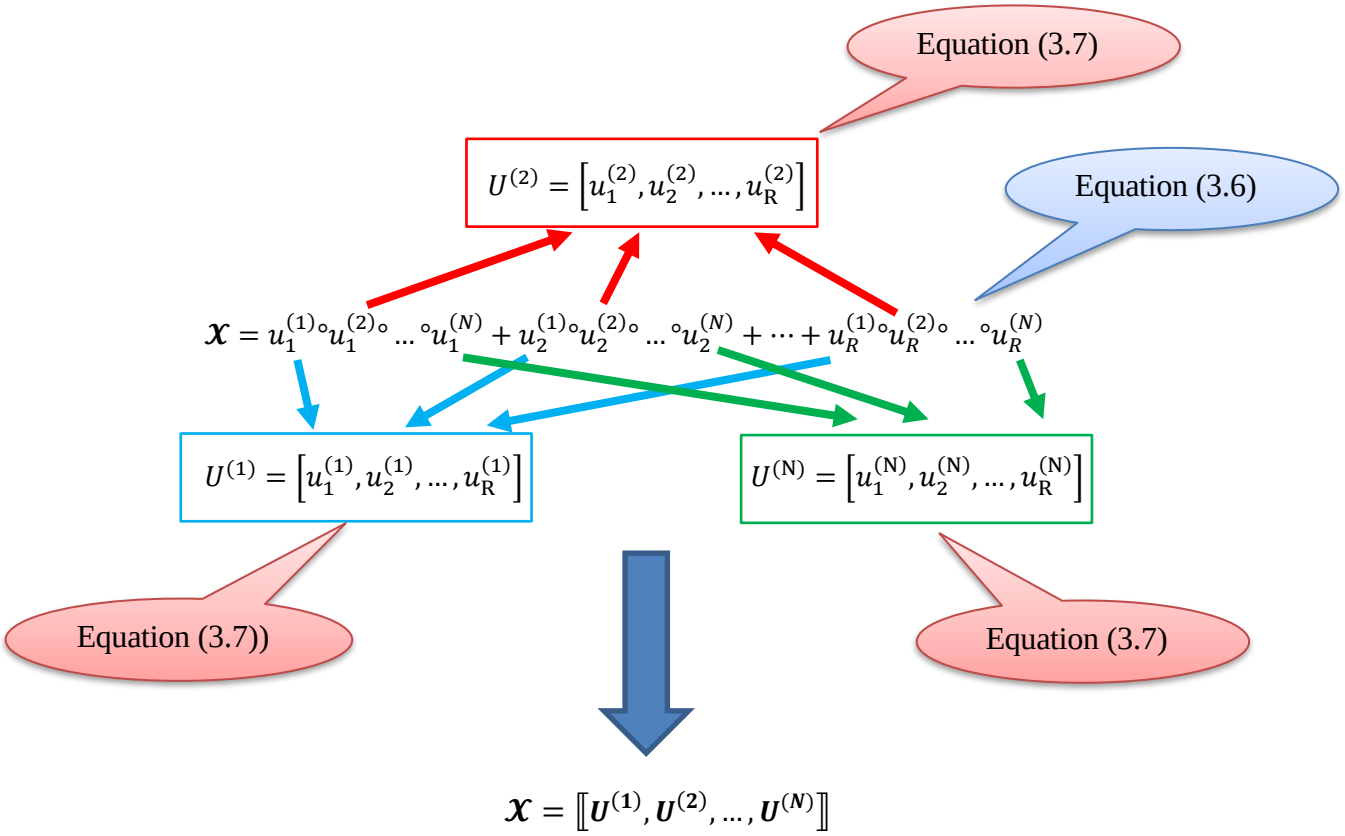


Figure 3.10 Regular tensor decomposition with factor matrix

For example, if we assume 3-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}, N = 3,$ R is the rank of the tensor. After decomposition, the tensor $\mathcal{X}$ consists of one set of 3-way rank-one tensors shown in Table 3.2.

Table 3.2 Examples of regular tensor decomposition with factor matrix

Rank	Decomposition	Factor matrix
1	$\mathcal{X} = u_1^{(1)} \circ u_1^{(2)} \circ u_1^{(3)}$ $= [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$	$U^{(1)} = \left[u_1^{(1)}\right]$, $U^{(2)} = \left[u_1^{(2)}\right]$, $U^{(3)} = \left[u_1^{(3)}\right]$
2	$\mathcal{X} = u_1^{(1)} \circ u_1^{(2)} \circ u_1^{(3)} + u_2^{(1)} \circ u_2^{(2)} \circ u_2^{(3)}$ $= [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$	$U^{(1)} = [u_1^{(1)}, u_2^{(1)}]$, $U^{(2)} = [u_1^{(2)}, u_2^{(2)}]$, $U^{(3)} = [u_1^{(3)}, u_2^{(3)}]$.
3	$\mathcal{X} = u_1^{(1)} \circ u_1^{(2)} \circ u_1^{(3)} + u_2^{(1)} \circ u_2^{(2)} \circ u_2^{(3)} + u_3^{(1)} \circ u_3^{(2)} \circ u_3^{(3)}$ $= [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$	$U^{(1)} = [u_1^{(1)}, u_2^{(1)}, u_3^{(1)}]$, $U^{(2)} = [u_1^{(2)}, u_2^{(2)}, u_3^{(2)}]$, $U^{(3)} = [u_1^{(3)}, u_2^{(3)}, u_3^{(3)}]$.
4	$\mathcal{X} = u_1^{(1)} \circ u_1^{(2)} \circ u_1^{(3)} + u_2^{(1)} \circ u_2^{(2)} \circ u_2^{(3)} + u_3^{(1)} \circ u_3^{(2)} \circ u_3^{(3)} + u_4^{(1)} \circ u_4^{(2)} \circ u_4^{(3)}$ $= [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$	$U^{(1)} = \left[u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, u_4^{(1)}\right]$, $U^{(2)} = \left[u_1^{(2)}, u_2^{(2)}, u_3^{(1)}, u_4^{(2)}\right]$, $U^{(3)} = [u_1^{(3)}, u_2^{(3)}, u_3^{(3)}, u_4^{(3)}]$, $U^{(4)} = [u_1^{(4)}, u_2^{(4)}, u_3^{(4)}, u_4^{(4)}]$.

(3) Optimization of regular tensor decomposition

The regular tensor decomposition aims to find a suitable approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$, which can fit the original tensor $\boldsymbol{\mathcal{X}}$ as much as possible. Thus, the regular tensor decomposition problem can be formulated as an alternating least-squares(ALS) optimization problem,

$$\min_{U^{(1)},U^{(2)},\dots,U^{(N)}} \left(\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\| = \left\|\boldsymbol{\mathcal{X}} - [\![U^{(1)}, U^{(2)}, \dots, U^{(N)}]\!]\right\|^2 \right) \quad (3.9)$$

where $||\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}||^2$ is the tensor norm, $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ denotes the loss function, and $U^{(n)}\ n = 1,2,..N$ denotes the factor matrix.

Following a regular tensor decomposition process [Kolda & Bader, 2009], the n-th frontal slices of an *N*-way tensor are applied in the optimization. Along the mode-n matricization, $\mathrm{X}_{(\mathrm{n})}, n = 1,2,..,N$ denotes as the n-th frontal slices of an *N*-way tensor as follows,

$$\mathrm{X}_{(1)} = \mathrm{U}^{(1)}\left[\mathrm{U}^{(2)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \quad (3.10)$$

$$\mathrm{X}_{(2)} = \mathrm{U}^{(2)}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \quad (3.11)$$

……

$$\mathrm{X}_{(\mathrm{n})} = \mathrm{U}^{(\mathrm{n})}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \mathrm{U}^{(\mathrm{n}-1)} \odot \mathrm{U}^{(\mathrm{n}+1)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \quad (3.12)$$

…

$$\mathrm{X}_{(\mathrm{N})} = \mathrm{U}^{(\mathrm{N})}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \dots \odot \mathrm{U}^{(\mathrm{n})} \odot \dots \odot \mathrm{U}^{(\mathrm{N}-1)}\right]^T \quad 3.13)$$

where the symbol T denotes the matrix transpose, $U^{(n)}\ n = 1,2,..N$ denotes the factor matrix.

Substituting the equations (3.10), (3.11), (3.12), and (3.13), the equation (3.9) is written as follows,

$$\min_{\mathrm{U}^{(1)}} \left\| \mathrm{X}_{(1)} - \mathrm{U}^{(1)}\left[\mathrm{U}^{(2)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \right\| \quad (3.14)$$

$$\min_{\mathrm{U}^{(2)}} \left\| \mathrm{X}_{(2)} - \mathrm{U}^{(2)}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \right\| \quad (3.15)$$

……

$$\min_{\mathrm{U}^{(\mathrm{n})}} \left\| \mathrm{X}_{(\mathrm{n})} - \mathrm{U}^{(\mathrm{n})}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \dots \odot \mathrm{U}^{(\mathrm{n}-1)} \odot \mathrm{U}^{(\mathrm{n}+1)} \odot \dots \odot \mathrm{U}^{(\mathrm{N})}\right]^T \right\| \quad (3.16)$$

…

$$\min_{\mathrm{U}^{(\mathrm{N})}} \left\| \mathrm{X}_{(\mathrm{N})} - \mathrm{U}^{(\mathrm{N})}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \dots \odot \mathrm{U}^{(\mathrm{n})} \odot \dots \odot \mathrm{U}^{(\mathrm{N}-1)}\right]^T \right\|. \quad (3.17)$$

Solving the above equations, the factor matrix $U^{(n)}$ is obtained by iterative solution

formula,

$$\mathrm{U}^{(1)} \leftarrow \mathrm{X}_{(1)} \left[\left[\mathrm{U}^{(2)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \ldots \odot \mathrm{U}^{(\mathrm{N})} \right]^{T} \right]^{+} \tag{3.18}$$

$$\mathrm{U}^{(2)} \leftarrow \mathrm{X}_{(2)} \left[\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \odot \ldots \odot \mathrm{U}^{(\mathrm{N})} \right]^{T} \right]^{+} \tag{3.19}$$

……

$$\mathrm{U}^{(\mathrm{n})} \leftarrow \mathrm{X}_{(\mathrm{n})} \left[\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \ldots \odot \mathrm{U}^{(\mathrm{n}-1)} \odot \mathrm{U}^{(\mathrm{n}+1)} \odot \ldots \odot \mathrm{U}^{(\mathrm{N})} \right]^{T} \right]^{+} \tag{3.20}$$

……

$$\mathrm{U}^{(\mathrm{N})} \leftarrow \mathrm{X}_{(\mathrm{N})} \left[\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \ldots \odot \mathrm{U}^{(\mathrm{n})} \odot \ldots \odot \mathrm{U}^{(\mathrm{N}-1)} \right]^{T} \right]^{+} \tag{3.21}$$

where the symbol $\odot$ denotes the Khatri-Rao product, T denotes the matrix transpose, and the symbol " $+$ " denotes the pseudoinverse. The property of pseudoinverse is as follows [Kolda & Bader, 2009],

$$[[M \odot \mathrm{N}]^{T}]^{+} = [M \odot \mathrm{N}][M^{T}M * N^{T}N]^{+} \tag{3.22}$$

where the symbol denotes the Hadamard product, M and N denote two matrices.

Thus, we rewrite the equations (3.18), (3.19), (3.20), and (3.21) as follows,

$$\mathrm{U}^{(1)} \leftarrow \mathrm{X}_{(1)} \left[\mathrm{U}^{(2)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \ldots \odot \mathrm{U}^{(\mathrm{N})} \right] \left[\left(\mathrm{U}^{(2)} \right)^{T} \mathrm{U}^{(2)} * \left(\mathrm{U}^{(3)} \right)^{T} \mathrm{U}^{(3)} * \ldots * \left(\mathrm{U}^{(\mathrm{N})} \right)^{T} \mathrm{U}^{(\mathrm{N})} \right]^{+} \tag{3.23}$$

$$\mathrm{U}^{(2)} \leftarrow \mathrm{X}_{(2)} \left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(3)} \odot \mathrm{U}^{(4)} \ldots \odot \mathrm{U}^{(\mathrm{N})} \right] \left[\left(\mathrm{U}^{(1)} \right)^{T} \mathrm{U}^{(1)} * \left(\mathrm{U}^{(3)} \right)^{T} \mathrm{U}^{(3)} * \ldots * \left(\mathrm{U}^{(\mathrm{N})} \right)^{T} \mathrm{U}^{(\mathrm{N})} \right]^{+} \tag{3.24}$$

$$\mathrm{U}^{(\mathrm{n})} \leftarrow \mathrm{X}_{(\mathrm{n})} \left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \ldots \odot \mathrm{U}^{(\mathrm{n}-1)} \odot \mathrm{U}^{(\mathrm{n}+1)} \odot \ldots \odot \mathrm{U}^{(\mathrm{N})} \right] \left[\left(\mathrm{U}^{(2)} \right)^{T} \mathrm{U}^{(2)} * \left(\mathrm{U}^{(3)} \right)^{T} \mathrm{U}^{(3)} * \ldots * \left(\mathrm{U}^{(\mathrm{n}-1)} \right)^{T} \mathrm{U}^{(\mathrm{n}-1)} * \left(\mathrm{U}^{(\mathrm{n}+1)} \right)^{T} \mathrm{U}^{(\mathrm{n}+1)} * \ldots * \left(\mathrm{U}^{(\mathrm{N})} \right)^{T} \mathrm{U}^{(\mathrm{N})} \right]^{+} \tag{3.25}$$

$$\mathrm{U}^{(\mathrm{N})} \leftarrow \mathrm{X}_{(\mathrm{N})} \left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \ldots \odot \mathrm{U}^{(\mathrm{n})} \odot \ldots \odot \mathrm{U}^{(\mathrm{N}-1)} \right] \left[\left(\mathrm{U}^{(1)} \right)^{T} \mathrm{U}^{(1)} * \left(\mathrm{U}^{(2)} \right)^{T} \mathrm{U}^{(2)} * \ldots *$$

$$\left(\mathrm{U}^{(\mathrm{n})}\right)^{T}\mathrm{U}^{(\mathrm{n})} * \ldots * \left(\mathrm{U}^{(\mathrm{N}-1)}\right)^{T}\mathrm{U}^{(\mathrm{N}-1)}\Big]^{+}. \tag{3.26}$$

In summary, given an *N*-way tensor $\mathcal{X} \in \mathbb{R}^{\mathrm{I}_1\times\mathrm{I}_2\times\ldots\times\mathrm{I}_N}$, its factor matrix $U^{(n)} \in \mathbb{R}^{I_\mathrm{N}\times\mathrm{R}}$, and frontal slice $\mathrm{X}_{(n)}$ obtained by unfolding along the n*th*-order, the resulting matrix $\mathrm{U}^{(\mathrm{n})}$ describes as follows:

$$\mathrm{U}^{(\mathrm{n})} \leftarrow \mathrm{X}_{(\mathrm{n})}\left[\mathrm{U}^{(1)} \odot \mathrm{U}^{(2)} \odot \ldots \odot \mathrm{U}^{(\mathrm{n}-1)} \odot \mathrm{U}^{(\mathrm{n}+1)} \odot \ldots \odot \mathrm{U}^{(\mathrm{N})}\right]\Big[\left(\mathrm{U}^{(2)}\right)^{T}\mathrm{U}^{(2)} *$$
$$\left(\mathrm{U}^{(3)}\right)^{T}\mathrm{U}^{(3)} * \ldots * \left(\mathrm{U}^{(\mathrm{n}-1)}\right)^{T}\mathrm{U}^{(\mathrm{n}-1)} * \left(\mathrm{U}^{(\mathrm{n}+1)}\right)^{T}\mathrm{U}^{(\mathrm{n}+1)} * \ldots * \left(\mathrm{U}^{(\mathrm{N})}\right)^{T}\mathrm{U}^{(\mathrm{N})}\Big]^{+}. \tag{3.27}$$

3.1.5 Example of regular tensor decomposition

Given a $\mathbf{2}\times\mathbf{2}\times\mathbf{2}$ 3-way tensor $\boldsymbol{\mathcal{X}}$,

$$\boldsymbol{\mathcal{X}} = \begin{matrix} 1 & 2 \\ 3 & 6 \end{matrix} \quad \begin{matrix} 2 & 4 \\ 6 & 12 \end{matrix}.$$

The n-th frontal slice $\mathrm{X}_{(n)}, n = 1,2, and\ 3$ is obtained by the mode-n matricization,

$$\mathrm{X}_{(1)} = \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}$$
$$\mathrm{X}_{(2)} = \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}$$
$$\mathrm{X}_{(3)} = \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}.$$

The initial setup is fixing vectors $b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ in equation (3.21) to compute a vector a_1,

$$a_1 = \mathbf{X}_{(1)}[c_0 \odot b_0][(c_0)^T c_0 * (b_0)^T b_0]^{+}$$
$$= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^{T}\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \end{pmatrix}^{T}\begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]^{+}$$
$$= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}[(1) * (2)]^{+}$$

$$= \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}.$$

Then set vectors $a_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} and\ c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, compute vector b_1,

$$b_1 = \mathbf{X}_{(2)}[c_0 \odot a_1][(c_0)^T c_0 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}^T \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} -1/2 \\ -3/2 \\ 0 \\ 0 \end{bmatrix}[(1) * (5/2)]^+ = \begin{pmatrix} -2 \\ -4 \end{pmatrix}.$$

Then set vectors $b_1 = \begin{pmatrix} -2 \\ -4 \end{pmatrix} and\ a_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}$, compute vector c_1,

$$c_1 = \mathbf{X}_{(3)}[b_1 \odot a_1][(b_1)^T b_1 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} -2 \\ -4 \end{pmatrix}\right.$$

$$\left.\odot \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]\left[\begin{pmatrix} -2 \\ -4 \end{pmatrix}^T \begin{pmatrix} -2 \\ -4 \end{pmatrix} * \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}^T \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 2 \\ 6 \end{bmatrix}[(20) * (5/2)]^+ = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Repeating the above steps to find vectors $a_2, b_2, and\ c_2$, we obtain the vectors set $a_1 = a_2 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}, b_1 = b_2 = \begin{pmatrix} -2 \\ -4 \end{pmatrix}$, and $c_1 = c_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$. An approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ is constructed from the vector set $a, b, and\ c$. Thus, tensor $\boldsymbol{\mathcal{X}}$ can write as $\bar{\boldsymbol{\mathcal{X}}}$,

$$\boldsymbol{\mathcal{X}} = \begin{matrix} & & 2 & 4 \\ & & 6 & 12 \\ 1 & 2 & & \\ 3 & 6 & & \end{matrix},$$

$$\bar{\boldsymbol{\mathcal{X}}} = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} \circ \begin{pmatrix} -2 \\ -4 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 2 \end{pmatrix} == \begin{matrix} 1.0000 & 2.0000 \\ 3.0000 & 6.0000 \end{matrix} \quad \begin{matrix} 2.0000 & 4.0000 \\ 6.0000 & 12.0000 \end{matrix}.$$

3.2 Tensor decomposition processing

The process of web service recommendation is that, given a web services dataset that includes the service invocation history between users and services, the recommendation system recommends an optimized list of services to users.

We consider a web service dataset WSDream describes real-world QoS attribute evaluation results from 142 users on 4,500 web services over 64 different times [Zheng, Ma, Lyu, & King, 2010]. There are four attributes of WSDream highlighted in Table 3.3.

Table 3.3 The attributes of web service dataset WSDream

Attributes	Content	Number
User	all users whom the web services are recommended	142
Service	all web services that can be recommended	4500
Time	the access time when the user requests the service	64
Rating	the QoS attribute value (the response time or throughput) when the user accessing service in the time period	

Each rating value is described by three users, service and time dimensionality as follows,

$$\text{User} \times \text{Service} \times \text{Time} \rightarrow \text{Rating}.$$

Thus, a 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{142\times4500\times64}$ with elements x_{ijk} can be constructed, where x_{ijk} represents the rating value of user i accessing service j at time period k, $\mathrm{i} = 1,..,142, \mathrm{j} = 1,\ldots,4500,$ and $\mathrm{k} = 1,\ldots,64$. The rating value is always positive. If user i has not accessed service j at time period k, then x_{ijk} is null.

Since most users access only a very limited number of web services, the user-service-time tensor $\boldsymbol{\mathcal{X}}$ is a large number of records but sparse tensor. To obtain the missing QoS attribute in the tensor $\boldsymbol{\mathcal{X}}$, the web service QoS attribute can be predicted by the observed service invocation records from the current user.

The main purpose of the tensor decomposition is to iteratively decompose the user-service-time tensor $\boldsymbol{\mathcal{X}}$, obtain the factor matrices $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$, and construct an approximation

tensor $\bar{\mathcal{X}}$ based on $\bar{\mathcal{X}} = [\![U^{(1)}, U^{(2)}, \dots, U^{(N)}]\!]$. The approximation tensor $\bar{\mathcal{X}}$ fills the missing elements to predict the QoS attribute for users accessing different web services at different time periods. The prediction process based on tensor decomposition can be divided into five main steps:

- First, the tensor decomposition rules are defined.

The regular tensor decomposition model of the user-service-time tensor $\mathcal{X} \in \mathbb{R}^{142\times4500\times64}$ with factor matrix $U^{(n)}$ $n = 1,2,3$ is given in the following equation,

$$\mathcal{X} = [\![U^{(1)}, U^{(2)}, U^{(3)}]\!] \tag{3.28}$$

where $U^{(1)}$ is the factor matrix for user i, $U^{(2)}$ is the factor matrix for service j, and $U^{(3)}$ is the factor matrix for time period k.

- Second, the loss function is defined.

Given the user-service-time tensor $\mathcal{X}$ and its approximate tensor $\bar{\mathcal{X}}$, the loss function $\ell(\mathcal{X}, \bar{\mathcal{X}})$ calculates the loss value between two tensors $\mathcal{X}$ and $\bar{\mathcal{X}}$ as following,

$$\ell(\mathcal{X}, \bar{\mathcal{X}}) = \|\mathcal{X} - \bar{\mathcal{X}}\|^2 = \left\|\mathcal{X} - [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]\right\|^2 \tag{3.29}$$

where $U^{(1)}$ is the factor matrix for user i, $U^{(2)}$ is the factor matrix for service j, and $U^{(3)}$ is the factor matrix for time period k.

- Third, the regularization term is added to the loss function.

Tensor decomposition is a complex model. Generally, a more complex model usually leads to overfitting. It might fail to predict future observations reliably [Hawkins, 2004]. A regularization term needs to be added with the original loss function to handle more complex learning tasks to solve this problem. The commonly used regularization terms are the L2 norm of each factor matrix of a tensor. The essential regularization term $\Omega(\bar{\mathcal{X}})$ is shown in the following formula,

$$\Omega(\bar{\mathcal{X}}) = \frac{1}{2}\lambda\left(\left\|U^{(1)}\right\|^2 + \left\|U^{(2)}\right\|^2 + \cdots + \left\|U^{(N)}\right\|^2\right) \tag{3.30}$$

where $\bar{\mathcal{X}}$ denotes an approximate tensor. $U^{(1)}$ is the factor matrix for user i, $U^{(2)}$ is the factor matrix for service j, and $U^{(3)}$ is the factor matrix for time period k. λ are parameters of the factor matrix in the regularization term.

- Fourth, the loss function is optimized and iteratively solved.

The objective function is a function constructed as the sum of lost function and regularization terms. Given an original tensor $\mathcal{X}$ and an approximate tensor $\bar{\mathcal{X}}$, the objective function $L(\mathcal{X}, \mathcal{X})$ is as follows:

Objective function = Loss function + Regularization term

$$\Rightarrow L(\mathcal{X}, \bar{\mathcal{X}}) = \ell(\mathcal{X}, \bar{\mathcal{X}}) + \Omega(\mathcal{X}). \tag{3.31}$$

We perform an optimization task for the objective function $L(\mathcal{X}, \bar{\mathcal{X}}) = \ell(\mathcal{X}, \bar{\mathcal{X}}) + \Omega(\mathcal{X})$ as follows,

$$\min_{U^{(n)}|_{n=1}^{N}} L(\mathcal{X}, \bar{\mathcal{X}}) + \Omega(\mathcal{X})$$

$$\Rightarrow \min_{U^{(1)}, U^{(2)}, U^{(3)}} \left(\frac{1}{2}\|\mathcal{X} - \bar{\mathcal{X}}\|^2\right) + \frac{1}{2}\lambda\left(\left\|U^{(1)}\right\|^2 + \left\|U^{(2)}\right\|^2 + \left\|U^{(3)}\right\|^2\right). \tag{3.32}$$

The alternating least squares (ALS) method is applied to solve this optimization problem. The method fixes $U^{(1)}$ and $U^{(2)}$ to solve for $U^{(3)}$, fixes $U^{(1)}$ and $U^{(3)}$ to solve for $U^{(2)}$, and fixes $U^{(2)}$ and $U^{(3)}$ to solve for $U^{(1)}$. The iteration step is repeated until some convergence criterion is satisfied [Zhang, Sun, Liu, & Guo, 2014b].

- Finally, reconstruct approximate tensor and predict the QoS attribute

According to the inverse operation of tensor decomposition, a user-service-time approximate tensor $\bar{\mathcal{X}}$ is constructed. The prediction QoS attributes are filled by the approximate tensor.

3.3 Overview of the research

The tensor-based modeling for web service recommendation is given as the following, as illustrated in Figure 3.11. By reconstructing decomposition modes, improved regularization terms, and clustering method, the data is expressed as a tensor, and recommendation accuracy can be improved.

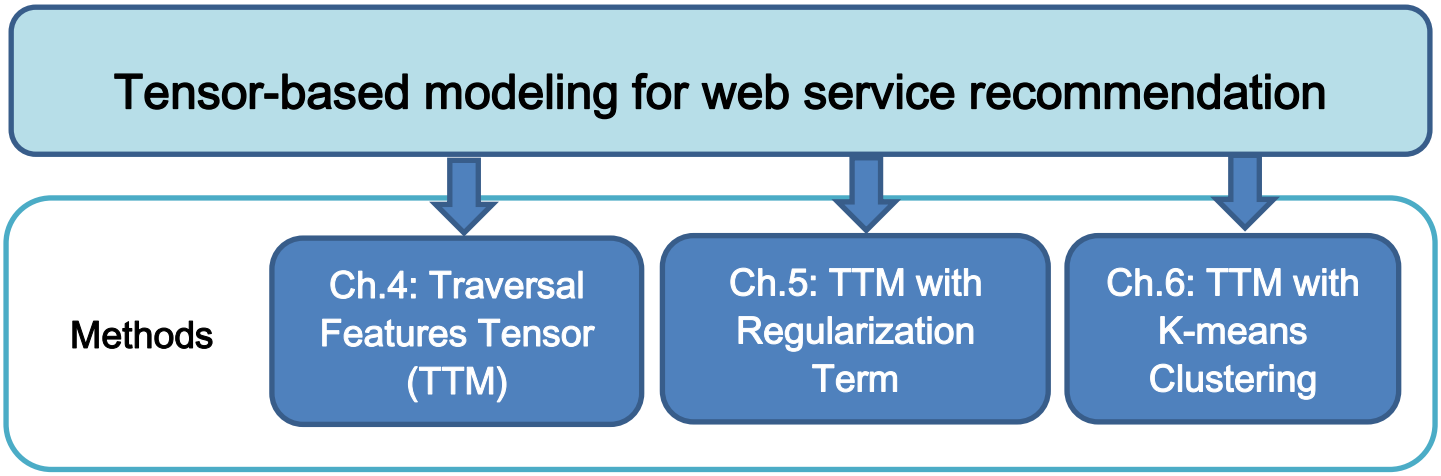


Figure 3.11 Framework of Tensor-based modeling for web service recommendation

(1) Traversal tensor method (TTM)

Given a 3-way original tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, an improved tensor decomposition can be expressed with new factor matrices $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$ as the following equation,

$$\boldsymbol{\mathcal{X}} \approx \bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{\boldsymbol{new}}^{(\boldsymbol{1})}, U_{\boldsymbol{new}}^{(\boldsymbol{2})}, U_{\boldsymbol{new}}^{(\boldsymbol{3})} \right]\!\right] \tag{3.33}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor which consists of QoS attribute value, $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor.

(2) TTM with a modified regularization term

Objective function L can be transformed with the regularization term as follows,

$$L(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) + \Omega(\bar{\boldsymbol{\mathcal{X}}}) \tag{3.34}$$

where $\mathcal{X}$ is an original tensor which consists of QoS attribute value, $\bar{\mathcal{X}}$ denotes an approximation tensor, $\ell(\mathcal{X}, \bar{\mathcal{X}})$ denotes lost function as $\ell(\mathcal{X}, \bar{\mathcal{X}}) = \|\mathcal{X} - \bar{\mathcal{X}}\|^2$, $\Omega(\bar{\mathcal{X}})$ denotes a regularization term.

(3) TTM with K-means algorithm

Using clustering as a preprocessing stage, the K-means algorithm is applied to obtain the clustered tensor, which is used as the input to the TTM as follows,

$$\mathcal{X} \xrightarrow[clustering]{K-means} \bar{\bar{\mathcal{X}}} \xrightarrow{TTM} \bar{\mathcal{X}} \tag{3.35}$$

where $\mathcal{X}$ is an original tensor which consists of QoS attribute value, $\bar{\bar{\mathcal{X}}}$ is a clustered tensor after K-means algorithm, $\bar{\mathcal{X}}$ denotes an approximation tensor.

From the above equations, tensor-based modeling for web service recommendation is summarized as the following:

$$\begin{cases} \mathcal{X} \approx \bar{\mathcal{X}} = \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)} \right]\!\right] \\ \text{improved tensor decomposition} \\ L(\mathcal{X}, \bar{\mathcal{X}}) = \ell(\mathcal{X}, \bar{\mathcal{X}}) + \Omega(\bar{\mathcal{X}}) \\ \text{loss function with a regularization term} \\ \mathcal{X} \xrightarrow[clustering]{K-means} \bar{\bar{\mathcal{X}}} \xrightarrow{TTM} \bar{\mathcal{X}} \\ \text{clustered tensor} \end{cases} \tag{3.36}$$

3.3.1 Traversal tensor method

The motivation of the traversal-tensor method study is to a maximum possible acquisition of features based on a limited number of samples and could be considered to improving QoS attribute prediction. Moreover, the method is also motivated by how regular tensor decomposition factorizes a tensor into a sum of component rank-one tensors [Kolda & Bader, 2009].

The traversal-tensor method defines three feature factor matrices $\Delta U^{(1)}, \Delta U^{(2)}$, and $\Delta U^{(3)}$, and refine the regular factor matrices $U^{(1)}, U^{(2)}, U^{(3)}$ as enhanced factor matrices follow,

$$U_{\boldsymbol{new}}^{(\mathbf{1})} = U^{(1)} + \Delta U^{(1)}, U_{\boldsymbol{new}}^{(\mathbf{2})} = U^{(2)} + \Delta U^{(2)}, and\ U_{\boldsymbol{new}}^{(\mathbf{3})} = U^{(3)} + \Delta U^{(3)}$$

Thus, an approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ satisfies $\bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{\boldsymbol{new}}^{(\mathbf{1})}, U_{\boldsymbol{new}}^{(\mathbf{2})}, U_{\boldsymbol{new}}^{(\mathbf{3})} \right]\!\right]$ and the loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ is modified as follows,

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|^2 = \left\| \boldsymbol{\mathcal{X}} - \left[\!\left[U_{\boldsymbol{new}}^{(\mathbf{1})}, U_{\boldsymbol{new}}^{(\mathbf{2})}, U_{\boldsymbol{new}}^{(\mathbf{3})} \right]\!\right] \right\|^2 \tag{3.37}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor that consists of QoS attribute value, $||\bullet||$ denotes the norm of tensor.

We find the optimal that minimizes the (3.37) equation in every iteration step,

$$\min_{U_{\boldsymbol{new}}^{(\boldsymbol{n})}} \left\| \boldsymbol{\mathcal{X}} - \left[\!\left[(U^{(1)} + \Delta U^{(1)}), (U^{(2)} + \Delta U^{(2)}), (U^{(3)} + \Delta U^{(3)}) \right]\!\right] \right\|. \tag{3.38}$$

In Chapter 4, we discuss the methods in detail and gives the corresponding decomposition algorithm. The traversal-tensor method algorithm is applied to QoS attribute prediction. The experimental results show that the traversal-tensor method improves the prediction performance.

3.3.2 TTM with a modified regularization term

The recommendation performance based on tensor decomposition is usually negatively affected by the overfitting problem and, consequently, cannot achieve state-of-the-art performance. This often requires regularization terms to enhance decomposition performance. The regularization method is part of the loss function in TTM, and its function is to help optimize the loss function and reduce overfitting.

Benefited the advantages of both lasso and ridge regressions, we improve the TTM with a regularization term $\Omega(\bar{\boldsymbol{\mathcal{X}}})$, which the function of proposed is denoted by,

$$\Omega(\bar{\boldsymbol{\mathcal{X}}}) = \lambda\left(\frac{1-p}{2}\ ||\bar{\boldsymbol{\mathcal{X}}}||_2^2 + p||\bar{\boldsymbol{\mathcal{X}}}||_1\right) \tag{3.39}$$

where $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor, $\|\bullet\|$ denotes the tensor norm, $\lambda > 0$ is the regularization parameter. The parameter $p = 0$ corresponds to the ridge method $\|\bullet\|_2$ and $p = 1$ to the lasso method $\|\bullet\|_1$.

To optimize the objective function L as follows,

$$L(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) + \Omega(\bar{\boldsymbol{\mathcal{X}}}) \tag{3.40}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor which consists of QoS attribute value, $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor, $\lambda > 0$ is the regularization parameter, $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ denotes lost function as $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|^2$, $\Omega(\bar{\boldsymbol{\mathcal{X}}})$ denotes a regularization term.

We perform an optimization task for the above objective function in every iteration step as follows,

$$\min_{U_{\boldsymbol{new}}^{(n)}, \boldsymbol{n=1,2,3}} \left(\left\| \boldsymbol{\mathcal{X}} - \left[\!\left[U_{\boldsymbol{new}}^{(1)}, U_{\boldsymbol{new}}^{(2)}, U_{\boldsymbol{new}}^{(3)} \right]\!\right] \right\| \right) + \lambda \left(\frac{1-p}{2} \left\| \left[\!\left[U_{\boldsymbol{new}}^{(1)}, U_{\boldsymbol{new}}^{(2)}, U_{\boldsymbol{new}}^{(3)} \right]\!\right] \right\|_2^2 + p \left|\left| \left[\!\left[U_{\boldsymbol{new}}^{(1)}, U_{\boldsymbol{new}}^{(2)}, U_{\boldsymbol{new}}^{(3)} \right]\!\right] \right|\right|_1 \right) \tag{3.41}$$

where $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor, $||\bullet||$ denotes the norm of tensor, $\lambda > 0$ is the regularization parameter, $\Omega(\bar{\boldsymbol{\mathcal{X}}}) = \Omega\left(\left[\!\left[U_{\boldsymbol{new}}^{(1)}, U_{\boldsymbol{new}}^{(2)}, U_{\boldsymbol{new}}^{(3)} \right]\!\right]\right)$, $U_{\boldsymbol{new}}^{(1)}$, $U_{\boldsymbol{new}}^{(2)}$, and $U_{\boldsymbol{new}}^{(3)}$ are enhanced factor matrices, and p denotes the parameter of the lasso or ridge method.

Chapter 5 applies this optimization algorithm in that one of the decomposition elements is optimized at each iteration when other elements are kept fixed. The experiment result shows that the proposed tensor learning method can effectively improve the estimation performance.

3.3.3 TTM with K-means algorithm

The purpose of clustering preprocessing is to deal with the initial unorganized data in the web service dataset. Since the K-means algorithm is currently the most widely used clustering algorithm, and it is suitable for classification applications with tensor data, we chose a K-means algorithm as a preprocessing method of TTM.

Our proposed two-step strategy method, TTM with K-means algorithm, is applied to fit the QoS attribute prediction scenario as follows,

$$\boldsymbol{\mathcal{X}} \xrightarrow{\boldsymbol{K-means\ clustering}} \bar{\bar{\boldsymbol{\mathcal{X}}}} \xrightarrow{\boldsymbol{TTM}} \bar{\boldsymbol{\mathcal{X}}} \tag{3.42}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor which consists of QoS attribute value, $\bar{\bar{\boldsymbol{\mathcal{X}}}}$ is a clustered tensor after K-means algorithm, $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor.

First, we compute the mean value in the same context cluster $P_{user}, P_{service}, and\ P_{time}$ for prediction $user, service, time$ items separately with an original tensor $\boldsymbol{\mathcal{X}}$. And we apply the $user, service, time$ items to generate a clustered tensor $\bar{\bar{\boldsymbol{\mathcal{X}}}} = [\![U^{(\text{user})}, U^{(\text{service})}, U^{(time)}]\!]$, where $U^{(\text{user})}, U^{(\text{service})}, and\ U^{(time)}$ are the factor matrices after clustering.

Second, this clustered tensor $\bar{\bar{\boldsymbol{\mathcal{X}}}}$ is used as an input tensor of TTM. As same as decomposition processing in Chapter 4, we find the optimal that minimizes the objective function L in every iteration step as follows,

$$L(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) + \lambda\Omega(\bar{\boldsymbol{\mathcal{X}}}) \tag{3.43}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor which consists of QoS attribute value, $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor, $\lambda > 0$ is the regularization parameter, $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ denotes as $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|^2$, $\Omega(\bar{\boldsymbol{\mathcal{X}}})$ denotes a regularization term.

In Chapter 6, we explain the relationship between tensor and K-means algorithm. Furthermore, we introduce the tensor decomposition model based on the K-means

algorithm. The experiment result shows that the algorithm improves the clustering and recognition performance.

3.3.4 Discussion

1) The importance of large samples

The sample size of the data volume is directly related to the prediction accuracy. The reasons are as follows. First, a few categories of samples make it difficult to train a model. Assuming that there are large categories of data, while there are few samples per category or even no sample for some categories, then it is not easy to train a model. Therefore, it is necessary to increase the sample size to cover all data categories.

Second, the model generated by insufficient training samples cannot accurately predict the QoS values. In this case, tensor decomposition can produce serious overfitting problems. Therefore, the training samples should be as complete as possible.

For example, there should be 2.7 billion records in the QoS dataset if all QoS values are fully collected. However, the current dataset only has 30 million records, which is 1% of what it should be. It is not possible to predict the other ninety data records based on just one record. However, the 30 million records of the QoS dataset are already a large amount of data. Therefore, the existing record can help TTM challenge this prediction task even though it is small compared to the whole dataset.

2) Role of TTM in overcoming the cold start problem

In this thesis, poor recommendation performance due to insufficient sample data is also referred to as the cold start problem. The typical approach to solve the cold start problem is to explore new data and utilize existing data. TTM focuses on utilizing existing data, enabling the system to make recommendations using limited QoS data to overcome the cold start problem. We observe that in the three-dimensional web service dataset, service is recommended to a user with a linear relationship between its QoS and the related features.

Moreover, the QoS correlates with the factor matrix composed of all features and the factor matrix composed of the certain feature itself. Based on this observation, the TTM first predicts the QoS value from the three-factor matrices of user-service, user-time, and service-time, which are composed of (user, service, time) triples, as in traditional tensor decomposition methods.

On the other hand, TTM constructs three feature factor matrices, user-user, service-service, and time-time, composed of three features themselves, reusing a limited number of samples. TTM substitutes all the factor matrices into the tensor decomposition iteration process until satisfactory conditions are reached and finally produces new predicted QoS values. In this process, the schema of the feature factor matrix remedy for shortcomings of the limited sample data, allowing the system to collect recommendation data and pass the cold start phase smoothly.

3) TTM and deep learning

With the development of deep learning techniques, neural network-based research has been conducted in the field of QoS prediction in recent years. We also note the application of neural networks in this area. Xiong et al. employed hidden features to calculate the similarity between users and services by introducing a deep learning model [Xiong, Wu, Li, & Gu, 2017]. Wu et al. proposed a generalized deep neural model for QoS prediction using multiple contextual features [Wu, Zhang, Luo, Yue, & Hsu, 2018]. Zhou et al. developed a neural network-based approach to predict the QoS values in a spatial-temporal context [Zhou, Wu, Yue, & Hsu, 2019]. We are trying to implement this approach based on the neural network as one of our future research directions.

Chapter 4

Algorithm for Tensor Decomposition and its Applications

This chapter formally introduces a new tensor decomposition method, TTM, for QoS attribute prediction. Performance evaluation is given by validation of numerical examples, mathematical explanations, and experiments result.

4.1 Introduction

The feature extraction is critical in processing high-dimensional data of the web service recommendation system. As the prediction task, the purpose of feature extraction is to select the appropriate features for the task description. In practical QoS attribute prediction applications, we expect to get enough samples to extract the suitable features. However, when meeting a small sample data, the current feature extraction techniques based on tensor decomposition mainly decompose each dimension's data at once according to the spatial characteristics of multiple dimensions. The regular tensor decomposition model (RTD) lacks the capability cause of relying on limited sample data. To remedy the low prediction accuracy rate caused by the lack of initial data samples, we focus on a tensor decomposition algorithm for small sample data and proposes a feature-oriented decomposition algorithm to solve the above issue. The significant contributions of this chapter are summarized as follows.

- Develop a feature factor matrix as a features-oriented collaboration scheme. The new method integrates the feature factor matrices to construct more data samples for tensor decomposition. The objective of a feature factor matrix is to the maximum possible acquisition of features based on a limited number of samples.
- Perform a novel traversal-tensor method (TTM) by traversing all feature-oriented on tensor decomposition. The key idea is that this method invokes the feature factor matrices based on the regular factor matrices in the decomposition algorithm. Thus, the TTM increases the computational factor in the iteration step for convergence to get a

reasonable result.

- Conduct comprehensive experiments in the real multiple datasets. The experimental results demonstrate that the TTM achieves better prediction performance by deducting the iteration number than the RTD.

The chapter is structured as follows. Section 4.2 introduces the motivation and reviews our previous research work. Section 4.3 describes the new TTM algorithm. We first elaborate on the problem formulation of a feature factor matrix, then present the derivations for modeling the features-oriented collaboration scheme and highlight the differences to emphasize the novelty of this work. Section 4.4 discusses the comparison of TTM and RTD, including its properties, startup issues, computational complexity, and validation of its results. We present experiment setup and performance evaluation results in different real-world datasets in Section 4.5 and Section 4.6. Section 4.7 summarizes the chapter. We also give the supplementary concepts about matrix factorization and regular tensor decomposition as Appendix A and Appendix B.

4.2 Motivation

The web service recommendation needs to extract the user and service historical data and predict the QoS attribute to recommend the relevant service for the user. Many methods for historical data extraction have been developed in the past: collaborative filtering, matrix factorization, and tensor decomposition [BAĞIRÖZ, Güzel, YAVANOĞLU, & Özdemir, 2019]. This thesis is concerned with the tensor decomposition to predict the QoS attribute. Tensor decomposition is a powerful computational tool for extracting valuable information from sample data [Shi, Cheung, Zhao, & Lu, 2018].

The historical data *(u, s, t, R)* predicts the QoS attribute for a future period. The quartet set is represented using a tensor $\boldsymbol{\mathcal{X}}$. Each sample of quartet set represents the response-time *R* of a user *u* to a service *s* invoked at a given time slice *t*. For example, (1,2,3,10) means that the response-time of service #2 invoked by user #1 in time slice #3 is 10 seconds.

By decomposing the tensor $\boldsymbol{\mathcal{X}}$, the corresponding factor matrices are obtained for different time slices. A factor matrix comprises multiple users, multiple services invoked, and response-time. After the outer product operation of the factor matrices, a new tensor is generated by the factor matrices in the iteration. The prediction of the response-time value for the next time slice is achieved.

Zhang *et al*. proposed a tensor decomposition algorithm that is able to deal with the triadic relations of the user-service-time model [Zhang, Sun, Liu, & Guo, 2014a]. Ma *et al*. proposed an integrated QoS attribute prediction method that enables efficient service recommendation for web service-based mobile clients via regular tensor decomposition algorithm [Ma, Wang, Yang, & Chang, 2015]. Their prediction method is considered the access time context, which is from different services by different users. Since these methods focus on only the time attribute or context, Cheng *et al*. propose a QDSHTD prediction method based on a combination of features and contextual information [Cheng et al., 2019]. The experimental results show that the QDSHTD algorithm gives higher estimation accuracy than other QoS attribute prediction algorithms. Although the

successful applications show themselves to be based on a large amount of sample data, these methods above have the same assumption premise that the number of samples used for prediction is sufficient.

However, the analysis reveals that new users will lack response-time value for the service at the time slice. New users may never access the existing service, or new users do not invoke the service at the time slice only. Thus, the number of samples of the quartet set is insufficient. With this limited sample data, the factor matrix remains unchanged, and the fixed factor matrix keeps the iteration processing stable. Factor matrices can be reconstructed if we need to speed up the iterative process to improve prediction performance [Rajih, Comon, & Harshman, 2008]. Rajih *et al*. proposed an enhanced line search (ELS) method based on the current sample in iteration processing. The ELS leads to an optimal solution that applies a summary of the factor matrix and direction indicator.

The idea of the ELS method inspired us to consider a sample reconstruction for modeling a tensor-based pattern. We consider restructuring samples with limited historical data in the iteration processing of the tensor decomposition. The problem to be solved is to complete the reconstruction of factor matrices from the limited response-time value of existing services that the users in the time slice have invoked.

In our previous research, a user collaboration model on tensor decomposition is proposed [Chai, Feng, & Hassanein, 2016]. We analyze the user collaboration tensorial data in tensor decomposition and design a user-oriented structure. The proposed method achieves better prediction accuracy than the regular model. Effectively extracting features computational algorithm from the limited samples deserves a more in-depth study. Our methods significantly reduce the iteration number overhead of the decomposition process for sparse tensors. To the best of our knowledge, this is one of the few methods in the context of the tensor decomposition algorithm.

4.3 The new algorithm: TTM

We start with a brief description of our previous research work that we elaborate a features-oriented collaboration scheme and propose a new algorithm of feature-oriented tensor decomposition TTM. Furthermore, we highlight the differences to emphasize the novelty of this work.

4.3.1 Preliminary result

The feature-oriented tensor data model has been studied in our previous work [Chai, Feng, & Hassanein, 2016]. To overcome the limitation of the CF method, we extend the data from the matrix representation into a three-dimensional tensor model. The model replaces the user-service matrix with user-user-service relations by considering the different QoS attributes for different users. We analyze how user collaboration of related services can help improve tensor decomposition performance. We conduct the experiments to evaluate the performance of the proposed method. The experimental results demonstrate that the proposed method achieves better QoS attribute prediction accuracy than the other methods. The above model reconstructs some sample data based on a single user feature only. If we load more features information as the sample data, or even all the features, would we still improve the prediction accuracy of the recommendation system? With this question in mind, we have done further research in this direction.

4.3.2 Regular tensor decomposition

The details of regular tensor decomposition have been investigated in Kolda's related literature [Kolda, 2006]. More preliminary details are addressed in Appendix B.

Given a 3-way tensor with rank-one $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and an approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ corresponding to $\boldsymbol{\mathcal{X}}$, where $\mathbb{R}$ is for the set of real numbers, I_n $n = 1,2, and\ 3$ is the dimension of the tensor. An approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ from decomposition satisfies the following equation,

$$\boldsymbol{\mathcal{X}} \approx \bar{\boldsymbol{\mathcal{X}}} = [\![U^{(1)}, U^{(2)}, U^{(3)}]\!] \tag{4.1}$$

where the regular factor matrix $U^{(\mathrm{n})}\ n = 1,2, and\ 3$. $U^{(1)}, U^{(2)}, U^{(3)}$ represent the user, service, and time features, respectively. The symbol $[\![\bullet]\!]$ denotes the collection of factor matrices.

The goal of the decomposition of $\boldsymbol{\mathcal{X}}$ is to find the regular factor matrix $U^{(\mathrm{n})}$ that produces the best approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$. The equation (4.1) can be written in a compact form using a Khatri–Rao product $\odot$.

$$\begin{cases} \mathrm{X}_{(1)} \approx U^{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T \\ \mathrm{X}_{(2)} \approx U^{(2)}\left(U^{(1)} \odot U^{(3)}\right)^T \\ \mathrm{X}_{(3)} \approx U^{(3)}\left(U^{(1)} \odot U^{(2)}\right)^T \end{cases} \tag{4.2}$$

where $\mathrm{X}_{(\mathrm{n})}\ n = 1,2, and\ 3$ is the *n*-th frontal slice of the original tensor $\boldsymbol{\mathcal{X}}$, the symbol $\odot$ denotes the Khatri-Rao product, and T denotes the matrix transpose.

The alternating least squares algorithm (ALS) is the most used algorithm for regular tensor decomposition. ALS estimates three factor matrices at each step by minimizing a loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ in the least squares sense the error like the following equation,

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = ||\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}||^2 = ||\boldsymbol{\mathcal{X}} - [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]||^2 \tag{4.3}$$

where $||\bullet||$ denotes the norm of tensor. With regular factor matrices $U^{(2)}$ and $U^{(3)}$ fixed to initial values, the estimate of $U^{(1)}$ is given by,

$$U^{(1)} = \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T$$

$$\Rightarrow U^{(1)} = \mathrm{X}_{(1)}\left[U^{(2)} \odot U^{(3)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(3)}\right)^T U^{(3)}\right]^+ \tag{4.4}$$

where the symbol $+$ denotes the pseudoinverse, and * denotes the Hadamard product.

We estimate regular factor matrices $U^{(2)}$ and $U^{(3)}$ equivalently, with $U^{(2)} = \mathbf{X}_{(2)}\left(U^{(1)} \odot U^{(3)}\right)^T$ and $U^{(3)} = \mathbf{X}_{(3)}\left(U^{(2)} \odot U^{(1)}\right)^T$, and repeat the same steps until the convergence criterion is satisfied.

The regular tensor decomposition algorithm is given in Algorithm 4.1.

Algorithm 4.1: the regular tensor decomposition algorithm

Input: a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \mathrm{I}_3}$, the regularization parameter λ.

Output: the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$, the factor matrices are the index of users, services, and time, respectively.

Step 1. Initialize regular factor matrices $U^{(2)}, U^{(3)}$ and slices $\mathrm{X}_{(1)}, \mathrm{X}_{(2)}, \mathrm{X}_{(3)}$.

Step 2a. Estimate the regular factor matrix $U^{(1)} = \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(3)}\right)^{T}$.

Step 2b. Estimate the regular factor matrix $U^{(2)} = \mathrm{X}_{(2)}\left(U^{(1)} \odot U^{(2)}\right)^{T}$.

Step 2c. Estimate the regular factor matrix $U^{(3)} = \mathrm{X}_{(3)}\left(U^{(1)} \odot U^{(2)}\right)^{T}$.

Step 3. Calculate the approximate tensor $\overline{\boldsymbol{\mathcal{X}}} = [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$.

Step 4. Repeat step 2 to step 3 to update the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$.

Step 5. Calculate the squared error ε to reduce the loss function value until convergence is exhausted.

Step 6. Return the final prediction tensor $\overline{\boldsymbol{\mathcal{X}}}$.

The RTD fixes $U^{(2)}$ and $U^{(3)}$ to find the factor matrix $U^{(1)}$, then fixes $U^{(1)}$ and $U^{(3)}$ to find the matrix $U^{(2)}$, then fixes $U^{(1)}$ and $U^{(2)}$ to find the matrix $U^{(3)}$, and continues to repeat the procedure until the convergence criterion is satisfied.

4.3.3 Features-oriented collaboration scheme

The loops of convergence can take several iterations and needs a large number of iterations [Paatero, 1997]. In each iteration step of the regular tensor decomposition algorithm, we further define feature factor matrices $\Delta U^{(\mathrm{n})}$, and the enhanced factor matrices $\overline{U^{(n)}}$. The feature factor matrix is a scheme that is formulated as a features-oriented collaboration in the iteration step. Furthermore, all enhanced factor matrices would minimize in equation (4.3). The objective is further specified via the following definitions.

Definition 4.1 $\Delta U^{(\mathrm{n})}$ $n = 1, 2, and\ 3$ denotes **feature factor matrices** as a scheme of features-oriented collaboration. Feature factor matrix $\Delta U^{(\mathrm{n})}$ is in connection with other fixed regular factor matrices $U^{(\mathrm{i})}$ ($\mathrm{i} \neq \mathrm{n}$ and $\mathrm{i} = 1, 2, and\ 3$) and n-th frontal slice $\mathrm{X}_{(\mathrm{n})}$ in the t-th (t=1, 2, ….) iteration step.

The feature factor matrix $\Delta U^{(1)}$ can be represented by,

$$\Delta U^{(1)} = \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(2)}\right)^{T} + \mathrm{X}_{(1)}\left(U^{(3)} \odot U^{(3)}\right)^{T} \tag{4.5}$$

where $U^{(2)}$ and $U^{(3)}$ denote regular factor matrices, and $\mathrm{X}_{(1)}$ denotes 1st frontal slice. $\Delta U^{(1)}$, $U^{(1)}$, and $\mathrm{X}_{(1)}$ are in the same iteration step. T denotes the matrix transpose.

The feature factor matrix $\Delta U^{(2)}$ and feature factor matrix $\Delta U^{(3)}$ are obtained equivalently as follows,

$$\Delta U^{(2)} = \mathrm{X}_{(2)}\left(U^{(1)} \odot U^{(1)}\right)^{T} + \mathrm{X}_{(2)}\left(U^{(3)} \odot U^{(3)}\right)^{T} \tag{4.6}$$

$$\Delta U^{(3)} = \mathrm{X}_{(3)}\left(U^{(1)} \odot U^{(1)}\right)^{T} + \mathrm{X}_{(3)}\left(U^{(2)} \odot U^{(2)}\right)^{T} \tag{4.7}$$

where $\mathrm{X}_{(2)}$ denotes 2nd frontal slice, $\mathrm{X}_{(3)}$ denotes 3rd frontal slice.

Definition 4.2 $\overline{U^{(n)}}$ $n = 1,2, and\ 3$ denotes **enhanced factor matrices** as a summary of regular factor matrices $U^{(\mathrm{n})}$ and feature factor matrices $\Delta U^{(\mathrm{n})}$,

$$\overline{U^{(1)}} = U^{(1)} + \Delta U^{(1)} \tag{4.8}$$

where $U^{(1)}$ is regular factor matrices. $\Delta U^{(1)}$ is feature factor matrices. $\Delta U^{(1)}$ and $U^{(1)}$ are obtained in the same t-th (t=1, 2, ….) iteration step. $\overline{U^{(1)}}$ is an enhanced factor matrix that will be applied in the $(t+1)$-th (t=1, 2, ….) iteration step instead of $U^{(\mathrm{n})}$.

With the enhanced factor matrix $\overline{U^{(1)}}$, the enhanced factor matrices $\overline{U^{(2)}}$ and $\overline{U^{(3)}}$ are obtained equivalently as follows,

$$\overline{U^{(2)}} = U^{(2)} + \Delta U^{(2)} \tag{4.9}$$

$$\overline{U^{(3)}} = U^{(3)} + \Delta U^{(3)} \tag{4.10}$$

where $U^{(2)}$ and $U^{(3)}$ denote regular factor matrices. $\Delta U^{(2)}$ and $\Delta U^{(3)}$ denote feature factor matrices.

In the absence of more sample data, we expect to exhaust the association of each factor matrix with all other features to make up for the loss of information due to the lack of sample data. Therefore, the feature factor matrices load a mapping collaboration between each regular factor matrix and other features separately, potentially describing the tensor data object more accurately and increasing the iteration step's size.

4.3.4 Example of remedy insufficient samples

We illustrate two-dimensional and three-dimensional datasets to help clearly understand the proposed feature-oriented collaborations by the user-oriented, service-oriented, and time-oriented collaborations separately.

(1) Two-dimensional data

Let us focus on two-dimensional data first. Given a *user-service* matrix, we predict the missing QoS attribute relevant to user and service based on the current value's observations. For example, we set four records in an original web service dataset, such as Table 4.1. The *response-time* value of 10 seconds reflects the performance when *user 1* requesting *service 1.* The *response-time* value 20 and 30 seconds are generated by *user 3* requesting *service 2* and *3,* respectively. The last response-time value of 40 seconds is generated by *user 4* requesting *service 4*.

Table 4.1 Four records in web service dataset

Users	Services	Response-time
user1	service1	10 seconds
	service 4	40 seconds
user3	service 2	20 seconds
	service 3	30 seconds

From the records of Table 4.1, we obtain the *(user, service, response-time)* full records have three users, four services, and four response-time values in Table 4.2.

Table 4.2 (user, service, response-time) full records

User	Service	Response-time (seconds)	Sample records
1	**1**	**10**	Sample
2	1		
3	1		
1	2		
2	2		
3	**2**	**20**	Sample
1	3		
2	3		
3	**3**	**30**	Sample
1	**4**	**40**	Sample
2	4		
3	4		

There are two dimensions in these full records: user and service, three per user dimension, and four per *service* dimension. Based on the number of features and dimensions, there should be 12 records, but only four are observed as the sample. It means that the 8 (=12-4) latent records should be predicted in Table 4.3.

Table 4.3 Statistics of (user, service, response-time) records

Dimensions	Features	Sample records	Latent records
User	3	4	8
Service	4		

The prediction encounters a problem of how to predict the latent records based on the limited sample records. If prediction uses more samples, the prediction performance will be enhanced more. To solve this problem, we proposed a reconstruction method to obtain more sample-based on historical sample records.

First, we extract the sample records from a full recordset as follows in Table 4.4.

Table 4.4 (user, service, response-time) sample records

Users	Service	Response-time (seconds)
1	1	10
3	2	20
3	3	30
1	4	40

Moreover, we construct two new records sets by using only sample records, based on the above *(user, service, response-time)* record set in Table 4.4,

- (*user, user, response-time*) recordset

We replace *(user, service)* columns with *(user, user)* columns when fixing the *response-time* value in sample records to construct a new record set in Table 4.5.

- (*service, service, response-time*) recordset

Following the same idea, we replace *(user, service)* columns with a *(service, service)* column when fixing the *response-time* value in sample records to construct a new record set in Table 4.6.

We construct two new matrixes: the user-based matrix *(user, user)* and service-based matrix *(service, service),* respectively, based on the original matrix. Given a *user-service* matrix, the *response-time* value can be predicted by (*user-service)* as an original matrix. We proposed a new construction that combines two new matrixes and the original matrix to enhance the prediction performance as follows,

((user, service), (user, user), (service, service)) --> response-time

where the *user* and *service* are two dimensions, *(user, service)* is an original matrix, *(user, user)* is a user-based matrix, *(service, service)* is a service-based matrix.

Table 4.5 (user, user, response-time) sample records

User	User	Response-time (seconds)	Sample records
1	1	10	Sample
3	3	20	Sample
3	3	30	Sample
1	1	40	Sample

Table 4.6 (service, service, response-time) sample records

Service	Service	Response-time (seconds)	Sample records
1	1	10	Sample
2	2	20	Sample
3	3	30	Sample
4	4	40	Sample

To illustrate the above construct, the three-dimensional mapping is shown as follows in Figure 4.1. The *(user, service, response-time)* recordset is mapped in two records: *(user, user, response-time)* and *(service, service, response-time)*.

In Figure 4.1, we can see that the *response-time* value mapping reflects different distributions when a viewpoint is based on a different direction as the coordinate level. For example, when the viewpoint is based on the *user-user* matrix, the mapping of four *response-time* values will be distributed on two straight lines with *user=1* and *user=3*, and when the viewpoint is based on the *service-service* matrix, the mapping of four *response-time* values will be distributed on four different straight lines with *service=1,2,3,* and *4*.

Thus, the *response-time* value distribution in the three-dimensional data transfer to a two-dimensional space, which is mapped from the user and service directions separately. We use three sets to predict latent *response-time* value: a *(user, service, response-time)* set, a *(user, user, response-time)* set, and a *(service, service, response-time)* set.

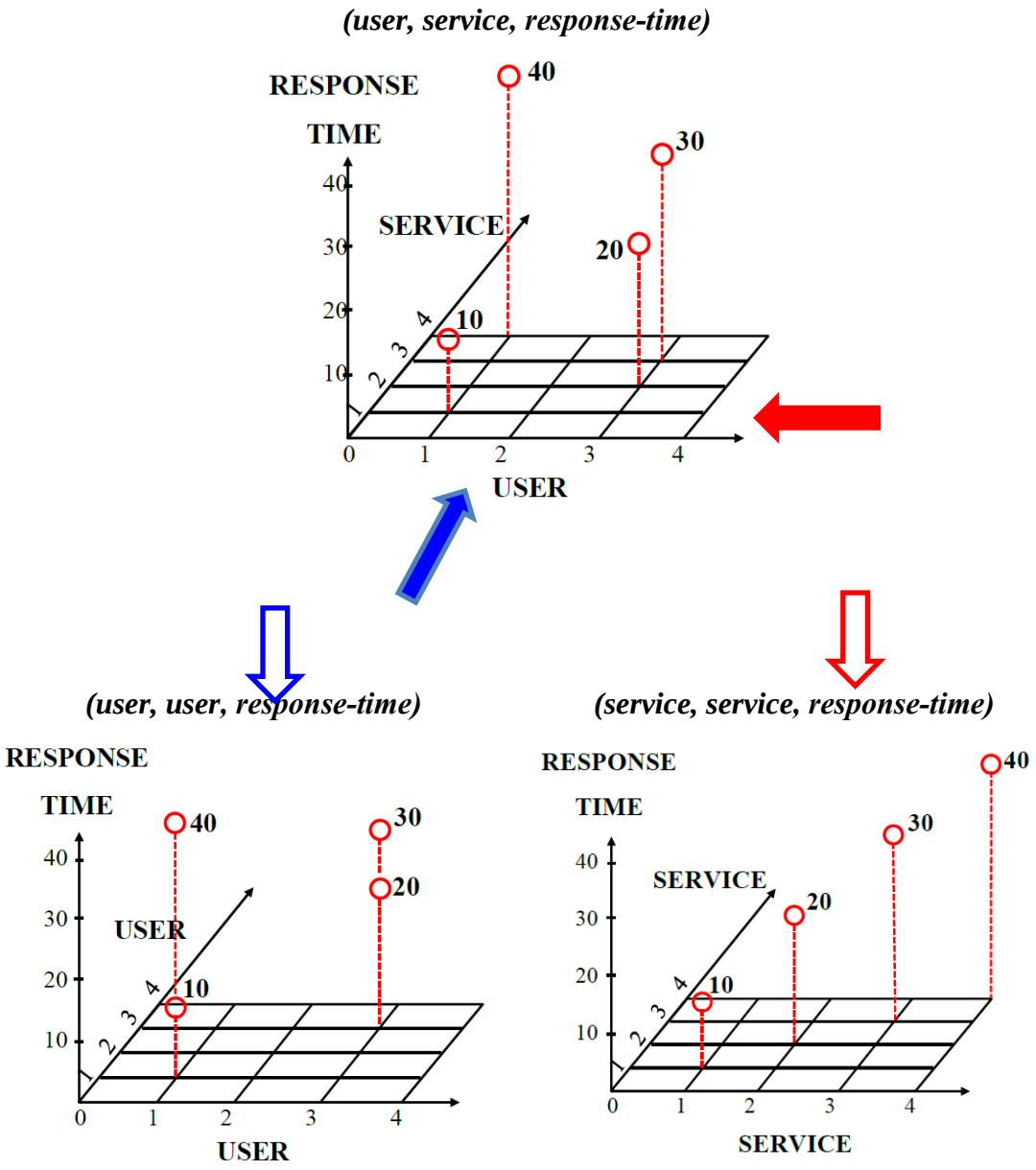


Figure 4.1 A three-dimensional data mapping

(2) Three-dimensional data

Let us apply the reconstruction idea to three-dimensional data. Given a *user-service-time* tensor, the missing QoS attribute is predicted with three dimensions: *user, service,* and *time,* based on the current value observations. For example, the *(user, service, time, response-time)* tensor has three users, four services, and two times for seven response-time values in Table 4.7.

Table 4.7 (user, service, time, response-time) tensor full records

User	Service	Time	Response-time (seconds)	Sample records
1	1	1	10	Sample
1	1	2		
1	2	1		
1	2	2	50	Sample
1	3	1		
1	3	2		
1	4	1	40	Sample
1	4	2		
2	1	1		
2	1	2	60	Sample
2	2	1		
2	2	**2**		
2	3	**1**		
2	**3**	2		
2	4	1		
2	4	2		
3	**1**	1		
3	1	**2**		
3	2	**1**	20	Sample
3	2	2		
3	3	1	30	Sample
3	**3**	2		
3	4	1		
3	4	**2**	70	Sample

Observing Table 4.7, the *response-time* value of 10 seconds reflects the performance when user *1* requesting *service 1 at time 1.* The response-time values 20 and 30 seconds are generated by *user 3* requesting *service 2* and *3 at time 1*. The response-time value of 40

seconds is generated by *user 4* requesting *service 4 at time 1*. Moreover, the response-time value is 50 seconds when user 1 requests service 2 at time 2. The response-time value of 60 seconds is generated by *user 2* requesting *service 1 at time 2*. At the same time 2, the response-time value of 70 seconds is generated by *user 3* requesting *service 4*. The value of latent records number 17 (=24-7) should be predicted in Table 4.8.

Table 4.8 Statistics of a (user, service, time, response-time) tensor

Dimension's name	**Features number**	**Sample records number**	**Latent records number**
User	3	7	17
Service	4		
Time	2		

Following the same idea, we extract the sample records from a tensor as follows in Table 4.9.

Table 4.9 (user, service, time, response-time) tensor samples records

User	**Service**	**Time**	**Response-time (seconds)**	**Sample records**
1	1	1	10	Sample
1	2	2	50	Sample
1	4	1	40	Sample
2	1	2	60	Sample
3	2	1	20	Sample
3	3	1	30	Sample
3	4	**2**	70	Sample

Moreover, we construct seven new tensors using only sample records, based on the above *(user, service, time, response-time)* tensor samples in Table 4.9.

We replace *(user, service, time)* columns with *(user, user, service)* columns when fixing the *response-time* value in sample records to construct a new record set as follows in Table 4.10.

The rest can be done with the same processing, and we obtain the six new tensor constructions as follows in Tables 4.11 to 4.15.

- *(user, user, service, response-time)* tensor
- *(user, user, time, response-time)* tensor
- *(service, service, user, response-time)* tensor
- *(service, service, time, response-time)* tensor
- *(time, time, user, response-time)* tensor
- *(time, time, service, response-time)* tensor

Table 4.10 (user, user, service, response-time) tensor samples records

User	**User**	**Service**	**Response-time (seconds)**	**Sample records**
1	1	1	10	Sample
1	1	2	50	Sample
1	1	4	40	Sample
2	2	1	60	Sample
3	3	2	20	Sample
3	3	3	30	Sample
3	3	4	70	Sample

Table 4.11 (user, user, time, response-time) tensor samples records

User	**User**	**Time**	**Response-time (seconds)**	**Sample records**
1	1	1	10	Sample
1	1	2	50	Sample
1	1	1	40	Sample
2	2	2	60	Sample
3	3	**1**	20	Sample
3	3	1	30	Sample
3	3	**2**	70	Sample

Table 4.12 (service, service, user, response-time) tensor samples records

Service	Service	User	Response-time (seconds)	Sample records
1	1	1	10	Sample
2	2	1	50	Sample
4	4	1	40	Sample
1	1	2	60	Sample
2	2	3	20	Sample
3	3	3	30	Sample
4	4	3	70	Sample

Table 4.13 (service, service, time, response-time) tensor samples records

Service	Service	Time	Response-time (seconds)	Sample records
1	1	1	10	Sample
2	2	2	50	Sample
4	4	1	40	Sample
1	1	2	60	Sample
2	2	**1**	20	Sample
3	3	1	30	Sample
4	4	**2**	70	Sample

Table 4.14 (user, time, time, response-time) tensor samples records

User	Time	Time	Response-time (seconds)	Sample records
1	1	1	10	Sample
1	2	2	50	Sample
1	1	1	40	Sample
2	2	2	60	Sample
3	**1**	**1**	20	Sample
3	1	1	30	Sample
3	**2**	**2**	70	Sample

Table 4.15 (service, time, time, response-time) tensor samples records

Service	Time	Time	Response-time (seconds)	Sample records
1	1	1	10	Sample
2	2	2	50	Sample
4	1	1	40	Sample
1	2	2	60	Sample
2	**1**	**1**	20	Sample
3	1	1	30	Sample
4	**2**	**2**	70	Sample

Finally, we divide the six tensors into three groups:

- **User-based**: *(user, user, service, response-time)* and *(user, user, time, response-time)*
- **Service-based**: *(service, service, user, response-time)* and *(service, service, time, response-time)*
- **Time-based**: *(time, time, user, response-time)* and *(time, time, service, response-time)*

We proposed a new construction that combines six new tensors and the original tensor to enhance the prediction performance as follows,

((user, service, time), (user, user, service), (user, user, time), (service, service, user), (service, service, time), (time, time, user), (time, time, service))

→ response-time

where the *user*, *service*, and *time* are three dimensions, *(user, service, time)* is an original tensor, *(user, user, service/time)* are User-based tensors, *(service, service, user/time)* are Service-based tensors, *(time, time, user/service)* are Time-based tensors.

Since the four-dimensional construction cannot be directly depicted, we illustrate the above construction by the flattening tables. For example, we construct the new tensor *(user, user, service, response-time)* from the original tensor *(user, service, time, response-time)* as in Figures 4.2 and 4.3.

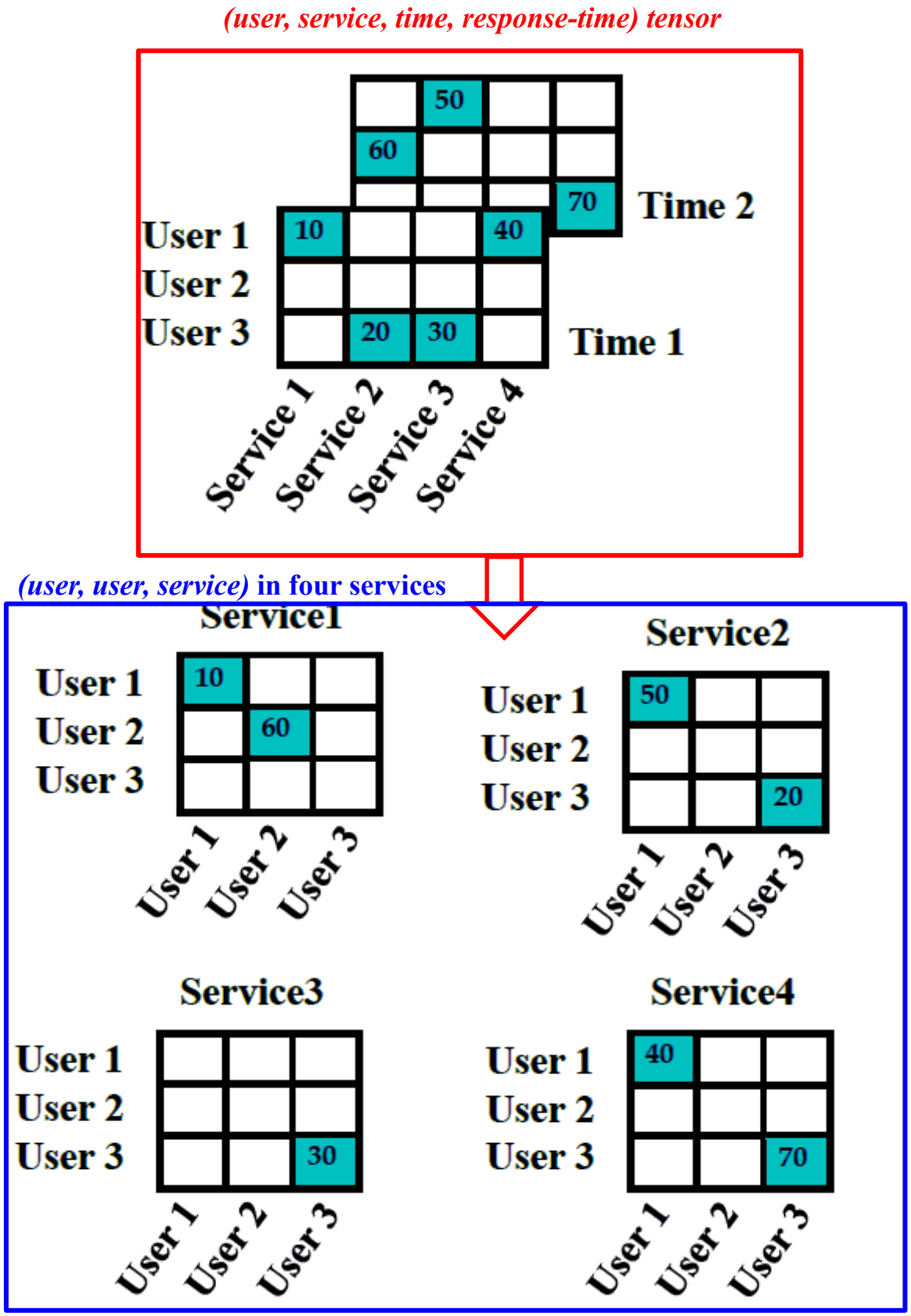


Figure 4.2 (user, user, service, response-time) tensor data mapping

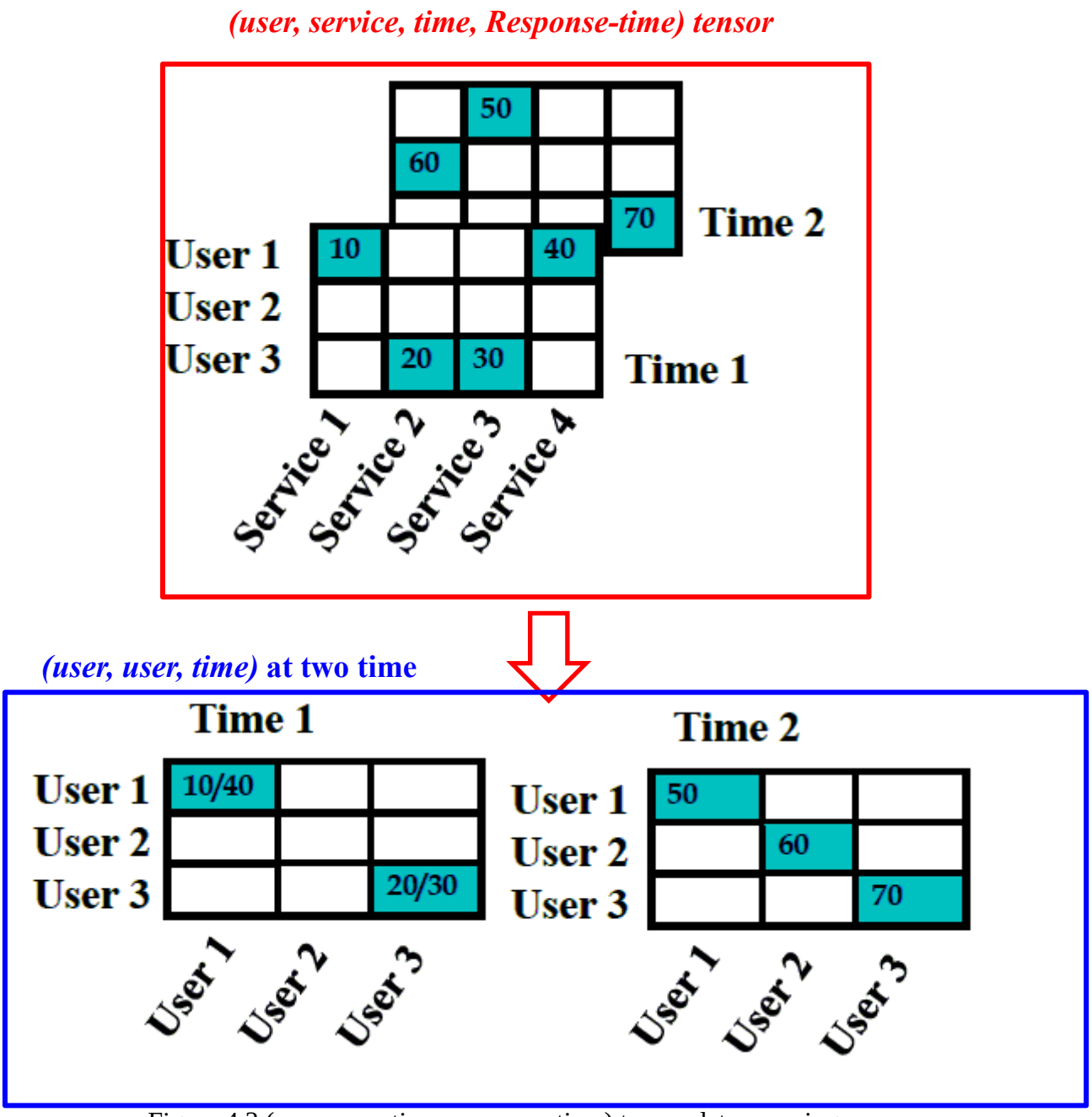


Figure 4.3 (user, user, time, response-time) tensor data mapping

Finally, the rest of the tensors can be done in the same processing to obtain the six new tensor constructions, and we predict latent *response-time* value based on a *(user, service, time, response-time)* tensor with the *user-based, service-based, time-based* tensors.

4.3.5 Traversal-tensor method (TTM)

We proposed an improved tensor decomposition method, TTM. The TTM is based on the above definition and determination of increments to all three factor matrices. It consists of repeatedly solving for the feature factor matrices $\Delta U^{(1)}$, $\Delta U^{(2)}$, and $\Delta U^{(3)}$, and the enhanced factor matrices $\overline{U^{(1)}}$, $\overline{U^{(2)}}$, and $\overline{U^{(3)}}$.

Considering that the newly generated factor matrix may cause the approximation tensor to be farther away from the original tensor, we need to set up the control mechanism.

Given three new factor matrices $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$, and two errors ε and ε_{new} are denoted as follows,

$$\varepsilon = \left\| \mathrm{X}_{(\mathrm{n})} - \overline{\mathrm{X}_{(\mathrm{n})}} \right\|^2,\ n = 1,2, and\ 3 \tag{4.11}$$

$$\varepsilon_{new} = \left\| \mathrm{X}_{(\mathrm{n})} - \overline{\mathrm{X}_{(\mathrm{n})}}' \right\|^2,\ n = 1,2, and\ 3 \tag{4.12}$$

where

$\mathrm{X}_{(\mathrm{n})}\ n = 1,2, and\ 3$ denotes the *n*-th frontal slice of the original tensor $\boldsymbol{\mathcal{X}}$,

$\overline{\mathrm{X}_{(\mathrm{n})}}\ n = 1,2, and\ 3$ denotes the *n*-th frontal slice of the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ ($\bar{\boldsymbol{\mathcal{X}}}$ is generated by the regular factor matrices $U^{(\mathrm{n})}\ n = 1,2, and\ 3$),

$\overline{\mathrm{X}_{(\mathrm{n})}}'\ n = 1,2, and\ 3$ denotes the *n*-th frontal slice of the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}'$ ($\bar{\boldsymbol{\mathcal{X}}}'$ is generated by the enhanced factor matrices $\overline{U^{(\mathrm{n})}}\ n = 1,2, and\ 3$).

At every step of updating the factor matrix, we calculate two squared errors ε and ε_{new} separately, then compare these two errors:

If $\varepsilon_{new} \geq \varepsilon$, this indicates that the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}'$ is further away from the original tensor $\boldsymbol{\mathcal{X}}$ than another approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$. The current iterative step size is slightly large, and the iteration seems to be going in the wrong direction. To keep

the optimization direction of the iteration, the regular factor matrices $U^{(\mathrm{n})}\ n = 1,2, and\ 3$ are set as new factor matrix $U_{new}^{(n)}\ n = 1,2, and\ 3$ at the next step.

If $\varepsilon_{new} < \varepsilon$, the current computation will continue, and the enhanced factor matrices $\overline{U^{(n)}}\ n = 1,2, and\ 3$ as new factor matrix $U_{new}^{(n)}\ n = 1,2, and\ 3$ at the next step.

This judgment is made whenever a new factor matrix is computed to ensure that the iteration goes down correctly.

Finally, an approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ satisfies $\bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]$ and the loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ of TTM is modified as follows,

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|^2 = \left\|\boldsymbol{\mathcal{X}} - \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]\right\|^2$$

$$= \left\|\boldsymbol{\mathcal{X}} - \left[\!\left[(U^{(1)} + \Delta U^{(1)}), (U^{(2)} + \Delta U^{(2)}), (U^{(3)} + \Delta U^{(3)})\right]\!\right]\right\|^2. \qquad (4.13)$$

We find the optimal that minimizes the (4.11) equation in every iteration step,

$$\min_{U_{new}^{(n)}} \left\|\boldsymbol{\mathcal{X}} - \left[\!\left[(U^{(1)} + \Delta U^{(1)}), (U^{(2)} + \Delta U^{(2)}), (U^{(3)} + \Delta U^{(3)})\right]\!\right]\right\|. \qquad (4.14)$$

We find that the loading feature factor matrix helps the computation evolve within a given loop by analyzing this algorithm. The convergence within the same loop requires multiple iterations. The following loops exhibit the same scenarios in the direction of the final solution for the optimization.

Algorithm 4.2: Traversal-tensor Method

Input: an original tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \mathrm{I}_3}$, the regularization parameter λ.

Output: the approximate tensor $\bar{\boldsymbol{\mathcal{X}}}$, the factor matrices are the index of users, services, and time, respectively.

Step 1. Initialize regular factor matrices $U^{(2)}, U^{(3)}$ and slices $\mathrm{X}_{(1)}, \mathrm{X}_{(2)}, \mathrm{X}_{(3)}$..

Step 2a. Fixing the $U^{(2)}$ and $U^{(3)}$ to estimate the factor matrices $U^{(1)}$, $\overline{U^{(1)}}$.

Step 2b. Compute the corresponding error ε_{new} and ε.

Step 2c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(1)}$.

Step 3a. Fixing the $U_{new}^{(1)}$ and $U^{(3)}$ to estimate the factor matrices $U^{(2)}$, $\overline{U^{(2)}}$.

Step 3b. Compute the corresponding error ε_{new} and ε.

Step 3c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(2)}$.

Step 4a. Fixing the $U_{new}^{(1)}$ and $U_{new}^{(2)}$ to estimate the factor matrices $U^{(3)}$, $\overline{U^{(3)}}$.

Step 4b. Compute the corresponding error ε_{new} and ε.

Step 4c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(3)}$.

Step 5. Repeat step 2a to step 4c, calculate the approximate tensor $\bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]$.

Step 6. Reduce the loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|^2$ until convergence is exhausted.

Step 7. Return the final prediction tensor $\bar{\boldsymbol{\mathcal{X}}}$.

Comparing with RTD, the TTM also fixes two factor matrices to find another factor matrix. However, TTM has a parallel computational process in each factor matrix computation:

- In step 2a, one fixes $U^{(2)}$ and $U^{(3)}$ to find an enhanced factor matrix $\overline{U^{(1)}}$ as in equation (4.5), and other is to obtain a regular factor matrix $U^{(1)}$ as in equation (4.4).
- In step 2b, the two processing generate two approximate tensors $\bar{\boldsymbol{\mathcal{X}}}$ and $\bar{\boldsymbol{\mathcal{X}}}'$ separately. Based on the two tensors, TTM estimates the two slices $\overline{\mathrm{X}_{(1)}}$ and $\overline{\mathrm{X}_{(1)}}'$, then computes the errors between these two slices and the corresponding slice $\mathrm{X}_{(1)}$ of the original tensor $\boldsymbol{\mathcal{X}}$, respectively.
- By comparing the two errors in step 2c, TTM chooses the factor matrix corresponding to the smaller error as a new factor matrix $U_{new}^{(1)}$ for the next step.
- As same as steps 2a to 2c, TTM generates the new factor matrices $U_{new}^{(2)}$ and $U_{new}^{(3)}$ separately in step 3 and step 4.
- Repeating the above steps, the approximate tensor $\bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]$ is obtained until convergence is exhausted.

4.4 Comparisons of TTM and RTD

In the 4.3 section, we focused on the solution for insufficient sample data, proposed a feasible algorithm, and summarized the steps of the TTM. We plotted two different algorithms for the feature-oriented and regular tensor decomposition in Figure 4.4 and Figure 4.5. Through the two figures, we point out two main differences between TTM and RTD.

- TTM uses more factor matrices to compute enhanced factor matrices during the iterative process.

The enhanced factor matrices are associated with more factor matrices, traversing all the features with the expectation that these features will affect the computation results. Later experiments demonstrate that the computation of the feature factor matrices increases the step size of the iterations.

- TTM adds a comparison error component over RTD, which determines the next step inside the iteration.

If the enhanced factor matrices lead the approximate tensor to move far away from the original tensor, TTM replaces the enhanced factor matrices instead of the regular factor matrices after error comparison. The next iterative process is prevented from moving in the wrong direction. Such a mechanism ensures that the TTM always has an optimal value.

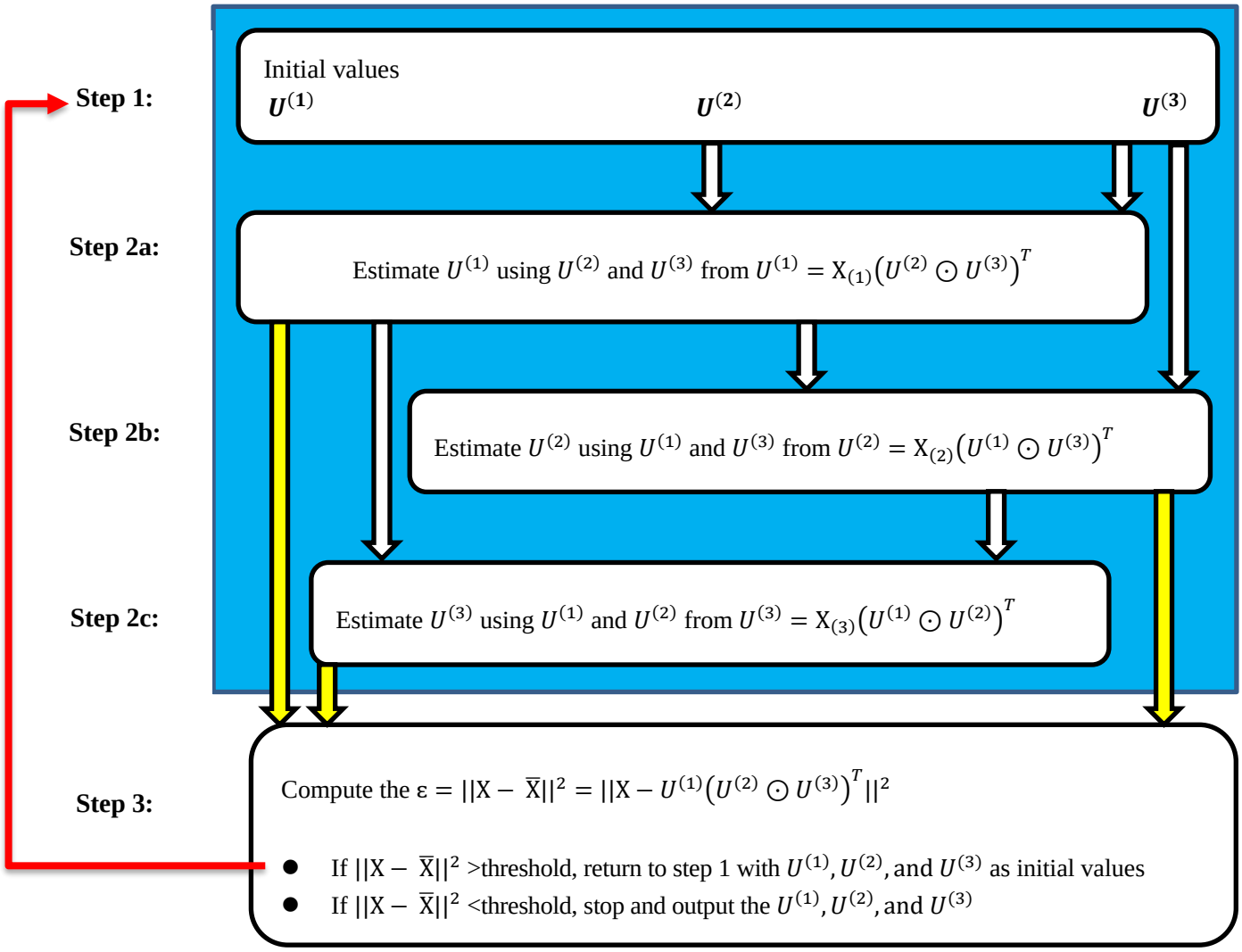


Figure 4.4 RTD algorithm

According to the algorithm shown in Figure 4.4, each iteration step of RTD are performed as the following:

Step 1. Set initial values $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$

Step 2a. Fixing the $U^{(2)}$ and $U^{(3)}$, compute the regular factor matrix $U^{(1)}$ as in equation (4.4).

Step 2b. Fixing the $U^{(1)}$ and $U^{(3)}$, compute the regular factor matrix $U^{(2)}$.

Step 2c. Fixing the $U^{(1)}$ and $U^{(2)}$, compute the regular factor matrix $U^{(3)}$.

Step 3. Perform steps 2a to 2c. Use $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ to estimate each slice of the approximate tensor $\bar{\boldsymbol{\mathcal{X}}}$ as in equation (4.2). Compare the norm of difference slices corresponding to each of the two tensors $\boldsymbol{\mathcal{X}}$ and $\bar{\boldsymbol{\mathcal{X}}}$.

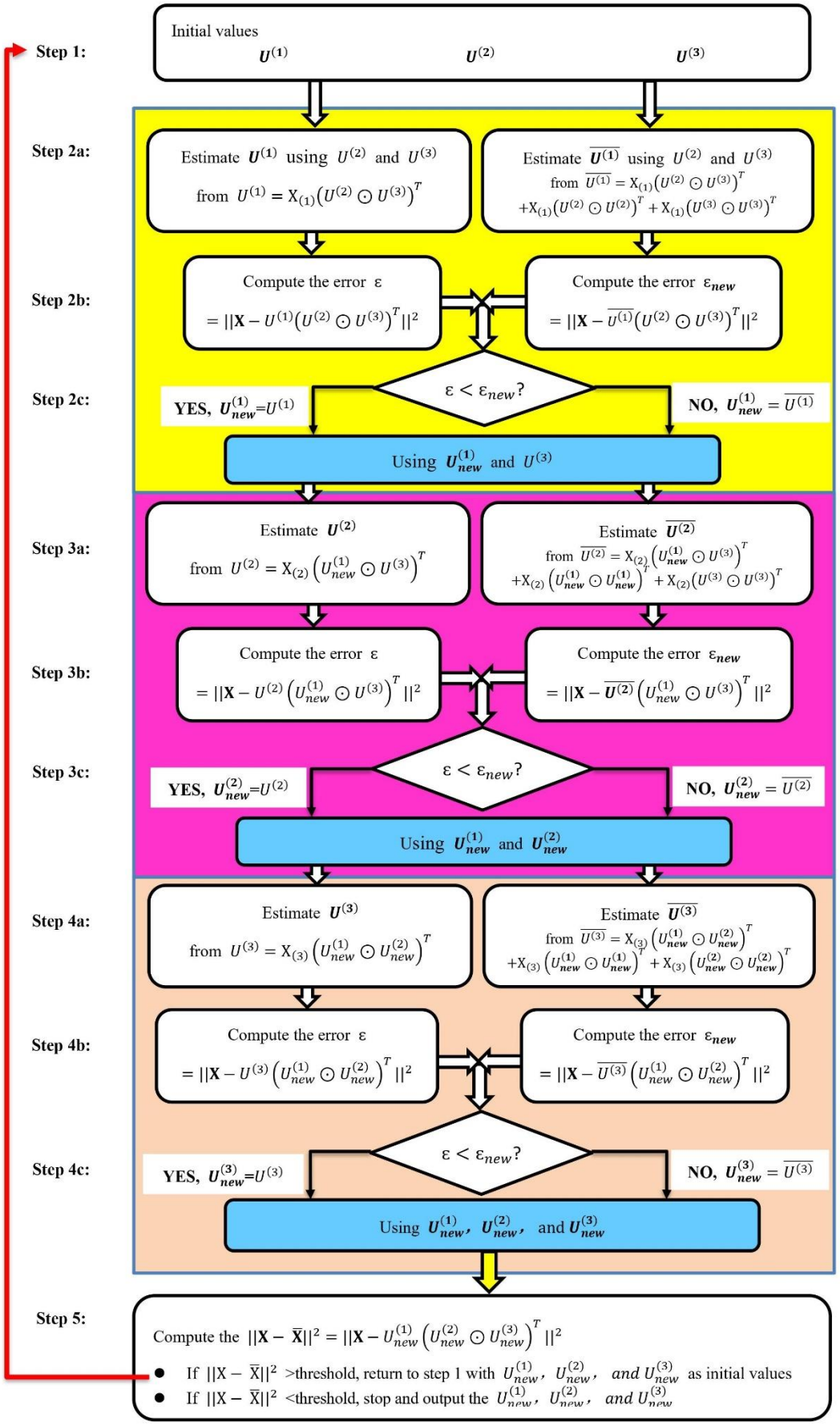

Step 1:
Initial values
$U^{(1)}$ $U^{(2)}$ $U^{(3)}$
Step 2a:
Estimate $U^{(1)}$ using $U^{(2)}$ and $U^{(3)}$ from $U^{(1)} = \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T$
Estimate $\overline{U^{(1)}}$ using $U^{(2)}$ and $U^{(3)}$ from $\overline{U^{(1)}} = \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T + \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(2)}\right)^T + \mathrm{X}_{(1)}\left(U^{(3)} \odot U^{(3)}\right)^T$
Step 2b:
Compute the error $\varepsilon = ||\mathbf{X} - U^{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T||^2$
Compute the error $\varepsilon_{new} = ||\mathbf{X} - \overline{U^{(1)}}\left(U^{(2)} \odot U^{(3)}\right)^T||^2$
Step 2c:
$\varepsilon < \varepsilon_{new}$?
YES, $U_{new}^{(1)} = U^{(1)}$
NO, $U_{new}^{(1)} = \overline{U^{(1)}}$
Using $U_{new}^{(1)}$ and $U^{(3)}$
Step 3a:
Estimate $U^{(2)}$ from $U^{(2)} = \mathrm{X}_{(2)}\left(U_{new}^{(1)} \odot U^{(3)}\right)^T$
Estimate $\overline{U^{(2)}}$ from $\overline{U^{(2)}} = \mathrm{X}_{(2)}\left(U_{new}^{(1)} \odot U^{(3)}\right)^T + \mathrm{X}_{(2)}\left(U_{new}^{(1)} \odot U_{new}^{(1)}\right)^T + \mathrm{X}_{(2)}\left(U^{(3)} \odot U^{(3)}\right)^T$
Step 3b:
Compute the error $\varepsilon = ||\mathbf{X} - U^{(2)}\left(U_{new}^{(1)} \odot U^{(3)}\right)^T||^2$
Compute the error $\varepsilon_{new} = ||\mathbf{X} - \overline{U^{(2)}}\left(U_{new}^{(1)} \odot U^{(3)}\right)^T||^2$
Step 3c:
$\varepsilon < \varepsilon_{new}$?
YES, $U_{new}^{(2)} = U^{(2)}$
NO, $U_{new}^{(2)} = \overline{U^{(2)}}$
Using $U_{new}^{(1)}$ and $U_{new}^{(2)}$
Step 4a:
Estimate $U^{(3)}$ from $U^{(3)} = \mathrm{X}_{(3)}\left(U_{new}^{(1)} \odot U_{new}^{(2)}\right)^T$
Estimate $\overline{U^{(3)}}$ from $\overline{U^{(3)}} = \mathrm{X}_{(3)}\left(U_{new}^{(1)} \odot U_{new}^{(2)}\right)^T + \mathrm{X}_{(3)}\left(U_{new}^{(1)} \odot U_{new}^{(1)}\right)^T + \mathrm{X}_{(3)}\left(U_{new}^{(2)} \odot U_{new}^{(2)}\right)^T$
Step 4b:
Compute the error $\varepsilon = ||\mathbf{X} - U^{(3)}\left(U_{new}^{(1)} \odot U_{new}^{(2)}\right)^T||^2$
Compute the error $\varepsilon_{new} = ||\mathbf{X} - \overline{U^{(3)}}\left(U_{new}^{(1)} \odot U_{new}^{(2)}\right)^T||^2$
Step 4c:
$\varepsilon < \varepsilon_{new}$?
YES, $U_{new}^{(3)} = U^{(3)}$
NO, $U_{new}^{(3)} = \overline{U^{(3)}}$
Using $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$
Step 5:
Compute the $||\mathbf{X} - \bar{\mathbf{X}}||^2 = ||\mathbf{X} - U_{new}^{(1)}\left(U_{new}^{(2)} \odot U_{new}^{(3)}\right)^T||^2$
If $||\mathrm{X} - \bar{\mathrm{X}}||^2$ >threshold, return to step 1 with $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$ as initial values
If $||\mathrm{X} - \bar{\mathrm{X}}||^2$ <threshold, stop and output the $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$

Figure 4.5 TTM algorithm

According to the algorithm shown in Figure 4.5, each iteration step of TTM are performed as the following:

Step 1. Set initial values $U^{(2)}$, and $U^{(3)}$

Step 2a. Fixing the $U^{(2)}$ and $U^{(3)}$, compute the regular factor matrix $U^{(1)}$ as in equation (4.4), and compute the enhanced factor matrix $\overline{U^{(1)}}$ as in equation (4.5).

Step 2b. Compute the corresponding error ε_{new} given by equation (4.11) and error ε given by equation (4.12) separately.

Step 2c. Compare the ε_{new} and ε. If the value of ε_{new} is greater than ε, then set the new factor matrix $U_{new}^{(1)}$ to $U^{(1)}$, otherwise set $U_{new}^{(1)}$ to $\overline{U^{(1)}}$.

Step 3a to 3c. Use $U_{new}^{(1)}$ and $U^{(3)}$ to estimate the regular factor matrix $U^{(2)}$ and enhanced factor matrix $\overline{U^{(2)}}$. Then TTM compares the ε_{new} and ε: If the value of ε_{new} is greater than ε, then set the new factor matrix $U_{new}^{(2)}$ to $U^{(2)}$, otherwise set $U_{new}^{(2)}$ to $\overline{U^{(2)}}$.

Step 4a to 4c. Use $U_{new}^{(1)}$ and $U_{new}^{(2)}$ to estimate the regular factor matrix $U^{(3)}$ and enhanced factor matrix $\overline{U^{(3)}}$. Then TTM compares the ε_{new} and ε: If the value of ε_{new} is greater than ε, then set the new factor matrix $U_{new}^{(3)}$ to $U^{(3)}$, otherwise set $U_{new}^{(3)}$ to $\overline{U^{(3)}}$.

Step 5. Perform steps 2 to 4. Use new factor matrices $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$ as starting values for the iteration instead of $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$. Estimate each slice of the approximate tensor $\bar{\boldsymbol{\mathcal{X}}}$ by using the $U_{new}^{(1)}$, $U_{new}^{(2)}$, and $U_{new}^{(3)}$ as in

equation (4.2). Compare the norm of difference slices corresponding to each of the two tensors $\boldsymbol{\mathcal{X}}$ and $\overline{\boldsymbol{\mathcal{X}}}$.

From these two figures, it can be compared that the iteration processing of TTM adds the calculation of the feature factor matrix. The enhanced factor matrix is updated by the feature factor matrix and regular factor matrix for the next step. TTM needs to compare the approximate tensor and the original tensor when an enhanced factor matrix is obtained. This comparison of differences is used to determine the nesting direction to ensure that the algorithm can move in the direction of optimization.

The RTD has neither the computation of this feature factor matrix nor the comparison of differences. Therefore RTD is simpler than the TTM.

Next, we discuss the TTM specifically in the following aspects: decomposition result, properties, computational complexity, and validation of its results. All these issues are further investigated as follows.

4.4.1 Comparison of decomposition result

To easily understand the decomposition of the two methods, we briefly describe the two decomposition results in one iterative step, which can be found by comparison in Table 4.16:

- After one iteration step, the approximation tensor obtained by the TTM is different from those obtained by the RTD.

  The reason is that TTM needs to calculate more factor matrices to complete the decomposition.

- The norm of tensor generated by TTM is larger than that of the RTD.

  It is indicated that the iteration step of TTM is longer. The increase of iteration size has a positive effect on the improvement of the tensor decomposition performance.

Table 4.16 Results comparison of TTM and RTD

	TTM	**RTD**
$\boldsymbol{\mathcal{X}}$	$\begin{matrix} & & 2 & 4 \\ & & 6 & 12 \\ 1 & 2 & & \\ 3 & 6 & & \end{matrix}$	$\begin{matrix} & & 2 & 4 \\ & & 6 & 12 \\ 1 & 2 & & \\ 3 & 6 & & \end{matrix}$
$\boldsymbol{a_1 \circ b_1 \circ c_1}$	$\begin{pmatrix}0.75\\2.25\end{pmatrix} \circ \begin{pmatrix}3.5778\\7.1556\end{pmatrix} \circ \begin{pmatrix}1.7408\\3.4818\end{pmatrix}$	$\begin{pmatrix}-1/2\\-3/2\end{pmatrix} \circ \begin{pmatrix}-2\\-4\end{pmatrix} \circ \begin{pmatrix}1\\2\end{pmatrix}$
$\bar{\boldsymbol{\mathcal{X}}}$	$\begin{matrix} & & 9.3429 & 18.6858 \\ & & 14.0135 & 28.0271 \\ 4.6712 & 9.3424 & & \\ 14.0135 & 28.0271 & & \end{matrix}$	$\begin{matrix} & & 2.0000 & 4.0000 \\ & & 6.0000 & 12.0000 \\ 1.0000 & 2.0000 & & \\ 3.0000 & 6.0000 & & \end{matrix}$
$\boldsymbol{Norm\ of\ \bar{\mathcal{X}}}$	73.8612	15.8114
$\boldsymbol{Norm\ of\ (\bar{\mathcal{X}} - \mathcal{X})}$	58.0498	4.4409×10^{-15}

In table 4.16, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathbf{2\times2\times2}}$ denotes a 3-way tensor with rank-one. $\bar{\boldsymbol{\mathcal{X}}} = \boldsymbol{a_1} \circ \boldsymbol{b_1} \circ \boldsymbol{c_1}$ denotes an approximation tensor. $\boldsymbol{a_1}, \boldsymbol{b_1}, \boldsymbol{c_1}$ are vector after tensor decomposition separately. The decomposition detail is illustrated in Section 4.4.4.

4.4.2 Convergence properties

Examining the TTM as an optimization algorithm from a mathematical perspective, the following properties can be observed.

I. The derivative of a loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ at the minimum is vanished, thus the minimum of the loss function is a fixed point.

The ALS algorithm does not reach a global minimum from any starting point [Shi, Li, & Zhang, 2019]. Since our method is based on the ALS algorithm, TTM can only ensure that a local minimum is reached.

II. The TTM result moves towards the minimum, no matter whether the negative or positive gradient.

An example of gradient descent is shown as follows.

Given a $2 \times 2 \times 2$ 3-way tensor $\mathcal{X} = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$, the vectors $a = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $b = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$, and $c = \begin{pmatrix} \cos\beta \\ \sin\beta \end{pmatrix}$ denote the initial values, which are the trigonometric function value of angle variables α, β respectively [De Lathauwer, De Moor, & Vandewalle, 2000]. α and β are in a range from -π to π.

Suppose the desired output of a TTM is an original tensor $\boldsymbol{\mathcal{X}}$. The TTM predicts an output of approximation tensor $\bar{\boldsymbol{\mathcal{X}}} = a \circ b \circ c$. Difference between the real output and the predicted output $(\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}})$ is converted into the loss function $\mathrm{f(a, b, c)} = \|\boldsymbol{\mathcal{X}} - a \circ b \circ$

$c\|^2$. Our goal is to optimize the loss function to make a loss as minimum as possible with the TTM result.

The loss function above is shaped surface — the partial derivative of the loss function $f(a, b, c)$ with respect to the weight is the slope of the surface at the location. By moving in the direction predicted by TTM, TTM moves towards the bottom of the surface - minimizing the loss function.

It is observed from Figure 4.6 that the minimum of the function gives the best approximations highlighted in deep blue.

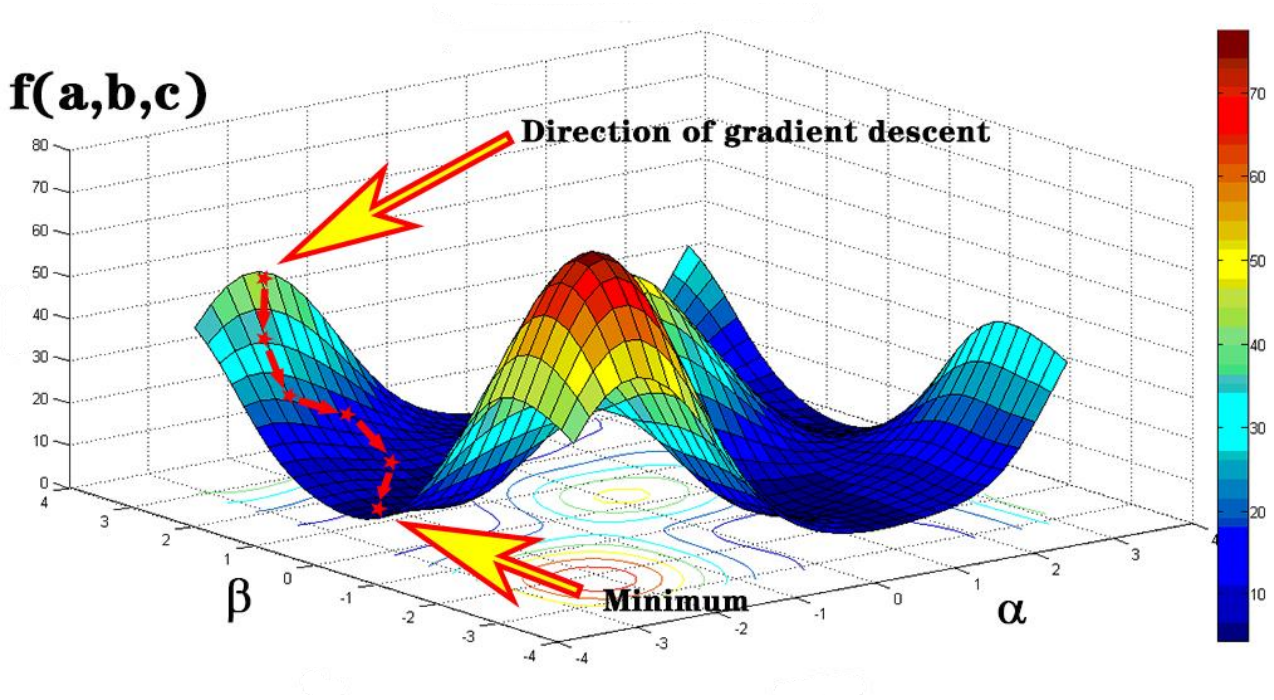


Figure 4.6 An example of gradient descent

III. The TTM reduced the number of iterations for the convergence to get a reasonable result.

The TTM uses more matrices to compute the same factor matrices in the iteration. This fact is noticed, which the large size step would reduce the number of iteration steps. Thus, the process of fitting the minimum value is easier than RTD.

The fact that the TTM has a small number of iteration steps leads to a question: which is better for improving the fit or increasing the computational cost? Question is particularly important in cases where the iteration becomes slow because of more factor matrices. For example, one can imagine that a big-size steps method might trade off some slower steps for the same finish line, when the computer performance has been very high and the cost in time is not much. This question can be answered well by applying a method to simulated or real data.

We give an example of a small number of iteration steps for comparing TTM and RTD. We set a simulated movie rating experiment, in which the movie dataset has three dimensions data: three users, four items, and four contexts. The movie rating is predicted using three-dimensional data in (user, item, context) -> rating. After prediction, the iteration number is recorded in Table 4.17.

Table 4.17 Iteration steps number comparison

Method	**Prediction Error rate**									
	1%	**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**
RTD	8969	1683	1289	1756	2018	1877	1293	1497	1755	1355
TTM	1582	801	1094	605	1062	761	439	335	520	318

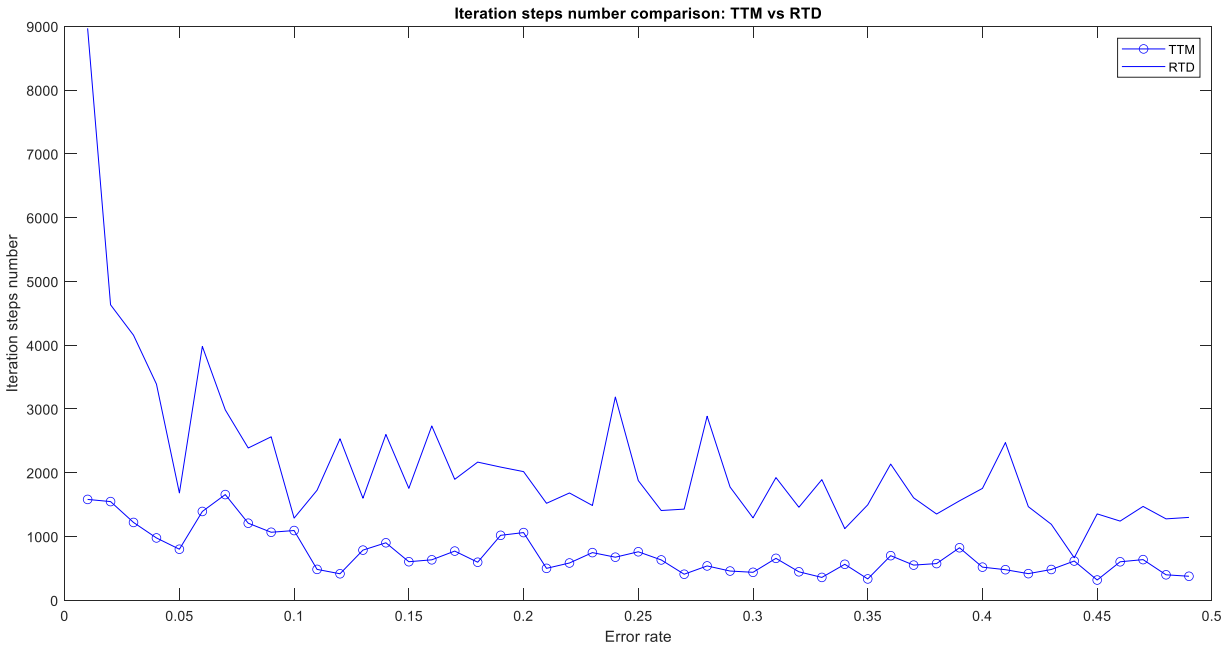


Figure 4.7 Iteration steps number comparison

The horizontal coordinate of Figure 4.6 represents the error rate of the prediction, and the vertical coordinate represents the number of iteration steps. When the prediction error rate becomes larger, the accuracy of the prediction will be lower.

We can find that the RTD curve decreases greatly as the error rate decreases, indicating that the number of iteration steps required for the RTD is gradually decreasing. At the same time, the TTM curve changes more slowly and requires fewer iteration steps than the RTD for the same error rate. For example, for an error rate of 1%, the number of iteration steps required for TTM is 1582, which is far less than 8969 for RTD. In the same way, comparing the number of iteration steps with the same error rate, the corresponding number of iterations for TTM is always smaller than that of RTD. Thus, we conclude that TTM can reach the minimum point faster and is more efficient.

The algorithm's convergence requires many iterations, and the initial value of the missing data will also affect the number of iterations. A good choice of starting values will help to reach the minimum quickly in some cases. Therefore, we discuss the effect of different

missing initial values on convergence and find that choosing a good initial value improves convergence efficiency in the experiment.

4.4.3 Validation of the results.

In this subsection, we analyze the different results between TTM and RTD through the visualization representation.

Given a $2 \times 2 \times 2$ 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and its approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$. The original tensor $\boldsymbol{\mathcal{X}}$ is defined as follows,

$$\boldsymbol{\mathcal{X}} = \begin{matrix} x_{111} & x_{121} \\ x_{211} & x_{221} \end{matrix} \quad \begin{matrix} x_{112} & x_{122} \\ x_{212} & x_{222} \end{matrix} \tag{4.15}$$

where $\boldsymbol{\mathcal{X}}$ denotes the tensor, $x_{i_1 i_2 i_3}$ denotes the tensor elements, and $all\ i_1, i_2, i_3 = 1,2$ denote each entry respectively of the tensor's elements.

Setting three vectors $a, b, c \in \mathbb{R}^2$, and $a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$, $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ where $a_1, a_2, b_1, b_2, c_1, c_2$ are the elements of the corresponding vectors, the definition of vector outer product is as follows,

$$\begin{aligned} \bar{\boldsymbol{\mathcal{X}}} &= a \circ b \circ c = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \circ \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \circ \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \\ &= \begin{matrix} a_1 b_1 c_1 & a_1 b_2 c_1 \\ a_2 b_1 c_1 & a_2 b_2 c_1 \end{matrix} \quad \begin{matrix} a_1 b_1 c_2 & a_1 b_2 c_2 \\ a_2 b_1 c_2 & a_2 b_2 c_2 \end{matrix}. \end{aligned} \tag{4.16}$$

To make a visual representation, we set the initial value for the vectors as follows (the same representation is used in De Lathauwer, etc. related literature [De Lathauwer, De Moor, & Vandewalle, 2000]),

$$b_1 = \cos\alpha\,, b_2 = \sin\alpha\,, c_1 = \cos\beta\,, c_2 = \sin\beta$$

where two angle variables α and β are in a range from -π to π.

Substituting the above initial value of vectors $\boldsymbol{b}$ *and* $\boldsymbol{c}$ into the equation (4.16), the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ is rewrite as follows,

$$\bar{\boldsymbol{\mathcal{X}}} = a \circ b \circ c$$

$$= \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \circ \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \circ \begin{pmatrix} \cos\beta \\ \sin\beta \end{pmatrix}$$

$$= \begin{matrix} a_1\cos\alpha\cos\beta & a_1\sin\alpha\cos\beta \\ a_2\cos\alpha\cos\beta & a_2\sin\alpha\cos\beta \end{matrix} \begin{matrix} a_1\cos\alpha\sin\beta & a_1\sin\alpha\sin\beta \\ a_2\cos\alpha\sin\beta & a_2\sin\alpha\sin\beta \end{matrix}. \quad (4.17)$$

where two variables α and β are in a range from -π to π.

Thus, the loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ of the original tensor $\boldsymbol{\mathcal{X}}$ and its approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ is obtained by substituting the equation (4.15) and (4.17),

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\|$$

$$= \left\| \begin{matrix} x_{111} & x_{121} \\ x_{211} & x_{221} \end{matrix} \begin{matrix} x_{112} & x_{122} \\ x_{212} & x_{222} \end{matrix} \right.$$

$$\left. - \begin{matrix} a_1\cos\alpha\cos\beta & a_1\sin\alpha\cos\beta \\ a_2\cos\alpha\cos\beta & a_2\sin\alpha\cos\beta \end{matrix} \begin{matrix} a_1\cos\alpha\sin\beta & a_1\sin\alpha\sin\beta \\ a_2\cos\alpha\sin\beta & a_2\sin\alpha\sin\beta \end{matrix} \right\|$$

$$= (x_{111} - a_1\cos\alpha\cos\beta)^2 + (x_{121} - a_1\sin\alpha\cos\beta)^2 + (x_{211} - a_2\cos\alpha\cos\beta)^2$$

$$+(x_{221} - a_2\sin\alpha\cos\beta)^2 + (x_{112} - a_1\cos\alpha\sin\beta)^2 + (x_{122} - a_1\sin\alpha\sin\beta)^2$$

$$+(x_{212} - a_2\cos\alpha\sin\beta)^2 + (x_{222} - a_2\sin\alpha\sin\beta)^2. \quad (4.18)$$

where the vectors b and c can be normalized by using the trigonometric function of α and β. The vector a is a set of different initial values as $a = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $a = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, and $a = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$. The $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ can be generated by two variables α and β when fixing the vector a.

By adjusting the equation (4.20) to satisfy the TTM and RTD, we obtain visualization results for both TTM and RTD methods by the curves of $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ in Table 4.18 to 4.21.

Table 4.18 First approximation tensor with rank 3

$$\mathcal{X} = \begin{matrix} & & 0 & 1 \\ & & -1 & 0 \\ 1 & 0 & & \\ 0 & 1 & & \end{matrix}$$

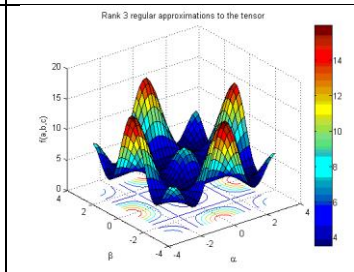
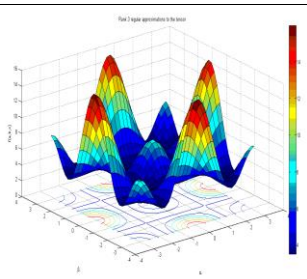
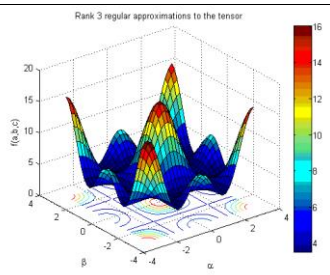
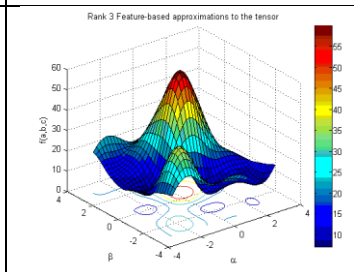
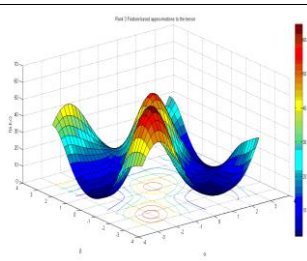
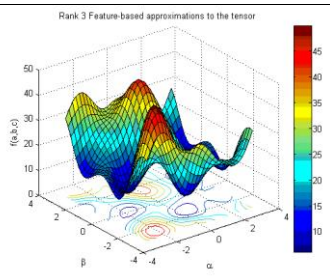
	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{-1} \\ \mathbf{1} \end{pmatrix}$
RTD			
TTM			

Table 4.19 Second approximation tensor with rank 3

$$\mathcal{X} = \begin{matrix} 5 & 6 \\ 7 & 8 \end{matrix}$$
$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

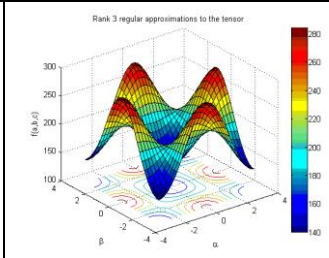
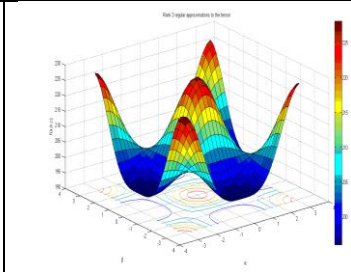
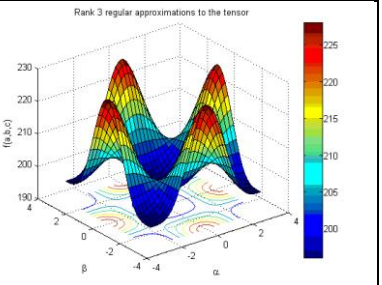
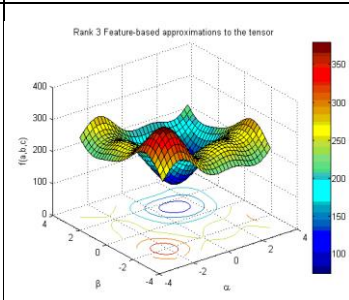
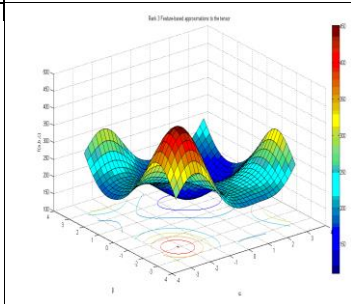
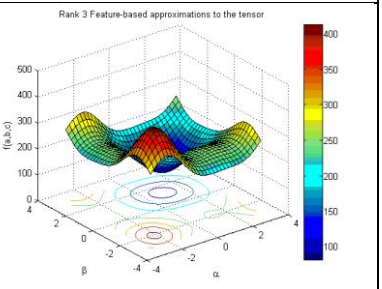
	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{-1} \\ \mathbf{1} \end{pmatrix}$
RTD			
TTM			

Table 4.20 Third approximation tensor with rank 2

$$\mathcal{X} = \begin{matrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{matrix} \quad \begin{matrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{matrix}$$

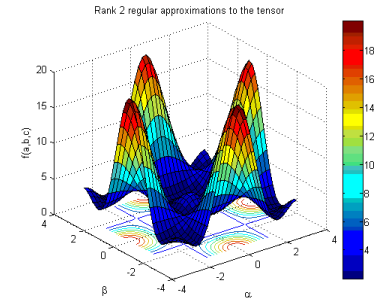
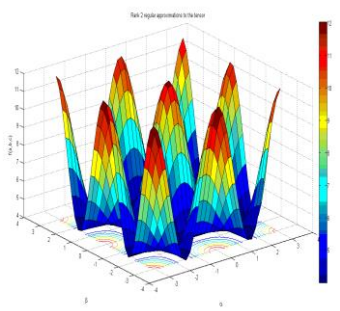
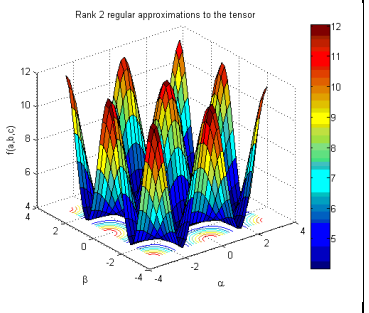
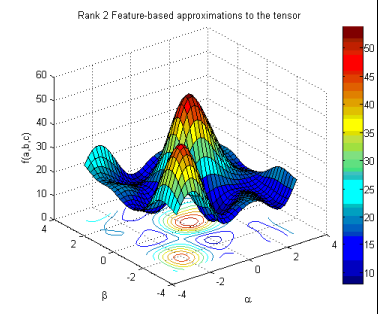
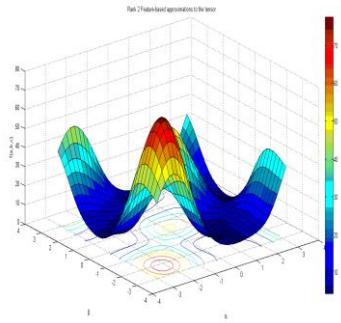
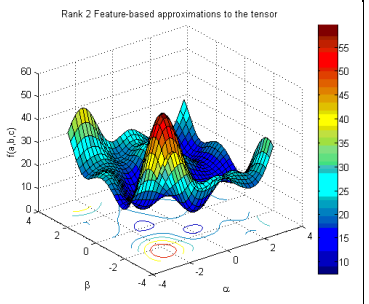
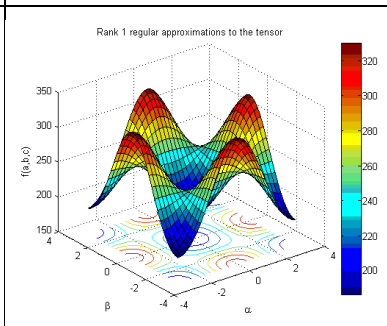
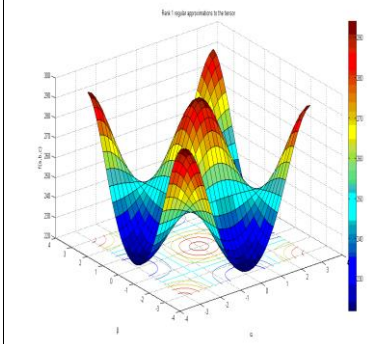
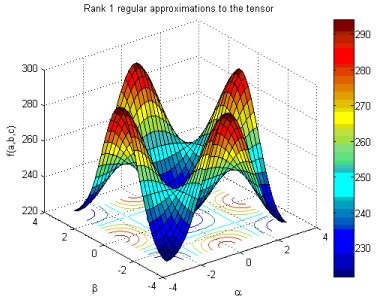
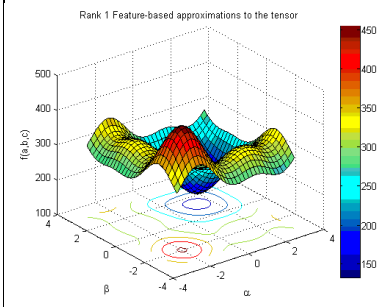
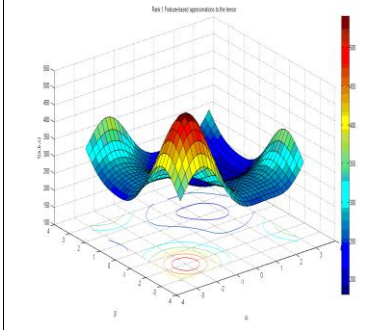
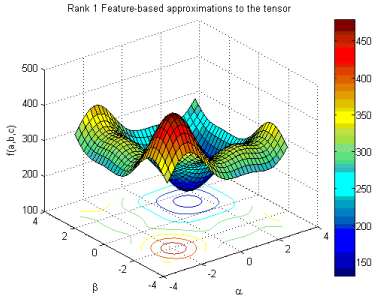
	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{-1} \\ \mathbf{1} \end{pmatrix}$
RTD			
TTM			

Table 4.21 Fourth approximation tensor with rank 1

$$\mathcal{X} = \begin{matrix} & \begin{matrix} 2 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 3 \end{matrix} & \begin{matrix} 2 \\ 6 \end{matrix} \end{matrix} \quad \begin{matrix} \\ \begin{matrix} 6 & 12 \end{matrix} \end{matrix}$$

	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$	$\boldsymbol{a} = \begin{pmatrix} \mathbf{-1} \\ \mathbf{1} \end{pmatrix}$
RTD			
TTM			

The coordinate system in which the curves are located is represented as that the horizontal plane is composed of two axes, representing the two variables α and β, and the vertical coordinate represents the value of the $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ lost function. The color of the curve changes progressively from warm to cool color, with cooler colors indicating lower values of the $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ lost function. The deep blue color indicates the lowest value, and the deep red color indicates the highest value.

Observing that these curves, we conclude that the proposed decomposition method is convergent.

(1) The lowest value obtained of lost function by the TTM remains a fixed positive value.

The concave point of the curve indicates the location of the lowest value of the function. All the curve of TTM has a concave point. The vertical coordinates of the positions of these points are all above the zero value.

(2) The process of finding the lowest value in TTM is easier than that of RTD.

If the curve has more concave points, indicating that it has a more minimal value, it may require more complicated processing steps. Since the number of convergence points obtained by the RTD is smaller than that of RTD, as shown in Table 4.22, the process of finding the lowest in RTD is more complex and less easy than that in TTM.

Table 4.22 Number of convergence points

Tensor Rank	**RTD**	**TTM**	**Input vector**
Rank = 1	3	1	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$
Rank = 2	>2	2	
Rank = 3 (First approximation tensor)	3	1	
Rank = 3 (Second approximation tensor)	>2	2	

Meanwhile, the curve obtained by TTM is smoother than that of RTD. Especially after decomposing, it is evident that the curve of RTD is steeper than the curve of TTM in Table 4.21. This means that the iteration size in the TTM is larger than the RTD iteration size, which is consistent with the conclusion we discussed in section 4.4.1.

(3) The lowest value obtained of lost function by the TTM is lower than or equal to that of RTD.

The lowest value indicates the difference between the approximate tensor and the original tensor.

The lower this value is, the lower the difference is. The result shown in Table 4.23 shows that the approximate tensor obtained by TTM is closer to the original tensor.

Table 4.23 Lowest value of lost function

Tensor Rank	RTD	TTM	Input vector
Rank = 1	220	100	
Rank = 2	4	0	$\boldsymbol{a} = \begin{pmatrix} \mathbf{1} \\ \mathbf{-1} \end{pmatrix}$
Rank = 3 (First approximation tensor)	190	100	
Rank = 3 (Second approximation tensor)	0	0	

4.4.4 Computational complexity

TTM has a higher computational complexity. In the following, we further compare the computational complexity of the two methods.

(1) Complexity computation

Let a 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, where I_1, I_2, and I_3 is the dimensionality of first, second, and third-dimensional data, respectively. R is the tensor's rank. $U^{(1)}, U^{(2)}$, and $U^{(3)}$ are the factor matrices.

According to the analysis of computational complexity in [Zhang et al., 2014b], for each iteration of RTD, the primary cost of the computation complexity is step 2a -2c of the RTD algorithm, each of which is $O(I_1I_2I_3R + (2I_1 + I_2 + I_3)R^2 + I_1I_2I_3R)$, and step3 of the RTD algorithm is $O(I_1I_2I_3R)$. The computation complexity can be considered in one iteration as follows,

$$O(I_1I_2I_3R + (2I_1 + I_2 + I_3)R^2 + I_1I_2I_3R) + O(I_1I_2I_3R). \quad (4.19)$$

In contrast, the prediction step of TTM includes extra operations, which means the TTM may take more time than expected. Such extra computation time depends on the implementation, the CPU performance, the system platform, and other factors. Therefore, its impact can not be computed exactly. The summarization of the TTM computational complexity is shown as follows,

The steps of the ELS and their computational complexities are listed as follows:

- Calculate the increments $U_{new}^{(n)} = U^{(n)} + \Delta U^{(n)}\ n = 1,2, and\ 3$, which needs no multiplications.
- Calculate the feature factor matrices of the following (given $\Delta U^{(1)}$ as an example),

  $$\Delta U^{(1)} = X_{(1)}\left(U^{(2)} \odot U^{(2)}\right)^T + X_{(1)}\left(U^{(3)} \odot U^{(3)}\right)^T, n = 1,2, and\ 3.$$

  The cost of the computation complexity is,

  $$O\left((10I_1 + 9I_2 + 9I_3)R^2 + \left(I_1I_2I_3 + I_1^2(I_2 + I_3) + I_2^2(I_1 + I_3) + I_3^2(I_1 + I_2)\right)R\right)$$
  $$+O(2(I_2I_3 + I_1I_2 + I_1I_3 + I_2I_3)R). \quad (4.20)$$

- Calculate the approximate tensor $\bar{\boldsymbol{\mathcal{X}}} = \left[\!\left[U_{\boldsymbol{new}}^{(\mathbf{1})}, U_{\boldsymbol{new}}^{(\mathbf{2})}, U_{\boldsymbol{new}}^{(\mathbf{3})}\right]\!\right]$, and the cost of the computation complexity is,

  $$O(I_1I_2I_3R). \quad (4.21)$$

Therefore, the computation complexity of the TTM prediction step is as follows, which is higher than that of RTD,

$$O\big(I_1I_2I_3R + (10I_1 + 9I_2 + 9I_3)R^2 + \big(I_1I_2I_3 + I_1^2(I_2 + I_3) + I_2^2(I_1 + I_3) + I_3^2(I_1 + I_2)\big)R\big) + O(2(I_2I_3 + I_1I_2 + I_1I_3 + I_2I_3)R). \quad (4.22)$$

(2) Numerical examples

Considering that the tensor decomposition process uses rather cumbersome mathematical notation, the numerical examples are visualizations to facilitate understanding. Thus, we give numerical examples to compare the regular tensor decomposition model RTD and the proposed decomposition method TTM. The original tensor example is selected from the classical example in Kolda's literature [Kolda, 2006]. It is noted that we only consider the tensor with rank one. Because computing the rank of a three-dimensional tensor over any finite field is NP-complete, it cannot be determined randomly[Hillar & Lim, 2013].

The complexity can also be verified through the following examples that TTM has higher complexity since it requires more steps to calculate the enhanced factor matrices.

Given a 3-way tensor with rank-one $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\mathbf{2}\times\mathbf{2}\times\mathbf{2}}$,

$$\boldsymbol{\mathcal{X}} = \begin{matrix} & & 2 & 4 \\ & & 6 & 12 \\ 1 & 2 & & \\ 3 & 6 & & \end{matrix}$$

by when unfolding tensor, the frontal slices $\mathrm{X}_{(n)}, n = 1,2, and\ 3$ are obtained as follows,

$$\mathrm{X}_{(1)} = \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}$$

$$\mathrm{X}_{(2)} = \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}$$

$$\mathrm{X}_{(3)} = \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}.$$

The initial setup is fixing the vectors $b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Next, we calculate the new vectors $a_1, b_1, and\ c_1$.

According to the properties of the tensor decomposition with rank-one, the factor matrices $U^{(1)}, U^{(2)}, and\ U^{(3)}$ are updated sequentially by these vectors $a_n, b_n, and\ c_n\ n = 0,1$ in iteration steps. Finally, the approximation tensor with rank-one $\bar{\boldsymbol{\mathcal{X}}}$ is calculated by the following equation,

$$\bar{\boldsymbol{\mathcal{X}}} = [\![U^{(1)}, U^{(2)}, U^{(3)}]\!] = a_1 \circ b_1 \circ c_1.$$

Based on the initial factor matrices: $U^{(2)} = b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we are computing other factor matrices $U^{(1)} = a_1$, and updating the factor matrices $U^{(2)}, U^{(3)}$ by $U^{(2)} = b_1$ and $U^{(3)} = c_1$ separately.

A. RTD

The initial setup is fixing the initial factor matrices $U^{(2)} = b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ to compute the factor matrix $U^{(1)} = a_1$,

$$U^{(1)} = \mathrm{X}_{(1)}\left[U^{(3)} \odot U^{(2)}\right]\left[\left(U^{(3)}\right)^T U^{(3)} * \left(U^{(2)}\right)^T U^{(2)}\right]^+$$

$$\Longrightarrow a_1 = \mathrm{X}_{(1)}[c_0 \odot b_0][(c_0)^T c_0 * (b_0)^T b_0]^+$$

$$= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}[(1) * (2)]^+ = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}.$$

Then fixing the factor matrices $U^{(1)} = a_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} and\ U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ to update the factor matrix $U^{(2)} = b_1$,

$$U^{(2)} = \mathrm{X}_{(2)}\left[U^{(3)} \odot U^{(1)}\right]\left[\left(U^{(3)}\right)^T U^{(3)} * \left(U^{(1)}\right)^T U^{(1)}\right]^+$$

$$\Longrightarrow b_1 = \mathbf{X}_{(2)}[c_0 \odot a_1][(c_0)^T c_0 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}^T \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} -1/2 \\ -3/2 \\ 0 \\ 0 \end{bmatrix}[(1) * (5/2)]^+ = \begin{pmatrix} -2 \\ -4 \end{pmatrix}.$$

Then fixing the factor matrices $U^{(2)} = b_1 = \begin{pmatrix} -2 \\ -4 \end{pmatrix} and\ U^{(1)} = a_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}$ to update the factor matrix $U^{(3)} = c_1$,

$$U^{(3)} = \mathrm{X}_{(3)}\left[U^{(2)} \odot U^{(1)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(1)}\right)^T U^{(1)}\right]^+$$

$$\Rightarrow c_1 = \mathrm{X}_{(3)}[b_1 \odot a_1][(b_1)^T b_1 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \left[\begin{pmatrix} -2 \\ -4 \end{pmatrix} \odot \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} \right] \left[\begin{pmatrix} -2 \\ -4 \end{pmatrix}^T \begin{pmatrix} -2 \\ -4 \end{pmatrix} * \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}^T \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} \right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 6 \end{bmatrix} [(20) * (5/2)]^+ = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Finally, the factor matrices $U^{(1)} = a_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}, U^{(2)} = b_1 = \begin{pmatrix} -2 \\ -4 \end{pmatrix}$, and $U^{(3)} = c_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ are obtained. According to the rule of tensor decomposition, these vectors are used as factor matrices to calculate the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ as follows,

$$\bar{\boldsymbol{\mathcal{X}}} = a_1 \circ b_1 \circ c_1 = \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} \circ \begin{pmatrix} -2 \\ -4 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{matrix} 1.0000 & 2.0000 \\ 3.0000 & 6.0000 \end{matrix} \quad \begin{matrix} 2.0000 & 4.0000 \\ 6.0000 & 12.0000 \end{matrix}.$$

B. TTM

TTM compute the regular factor matrix $U^{(n)}$ and feature factor matrix $\Delta U^{(\mathrm{n})}$. And the enhanced factor matrix $\overline{U^{(n)}}\ n = 0{,}1$ is obtained by the $\overline{U^{(n)}} = U^{(n)} + \Delta U^{(\mathrm{n})}$. Finally, the approximation tensor with rank-one $\bar{\boldsymbol{\mathcal{X}}}$ is calculated by the following equation,

$$\bar{\boldsymbol{\mathcal{X}}} = [\![\overline{U^{(1)}}, \overline{U^{(2)}}, \overline{U^{(3)}}]\!] = a_1 \circ b_1 \circ c_1.$$

Based on the initial factor matrices: $U^{(2)} = b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we are computing another enhanced factor matrix $\overline{U^{(1)}}$, and generating the enhanced factor matrices $\overline{U^{(2)}} = b_1$ and $\overline{U^{(3)}} = c_1$ separately.

(a) Enhanced factor matrix $\overline{U^{(1)}} = U^{(1)} + \Delta U^{(1)}$

First, as same as the factor matrix computation of regular tensor decomposition, when fixing the initial factor matrices $U^{(2)} = b_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} and\ U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we compute the regular factor matrix $U^{(1)}$ as follows,

$$
\begin{aligned}
U^{(1)} &= X_{(1)}\left[U^{(3)} \odot U^{(2)}\right]\left[\left(U^{(3)}\right)^T U^{(3)} * \left(U^{(2)}\right)^T U^{(2)}\right]^+ \\
&= \mathrm{X}_{(1)}[c_0 \odot b_0][(c_0)^T c_0 * (b_0)^T b_0]^+ \\
&= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]^+ \\
&= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}[(1) * (2)]^+ \\
&= \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix}.
\end{aligned}
$$

Second, we compute the feature factor matrix $\Delta U^{(1)}$ as follows,

$$
\begin{aligned}
\Delta U^{(1)} &= \mathrm{X}_{(1)}\left(U^{(2)} \odot U^{(2)}\right)^T + \mathrm{X}_{(1)}\left(U^{(3)} \odot U^{(3)}\right)^T \\
&= \mathbf{X}_{(1)}[b_0 \odot b_0][(b_0)^T b_0 * (b_0)^T b_0]^+ + \mathrm{X}_{(1)}[c_0 \odot c_0][(c_0)^T c_0 * (c_0)^T c_0]^+ \\
&= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ -1 \end{pmatrix} \odot \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right]^+ + \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]^+ \\
&= \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}[(2) * (2)]^+ + \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 6 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}[(1) * (1)]^+ \\
&= \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \end{pmatrix}.
\end{aligned}
$$

Third, the enhanced factor matrix $\overline{U^{(1)}} = U^{(1)} + \Delta U^{(1)}$ is computed based on the above results as follows:

$$
\begin{aligned}
\overline{U^{(1)}} &= U^{(1)} + \Delta U^{(1)} \\
&= \mathbf{X}_{(1)}[c_0 \odot b_0][(c_0)^T c_0 * (b_0)^T b_0]^+ + \mathbf{X}_{(1)}[b_0 \odot b_0][(b_0)^T b_0 * (b_0)^T b_0]^+ \\
&\quad + \mathbf{X}_{(1)}[c_0 \odot c_0][(c_0)^T c_0 * (c_0)^T c_0]^+ \\
&= \begin{pmatrix} -1/2 \\ -3/2 \end{pmatrix} + \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} = \begin{pmatrix} 0.75 \\ 2.25 \end{pmatrix}.
\end{aligned}
$$

Finally, we obtain the enhanced factor $\overline{U^{(1)}} = \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}$. This enhanced factor is used as the initial factor matrix to generate a second enhanced factor matrix $\overline{U^{(2)}}$ in the next step.

(b) Enhanced factor matrix $\overline{U^{(2)}} = U^{(2)} + \Delta U^{(2)}$

When fixing the factor matrices $\overline{U^{(1)}} = a_1 = \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}$ and $U^{(3)} = c_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we are computing the enhanced factor matrix $\overline{U^{(2)}}$.

First, we compute the regular factor $U^{(2)}$ as follows,

$$U^{(2)} = X_{(2)}\left[U^{(3)} \odot \overline{U^{(1)}}\right]\left[\left(U^{(3)}\right)^T U^{(3)} * \left(\overline{U^{(1)}}\right)^T \overline{U^{(1)}}\right]^+$$

$$= \mathrm{X}_{(2)}[c_0 \odot a_1][(c_0)^T c_0 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}^T \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} 3/4 \\ 9/4 \\ 0 \\ 0 \end{bmatrix}[(1) * (45/8)]^+ = \begin{bmatrix} 15/2 \\ 15 \end{bmatrix}[(1) * (45/8)]^+ = \begin{pmatrix} 4/3 \\ 8/3 \end{pmatrix}.$$

Second, we compute the feature factor matrix $\Delta U^{(2)}$ as follows,

$$\Delta U^{(2)} = \mathrm{X}_{(2)}\left(\overline{U^{(1)}} \odot \overline{U^{(1)}}\right)^T + \mathrm{X}_{(2)}\left(U^{(3)} \odot U^{(3)}\right)^T$$

$$= \mathrm{X}_{(2)}[a_1 \odot a_1][(a_1)^T a_1 * (a_1)^T a_1]^+ + \mathrm{X}_{(2)}[c_0 \odot c_0][(c_0)^T c_0 * (c_0)^T c_0]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} \odot \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}\right]\left[\begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}^T \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} * \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}^T \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}\right]^+ + \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} 9/16 \\ 27/16 \\ 27/16 \\ 81/16 \end{bmatrix}[(45/8) * (45/8)]^+ + \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}[(1) * (1)]^+$$

$$= \begin{pmatrix} 56/45 \\ 112/45 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Third, the enhanced factor matrix $\overline{U^{(2)}} = U^{(2)} + \Delta U^{(2)}$ is computed based on the above results as follows:

$$\overline{U^{(2)}} = U^{(2)} + \Delta U^{(2)}$$

$$= \mathbf{X}_{(2)}[c_0 \odot a_1][(c_0)^T c_0 * (a_1)^T a_1]^+ + \mathbf{X}_{(2)}[c_0 \odot c_0][(c_0)^T c_0 * (c_0)^T c_0]^+ + \mathbf{X}_{(2)}[a_1 \odot a_1][(a_1)^T a_1 * (a_1)^T a_1]^+$$

$$= \begin{pmatrix} 4 \\ \frac{3}{8} \\ \frac{3}{3} \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} \frac{56}{45} \\ \frac{112}{45} \end{pmatrix}$$

$$= \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix} = \begin{pmatrix} 3.5778 \\ 7.1556 \end{pmatrix}.$$

Finally, we obtain the enhanced factor $\overline{U^{(2)}} = \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix}$. This enhanced factor is used as the initial factor matrix to generate the enhanced factor matrix $\overline{U^{(3)}}$ in the next step.

(c) Enhanced factor matrix $\overline{U^{(3)}} = U^{(3)} + \Delta U^{(3)}$

When fixing the factor matrices $\overline{U^{(1)}} = a_1 = \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}$ *and* $\overline{U^{(2)}} = b_1 = \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix}$, we are computing the factor matrix $\overline{U^{(3)}}$.

First, we compute the regular factor $U^{(3)}$ as follows,

$$U^{(3)} = X_{(3)}\left[\overline{U^{(1)}} \odot \overline{U^{(2)}}\right]\left[\left(\overline{U^{(1)}}\right)^T \overline{U^{(1)}} * \left(\overline{U^{(2)}}\right)^T \overline{U^{(2)}}\right]^+$$

$$= \mathrm{X}_{(3)}[b_1 \odot a_1][(b_1)^T b_1 * (a_1)^T a_1]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\left[\begin{pmatrix} \frac{161}{45} \\ \frac{322}{45} \end{pmatrix} \odot \begin{pmatrix} \frac{3}{4} \\ \frac{9}{4} \end{pmatrix}\right]\left[\begin{pmatrix} \frac{161}{45} \\ \frac{322}{45} \end{pmatrix}^T \begin{pmatrix} \frac{161}{45} \\ \frac{322}{45} \end{pmatrix} * \begin{pmatrix} \frac{3}{4} \\ \frac{9}{4} \end{pmatrix}^T \begin{pmatrix} \frac{3}{4} \\ \frac{9}{4} \end{pmatrix}\right]^+$$

$$= \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix}\begin{bmatrix} 161/60 \\ 483/60 \\ 322/60 \\ 966/60 \end{bmatrix}\left[\left(\frac{161^2 + 322^2}{45^2}\right) * \left(\frac{45}{8}\right)\right]^+$$

$$= \begin{bmatrix} \frac{161}{60} + \frac{1449}{60} + \frac{966}{60} + \frac{5796}{60} \\ \frac{322}{60} + \frac{2898}{60} + \frac{1932}{60} + \frac{11592}{60} \end{bmatrix}\left[\left(\frac{161^2}{72}\right)\right]^+$$

$$= \begin{bmatrix} \frac{8372}{60} \\ \frac{16744}{60} \end{bmatrix} \left(\frac{72}{25921} \right) = \begin{pmatrix} 0.3876 \\ 0.7752 \end{pmatrix}.$$

Second, we compute the feature factor matrix $\Delta U^{(3)}$ as follows,

$$\begin{aligned}
\Delta U^{(3)} = & \quad \mathrm{X}_{(3)} \left(\overline{U^{(1)}} \odot \overline{U^{(1)}} \right)^T + \mathrm{X}_{(3)} \left(\overline{U^{(2)}} \odot \overline{U^{(2)}} \right)^T \\
= & \quad \mathrm{X}_{(3)} [a_1 \odot a_1][(a_1)^T a_1 * (a_1)^T a_1]^+ + \mathrm{X}_{(3)} [b_1 \odot b_1][(b_1)^T b_1 * (b_1)^T b_1]^+ \\
= & \quad \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \left[\begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} \odot \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} \right] \left[\begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}^T \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} * \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix}^T \begin{pmatrix} 3/4 \\ 9/4 \end{pmatrix} \right]^+ \\
& + \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \left[\begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix} \odot \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix} \right] \left[\begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix}^T \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix} * \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix}^T \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix} \right]^+ \\
= & \quad \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \begin{bmatrix} 9/16 \\ 27/16 \\ 27/16 \\ 81/16 \end{bmatrix} [(45/8) * (45/8)]^+ + \begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 6 & 4 & 12 \end{bmatrix} \begin{bmatrix} 25921/2015 \\ 51842/2015 \\ 51842/2015 \\ 103684/2015 \end{bmatrix} [(129605/2015) * (129605/2015)]^+ \\
= & \quad \begin{pmatrix} 1.2444 \\ 2.4889 \end{pmatrix} + \begin{pmatrix} 0.1088 \\ 0.2177 \end{pmatrix}.
\end{aligned}$$

Third, the enhanced factor matrix $\overline{U^{(3)}} = U^{(3)} + \Delta U^{(3)}$ is computed based on the above results as follows:

$$\begin{aligned}
\overline{U^{(3)}} &= U^{(3)} + \Delta U^{(3)} \\
&= \mathrm{X}_{(3)} [b_1 \odot a_1][(b_1)^T b_1 * (a_1)^T a_1]^+ + \mathbf{X}_{(3)} [a_1 \odot a_1][(a_1)^T a_1 * (a_1)^T a_1]^+ \\
&\quad + \mathbf{X}_{(3)} [b_1 \odot b_1][(b_1)^T b_1 * (b_1)^T b_1]^+ \\
&= \begin{pmatrix} 0.3876 \\ 0.7752 \end{pmatrix} + \begin{pmatrix} 1.2444 \\ 2.4889 \end{pmatrix} + \begin{pmatrix} 0.1088 \\ 0.2177 \end{pmatrix} = \begin{pmatrix} 1.7408 \\ 3.4818 \end{pmatrix}.
\end{aligned}$$

Finally, we obtain the enhanced factor $\overline{U^{(3)}} = \begin{pmatrix} 1.7408 \\ 3.4818 \end{pmatrix}$.

Based on the above results, the enhanced factor matrices $\overline{U^{(1)}} = a_1 = \begin{pmatrix} 0.75 \\ 2.25 \end{pmatrix}, \overline{U^{(2)}} = b_1 = \begin{pmatrix} 3.5778 \\ 7.1556 \end{pmatrix}$, and $\overline{U^{(3)}} = c_1 = \begin{pmatrix} 161/45 \\ 322/45 \end{pmatrix}$ are obtained. According to the rule of tensor decomposition, these vectors are used as factor matrices to calculate the approximation tensor $\bar{\boldsymbol{\mathcal{X}}}$ as follows,

$$\bar{\boldsymbol{\mathcal{X}}} = \overline{U^{(1)}} \circ \overline{U^{(2)}} \circ \overline{U^{(3)}} = a_1 \circ b_1 \circ c_1$$

$$= \begin{pmatrix} 0.75 \\ 2.25 \end{pmatrix} \circ \begin{pmatrix} 3.5778 \\ 7.1556 \end{pmatrix} \circ \begin{pmatrix} 1.7408 \\ 3.4818 \end{pmatrix}$$

$$= \begin{matrix} & & 9.3429 & 18.6858 \\ & & 14.0135 & 28.0271 \\ 4.6712 & 9.3424 & & \\ 14.0135 & 28.0271 & & \end{matrix}.$$

4.5 Experiment in QoS attribute prediction

This subsection implements the prediction experiments on the web service datasets to evaluate the proposed method TTM.

4.5.1 Web service dataset

We use the web service dataset WSDream offered by Zheng et al. [Zheng, Ma, Lyu, & King, 2010]. This dataset describes real-world QoS attribute evaluation results from 142 users on 4,500 web services over 64 different time slices. This dataset is the main benchmark dataset in web service recommendation and is still under research application [Hasnain et al., 2020] [Pandharbale, Mohanty, & Jagadev, 2021]. In addition, this dataset is still under research application.

The experiment is conducted on a Lenovo THINKCENTRE M58 desktop with a 3.0 GHz Intel Core™ 2 Duo *CP*U and an 8 GB RAM, running *Ubuntu* operation system. The program is implemented with Python 3.4 and Microsoft C++.

We use the standard mean absolute error (*MAE*), and root mean square error (*RMSE*) to

compare the quality of our prediction [Zheng et al., 2010]. The calculation formula or *MAE* and *RMSE* are

MAE is defined as follows:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|}{N}$$

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \hat{r}_{i,j})^2}{N}}$$

where $r_{i,j}$ denotes the expected QoS attribute of Web service j observed by user i, $\hat{r}_{i,j}$ is the predicted QoS attribute, and N is the number of the predicted value.

In this thesis, we focus on the mean value of the error between the QoS value predicted and the actual QoS value accessed service. MRSE and MAE are the most popular metrics, and they mainly describe the difference in QoS predicted by the algorithm, which is between the service that the user expects to access and the service that is accessed. Since the errors are squared before averaging, the RMSE gives relatively high weight to large errors. In addition to QoS service prediction, recommendation systems also use Normalized Discounted Cumulative Gain (NDCG), precision, and recall metrics. These metrics are mainly used to measure the quality of a set of recommendation lists. However, the recommendation lists are not in the context of the discussion in this thesis. Therefore, we used the MRSE and MAE as our evaluation criteria.

We verify the proposed algorithm's effectiveness, and the comparison is based on service collaboration with the following other methods.

- Probabilistic Matrix Factorization (PMF) This method is a probabilistic method using Gaussian assumptions on the data matrices. [Mnih & Salakhutdinov, 2007].
- User-based collaborative filtering method using Pearson Correlation Coefficient (UPCC): this generates a prediction based on similar user behavior [Shao et al., 2007].

- Item-based collaborative filtering method using Pearson Correlation Coefficient (IPCC): this generates a prediction based on similar item properties [Sarwar, Karypis, Konstan, & Riedl, 2001].
- User-based and Item-based Pearson Correlation Coefficient (UIPCC): this is a hybrid collaborative algorithm combining the UPCC and IPCC methods. The prediction is applied to similar users and similar web services.
- Tensor factorization (TF): This method is a user-service-time model based on RTD. It predicts the QoS attribute by considering the relations among user, service, and time [Zhang, Sun, Liu, & Guo, 2014b].

In this thesis, the above baseline methods are based on a three-dimensional QoS dataset. We need to address the issue of low prediction performance due to few sample features. However, along with the explosive growth of information, the dimensionality and feature types of data are increasing, and our method needs to be extended further to solve the issue. To deal with higher dimensional data, deep learning technology becomes a powerful recommendation tool. Besides, a recent study [Zhou, Wu, Yue, & Hsu, 2019] established a neural network-based approach to predict QoS values in a spatial-temporal context. As part of our future work, this approach can be further integrated into our tensor-based modeling framework for QoS prediction studies of higher-dimensional data.

The baseline methods predict the response time and throughput value with the *MAE* and *RMSE* values. Response time is defined as the persistent time between the user call the service and obtain the response. Throughput value is defined as the average rate of the message numbers per second. The smaller value means the method has high performance. The details of response time and throughput are shown in Table 4.24.

Table 4.24 Response Time Performance comparison in *MAE*

	Response Time (second)	**Throughput (kbps)**
Scale	0-200	0-1000
Mean	0.6840	7.2445

Since a user does not revoke all web services, the dataset is usually sparse in the real world. The implement will randomly remove QoS attribute with different density from 5%, 10%, 15%, 20%, 25%, and 30%. The 5% density means that 5% of the data is used for training, and 95% of the data is used for testing.

4.5.2 Recommendation performance evaluation

The comparison result of this experiment is presented in Table 4.25 to 4.28, and the discussion is introduced in the following subsections.

In table 4.25 and Table 4.26, the TTM has smaller *MAE* and *RMSE* values for most densities.

Table 4.25 Response Time Performance comparison in *MAE*

Methods	***MAE***					
	Density 5%	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
PMF	0.9252	0.8305	0.7932	0.7704	0.7538	0.7412
UPCC	0.9373	0.8496	0.7980	0.7688	0.7477	0.7309
IPCC	1.0290	0.9458	0.9243	0.8972	0.8740	0.8581
UIPCC	0.9329	0.8478	0.8002	0.7711	0.7496	0.7329
TF(RTD)	0.8227	0.7792	0.7451	0.7484	0.7332	0.7343
TTM	**0.6858**	**0.6730**	**0.6678**	**0.6698**	**0.6622**	**0.6662**

Table 4.26 Response Time Performance comparison in *RMSE*

Methods	***RMSE***					
	Density 5%	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
PMF	2.2624	2.0070	1.8774	1.7991	1.7472	1.7105
UPCC	1.8935	1.7854	1.7363	1.7012	1.6724	1.6478
IPCC	2.0181	1.8766	1.8464	1.8189	1.7922	1.7679
UIPCC	1.8860	1.7832	1.7361	1.7003	1.6695	1.6429
TF(RTD)	1.8562	1.7852	1.7459	1.7359	1.7224	1.7233
TTM	**1.5922**	**1.5745**	**1.5638**	**1.5660**	**1.5613**	**1.5592**

In Table 4.27 and Table 4.28, the TTM obtains smaller *MAE* and *RMSE* values for throughput with different matrix densities. Thus, TTM achieves better performance than others.

Table 4.27 Throughput Performance comparison in *MAE*

Methods	***MAE***					
	Density 5%	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
PMF	6.5653	5.9829	5.8317	5.7069	5.5513	5.3889
UPCC	10.3860	9.5014	8.9477	8.4906	8.1474	7.8938
IPCC	10.0405	9.6518	9.5135	8.9333	8.3484	7.9666
UIPCC	9.8959	9.3041	8.9638	8.3983	7.8756	7.5158
TF(RTD)	4.2583	4.2046	4.1276	4.0935	4.1902	4.2415
TTM	**4.1921**	**4.0458**	**4.0906**	**4.0514**	**4.0290**	**3.9897**

Table 4.28 Throughput Performance comparison in *RMSE*

Methods	***RMSE***					
	Density 5%	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
PMF	40.3278	35.9576	33.8194	32.4923	31.1695	30.2369
UPCC	43.2909	40.7598	38.8087	37.1719	35.6727	34.6294
IPCC	45.3464	43.1114	42.4567	41.0629	39.2411	37.8419
UIPCC	43.9639	41.5855	40.1955	38.5635	36.7211	35.2921
TF(RTD)	24.0221	23.423	**21.9477**	**21.6142**	21.8390	21.8049
TTM	**22.8952**	**22.5691**	22.6834	22.0979	**21.6888**	**21.6414**

The web service dataset is commonly very sparse since a service user just invokes a very small number of web services usually. We dismiss QoS attribute value to sparse the dataset and access the sparser dataset with different density from 5% to 30%, ascending by 5% each time. For instance, a dataset density 5% means that we leave 5% of the dataset for training at random, and the other 95% value is the testing set.

With the increase of the training matrix density from 5% to 30%, the prediction accuracy in the methods can also be improved. It indicates that the prediction has high accuracy if data with more significant density provides more QoS attribute values. The TTM can significantly improve the accuracy result of a sparse tensor if more information is provided. Better result's reason is that more contextual information that influences the client-side QoS attribute prediction’s performance (e.g., the service servers' workload, network conditions of the users) should be considered to improve the prediction accuracy.

4.5.3 Impact of tensor density

Tensor density impact relates to a finite number of latent factors. Figure 4.7 and Figure 4.8 are the *RMSE* and *MAE* of response-time. We noticed that all methods possessed high *MAE*/*RMSE* values in lower tensor density. The result shows that the sparse tensor needs to provide more information if improving the prediction. With the training density rise, prediction performance is enhanced by the TTM.

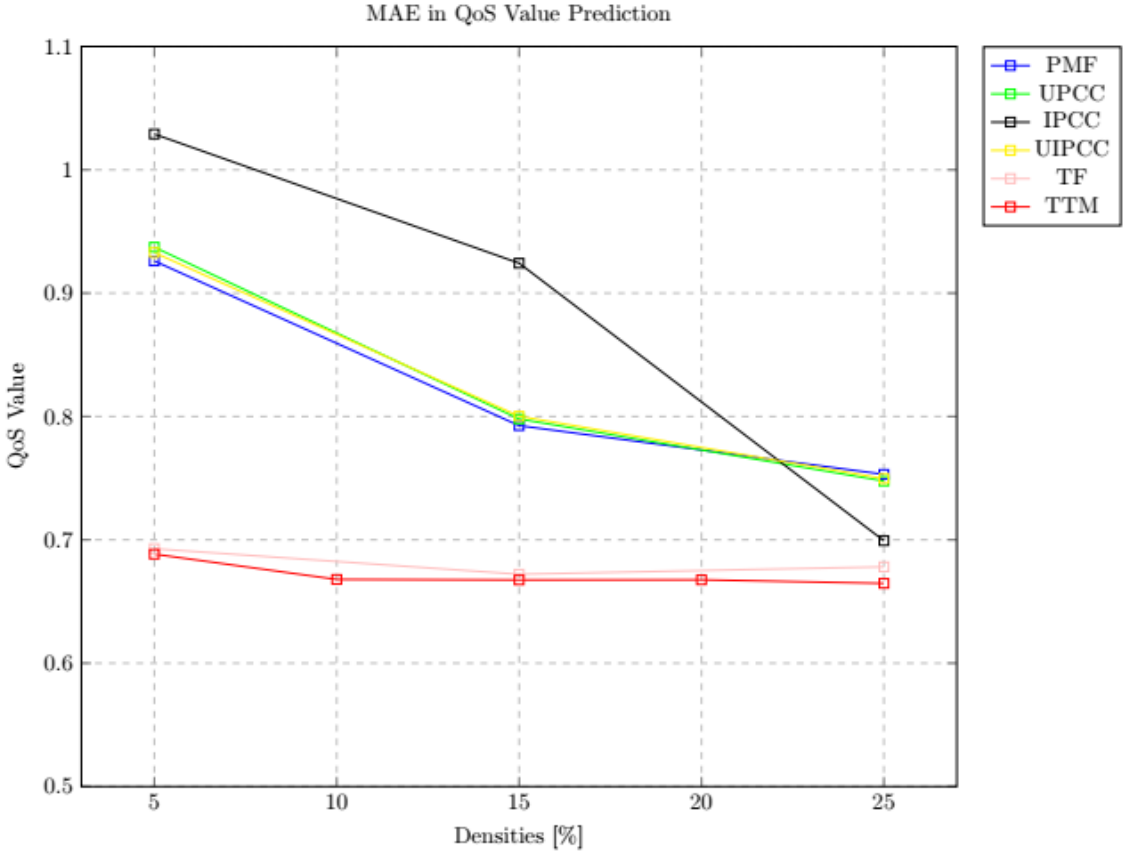


Figure 4.8 *MAE* in QoS attribute prediction

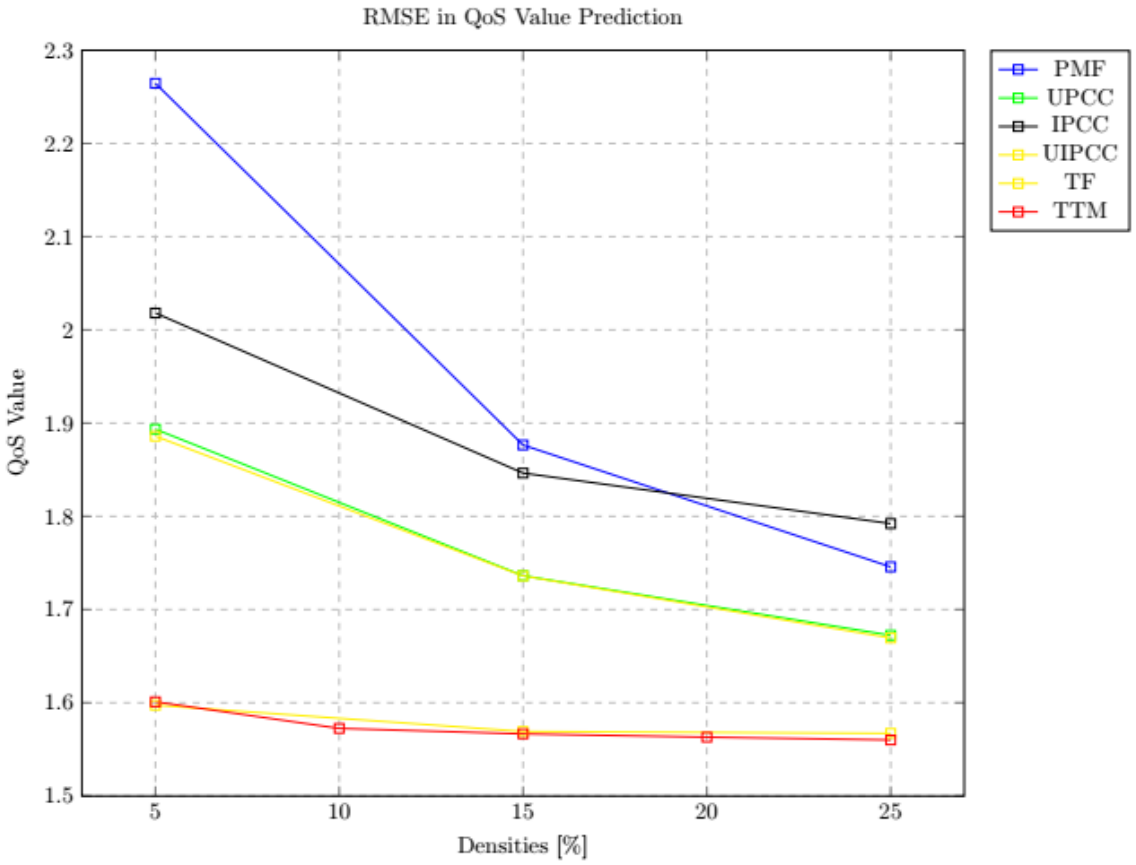


Figure 4.9 RMSE in QoS attribute prediction

4.5.4 Execution time comparison

We record the execution time as computation performance. In Table 4.29, the execution time of the methods is evaluated for different performances.

Table 4.29 Execution time comparison

	Response time computation HH: MM: SS	**Throughput computation HH: MM: SS**
PMF	2:35:51	2:36:38
UPCC	0:17:7	0:25:0
IPCC	0:17:7	0:25:0
UIPCC	0:17:7	0:25:0
TF(RTD)	7:51:26	5:0:0
TTM	8:13:43	5:16:36

It is evident from Table 4.29 that the UPCC, IPCC, and UIPCC methods have better execution performance than other methods. These methods require less dimensional data to computation the performance. The total execution time has around 60 minutes against PMF running time of 155 minutes, TF running time of 471 minutes, and TTM running time of 493 minutes for the response time computation.

UPCC, IPCC, and UIPCC methods execution performance are better (75 minutes). Moreover, the TTM execution performance is expected to be better than the TF one because the TTM has more factors to calculate.

4.5.5 Summary of experiment

The experimental result shows that the QoS attribute can be predicted using the TTM, which adds all user-service-time feature factor matrices. When the enhanced factor matrices in each iteration are obtained, the prediction results enhance the accuracy rate. Moreover, TTM makes predictions by constructing more sample data. With the increase of missing data, the accuracy of the other methods has a relatively large decrease, while TTM reaches higher prediction performance than other methods.

On the other hand, although TTM spends much time generating the feature factor matrices, the overall algorithm does not significantly increase running time than the RTD.

4.6 Experiment on recovering the missing traffic flow data

To evaluate the TTM, we applied the TTM on a traffic flow prediction and obtained positive results. Tensor decomposition is a part of the recovery missing data method with high accuracy and high applicability. We adopt TTM and RTD to the decomposition part of the recovery method with different missing rates. The comparative analysis is completed between two methods' effects in missing data cases.

4.6.1 Traffic flow prediction

Accurate traffic flow prediction information can help city managers make a traffic control decision and help drivers choose smoother routes to avoid traffic jams. A traffic flow dataset is mainly used for traffic flow prediction. Since missing data situations always occur, it is difficult to predict traffic flow accurately. Research of traffic flow prediction with missing data has been popular. The prediction methods are often adopted to recover the missing data. Tensor decomposition has been applied in recovery methods. The factor matrices are extracted by tensor decomposition, and the approximation tensor is generated as the outer product of these factor matrices. By setting the projection function $P_{\Omega}(\boldsymbol{\mathcal{X}})$ of the revealed dataset, the missing data can be recovered with the error between the approximation tensor and the original tensor as the optimization objective. The question of recovering is a tensor decomposition when only a small number of entries are revealed in a traffic dataset as follows [Song, Ge, Caverlee, & Hu, 2019] [Jain & Oh, 2014].

Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathcal{X}$ is recovered by using the given entries $P_{\Omega}(\mathcal{X})$,

$$P_{\Omega}(\boldsymbol{\mathcal{X}})_{ijk} \begin{cases} \boldsymbol{\mathcal{X}}_{ijk} & \text{if } (i, j, k) \in \Omega \\ 0 & otherwise \end{cases}$$

where $P_{\Omega}(\cdot)$ denotes the projection of a tensor $\boldsymbol{\mathcal{X}}$ onto the revealed set. $\Omega \subseteq I_1 \times I_2 \times I_3$ denotes a subset revealed out of $I_1 \times I_2 \times I_3$ entries of $\boldsymbol{\mathcal{X}}$. Each (i, j, k) $i \leqq I_1, j \leqq I_2, k \leqq I_3$ is included in a subset revealed Ω.

In the context of this section, we are concerned with the problem of how to apply TTM in the decomposition part of the recovery method. We assume three values as the initial missing

data: 0, 6, and 55 (0 means no traffic count, 6 means the lowest average traffic count in the current dataset, and 55 means the average of all traffic data in the current dataset).

4.6.2 Traffic dataset

The data are collected in real-time from One-ITS Toronto Traffic Dataset for each intersection of a section of highway in Toronto city [Middleware Systems Research Group, 2020]. It is assumed that the number of vehicles in the inlet lane of the intersection of the section is the traffic volume of the corresponding section. The traffic flow data is collected daily by 162 loop detectors. We construct traffic tensor data, including thirteen relevant road sections, seventeen days, and twenty-four hours per day of relevant traffic information.

Data recovery experiments are implemented for both RTD and TTM. For the simulation of missing data, the experiments randomly select 5%, 15%, ..., 95% of the historical data in 10 sections of 17 days at 5% intervals, and Figure 4.9 shows the example of random missing data.

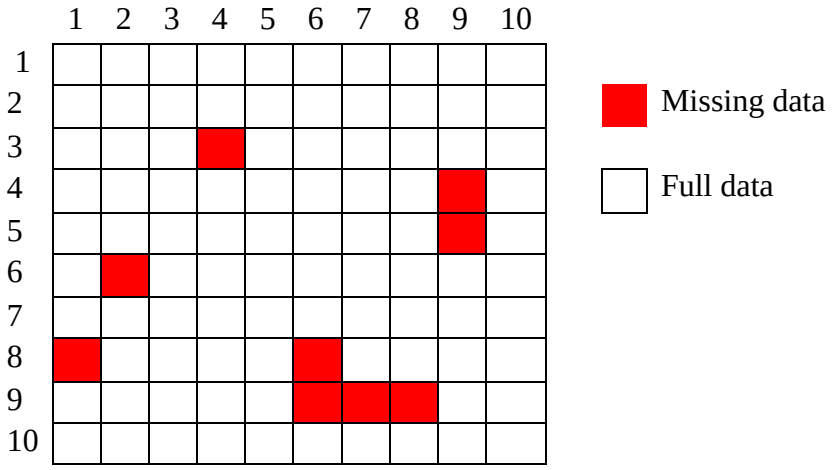


Figure 4.10 Example of random missing data

Two types of errors are used to measure the recovery effect of missing data: relative error *MAE* (mean absolute error) and root-mean-square error *RMSE* (root-mean-square deviation). The two types of errors are calculated as follows.

MAE is defined as follows:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|}{N}$$

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \hat{r}_{i,j})^2}{N}}$$

where $r_{i,j}$ denotes the expected QoS attribute of web service j observed by user i, $\hat{r}_{i,j}$ is the predicted QoS attribute, and N is the number of predicted values.

4.6.3 Data recovery performance

Table 4.30 and Table 4.31 show the recovery errors of the two methods in the case of ten times simulation of random missing data, and the results are the average of 10 round experiments for each missing rate according to the literature. The larger the value of the missing rate, the fewer data are available in the dataset. The smaller the error value, the better the recovery performance.

Table 4.30 *MAE* errors in the random missing rates

Initial value	**Number of flows=0**		**Number of flows= 6**		**Number of flows= 55**	
Missing rate	**TTM**	**RTD**	**TTM**	**RTD**	**TTM**	**RTD**
5%	0.5284	0.1876	0.543	0.1897	0.6273	0.1999
10%	0.4601	0.2013	0.4768	0.1949	0.6543	0.2151
15%	0.3942	0.2164	0.4108	0.2141	0.6731	0.2425
20%	0.3238	0.2399	0.3577	0.2304	0.6999	0.271
25%	0.2687	0.2733	0.3056	0.2574	0.7303	0.3024
30%	0.2319	0.3086	0.2641	0.2864	0.7512	0.3333
35%	0.2151	0.3484	0.2364	0.3192	0.7892	0.3728
40%	0.2143	0.3909	0.2257	0.353	0.8211	0.4048
45%	0.2391	0.4368	0.229	0.3901	0.8553	0.446
50%	0.2774	0.4835	0.2495	0.4284	0.8971	0.4826
55%	0.3314	0.5333	0.2826	0.4688	0.9318	0.5224
60%	0.3929	0.5832	0.3223	0.5096	0.9808	0.5655
65%	0.4628	0.6345	0.3707	0.5508	1.0316	0.6121
70%	0.5347	0.6855	0.4235	0.593	1.0821	0.6523
75%	0.6095	0.7371	0.4799	0.6355	1.1379	0.6983
80%	0.6869	0.7901	0.5388	0.6786	1.1884	0.7442
85%	0.7659	0.8439	0.5999	0.7208	1.2472	0.7904
90%	0.8467	0.8973	0.6628	0.7641	1.309	0.8407
95%	0.9264	0.9498	0.7261	0.8062	1.3748	0.8879

Table 4.31 *RMSE* errors in the random missing rates

Initial value	**Number of flows=0**		**Number of flows= 6**		**Number of flows= 55**	
Missing rate	**TTM**	**RTD**	**TTM**	**RTD**	**TTM**	**RTD**
5%	35.2342	13.7778	35.803	13.5521	38.8253	13.2141
10%	30.0366	15.7724	30.8491	15.2414	37.134	13.641
15%	25.2614	18.1651	26.1761	17.6964	35.3394	14.548
20%	20.7516	20.9704	21.9691	20.1599	33.7469	15.5895
25%	17.0253	24.2158	18.2339	22.9864	32.4756	16.8063
30%	14.8968	27.5279	15.5684	26.1248	31.3799	18.3777
35%	14.8014	30.7798	14.449	29.3071	30.5568	19.8605
40%	16.7132	34.3626	15.4077	32.4305	30.014	21.3546
45%	20.3172	37.9138	17.9337	35.8464	29.7642	22.9892
50%	24.5642	41.6046	21.7558	38.9337	29.842	24.6458
55%	29.6531	45.2418	25.8747	42.5423	30.1475	26.5184
60%	34.6874	49.0597	30.3599	46.0807	30.8274	28.2337
65%	39.6862	53.0356	35.3599	49.3968	31.7732	30.0507
70%	45.2889	56.4309	40.1336	52.8259	32.9926	31.8841
75%	51.1029	60.1379	45.3329	56.3854	34.4334	33.5332
80%	56.5173	63.877	50.3796	59.8021	35.959	35.4371
85%	62.0534	67.6802	55.7179	63.3642	37.7278	37.3047
90%	67.7637	71.5787	60.9097	66.9731	39.6247	39.1858
95%	73.5912	75.2673	66.1629	70.3741	41.5641	40.9428

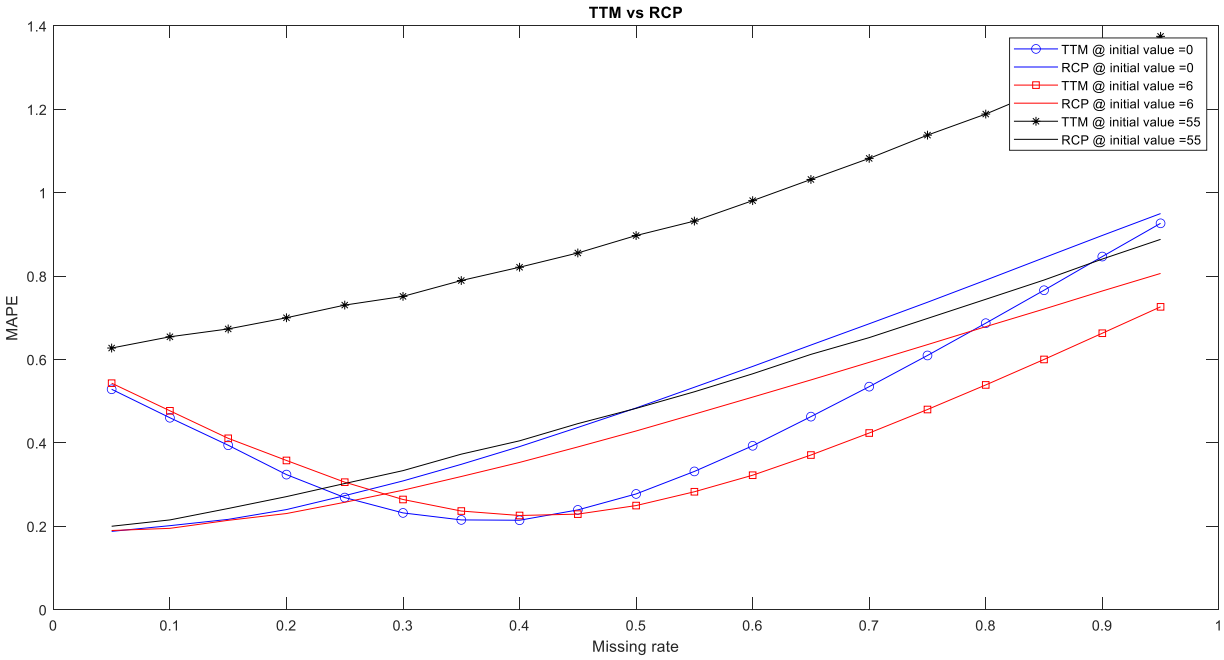


Figure 4.11 *MAE* error curve of TTM vs. RTD

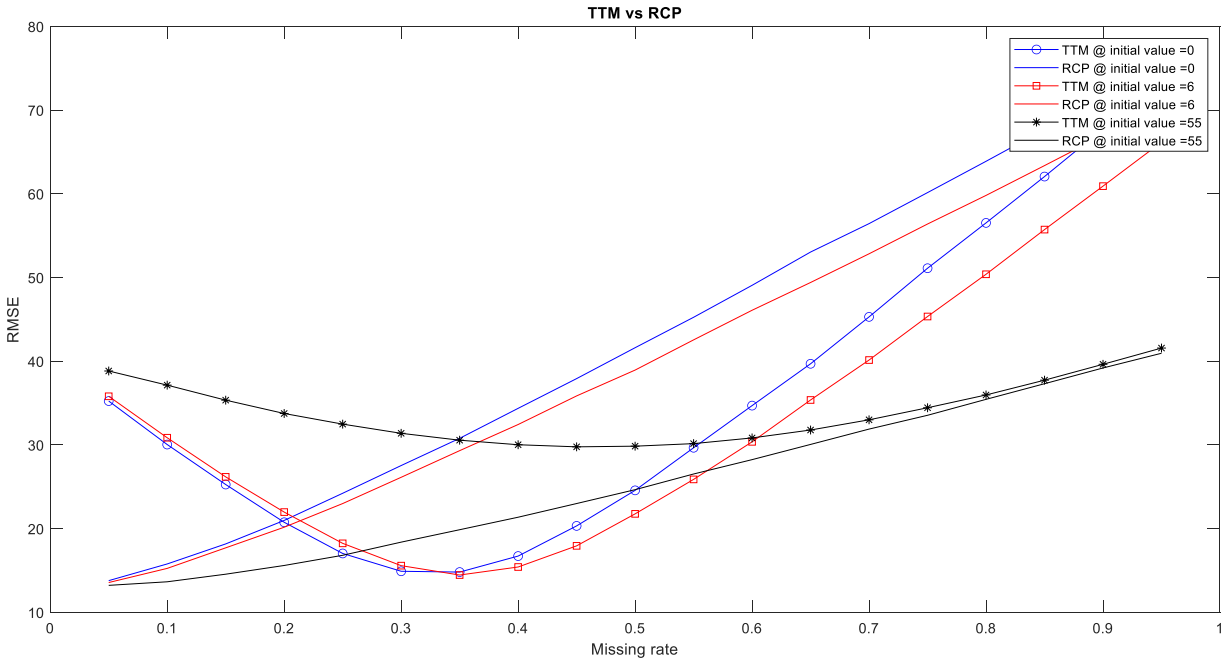


Figure 4.12 *RMSE* error curve of TTM vs. RTD

Figure 4.10 and Figure 4.11 show that the effectiveness of RTD recovered data does not differ much for different missing rates, and all errors increase with the increasing missing rate. However, the recovery performance of TTM needs to be viewed in segments. When the missing rate is lower than 30-35%, the *MAE* generated by the TTM decreases with the increase of the missing rate. When the missing rate is higher than 30-35%, his *MAE* decreases with the increase of the missing rate. Moreover, the error generated by the TTM is lower than those generated by the RTD, which means that the TTM is more effective when the missing data rate is higher than 30-35%.

It can also be found that the difference in *MAE* between TTM and RTD is the largest when the missing rate reaches 50%. The difference of *RMSE* between TTM and RTD is the largest when the missing rate reaches 60%.

4.6.4 Impact of initial missing values

In the traffic dataset, the missing data is represented by a null value. However, before the tensor decomposition, it is usually necessary to set an initial value for the null element of the original tensor. We call this setting the initial missing value.

In order to further investigate the recovery effect of TTM and RTD, a comparison of the recovery error rates is conducted by selecting different initial missing values. Table 4.32 shows the *MAE*/*RMSE* average with initial missing values under different initial missing values. We use three initial values, such as lowest flow value 0, lowest average flow value 6, and average flow value 55. The results show how the initial missing value influences the TTM and RTD as follows.

- The initial missing value is settled as 0: the *MAE* average reaches the lower values for both TTM and RTD.
- The initial missing value is settled as 6: the *MAE* average reaches the lowest values for both TTM and RTD.
- The initial missing value is 55: the *MAE* average reaches the top values for TTM, and the *RMSE* average reaches the lowest values for RTD.

In tensor decomposition, the initial values are usually estimated based on experience, such as zero values, the average of the observed data, etc. This estimation is often adjusted until a better prediction performance is obtained.

In this experiment, three initial values were selected by experience. In general, the recovery performance of TTM is better than RTD when the initial value is lower than the average traffic flow value. When the initial value is higher than the average traffic flow value, the recovery performance of TTM becomes worse than RTD. The initial missing value should be as close as possible to the lowest average value.

Table 4.32 *MAE*/*RMSE* average with initial missing values

Initial value	**Content**	***MAE* average**		***RMSE* average**	
		TTM	RTD	TTM	RTD
0	No flow	0.46	0.51	35.78	42.49
6	Lowest Average flow	0.41	0.45	33.07	40.00
55	Average flow	0.94	0.50	33.90	25.48

4.6.5 Summary of experiment

We implement the missing data recovery experiments based on TTM and RTD. The experimental results show that the recovery error of the RTD increases gradually with the increase of the missing rate. For different random missing rates, the recovery error of the TTM all varies as the missing rate increases.

By further analyzing the recovery errors under different initial missing rates, it is found that how the initial missing values more influence the TTM. When the missing rate is lower than 30-35%, the recovery error of the TTM is larger than that of the RTD. In the case of the same large initial missing rate, for example, the missing rate is higher than 35%, the recovery error of the TTM is smaller than that of the RTD, which means that the TTM is better at this time.

4.7 Summary

We addressed the problem of the lack of samples for the high-dimensional limited sample dataset, and the goal is set on how to construct more sample records from the current limited information, expecting to improve the prediction performance. The feature factor matrix as a features-oriented collaboration scheme is established, and an improvement method TTM is implemented according to the definition of this scheme. The iteration step size of the TTM is increased making the convergence efficient, and the TTM converges with better accuracy than other methods for the same number of iterations.

At the core of the work presented in this chapter, we argue for the importance of

- **(1)** designing a feature-oriented collaboration scheme, which mapping the limited sample into feature factor matrices as the base of the method.
- **(2)** proposing TTM, the feature-oriented tensor decomposition algorithm based on the regular tensor decomposition for fitting the feature-oriented collaboration scheme.
- **(3)** establishing a comparative analysis to validate the TTM, including discussing the convergence properties, verifying the convergence results, and presenting the computational complexity.
- **(4)** conducting experiments on the real datasets to reach data prediction and recovery application requirements. The experiments demonstrate that TTM enhances the level of prediction accuracy.

Overall, TTM outperforms RTD in predicting and recovering information with a consistent rate of missing data.

Chapter 5

A Modified Regularization Term

This chapter focuses on a modified regularization term for supporting TTM. The lasso and ridge regression are introduced separately. The modified regularization term is formulated. The experimental results effectively improve the QoS attribute prediction performance in the traversal tensor method (TTM).

5.1 Introduction

We propose a novel regularization term for tensor decomposition based on TTM. The major novelty is a combination method for estimating prediction results for the web service recommendation, which simultaneously exploits lasso and ridge regression. Using this modified regularization term, we can prevent them from the overfitting problem.

The main contributions of this chapter are:

(1) Identify the lasso and ridge regression and find an effective way to help to predict the QoS attribute.

(2) By combining the lasso and ridge regression regularization, the modified regularization term based on TTM could enhance the prediction performance and reduce overfitting.

The chapter's structure follows: Section 5.2 introduces a motivation and reviews our previous research work. Section 5.3 introduces the regularization techniques: ridge regression, lasso regression, and Elastic Net regression, respectively. Section 5.4 shows the modified regularization term based on TTM to obtain the solution of the factor matrix. Section 5.5 applies the proposed algorithm to the web service prediction problem in the experimental and analyzes the results. Finally, Section 5.6 summarizes the work of this chapter.

5.2 Motivation

Overfitting is a phenomenon in which the method is overfitted to the observed data due to few sample data in the analysis method so that the prediction by the method will be very different from the expected value. The recommendation performance based on tensor decomposition is usually negatively affected by the overfitting problem and, consequently, cannot achieve state-of-the-art performance. This often requires regularization techniques to enhance decomposition performance.

Commonly used regularization techniques are lasso, ridge, and Elastic Net regressions [Ogutu, Schulz-Streeck, & Piepho, 2012] [Ji, Wang, Li, & Liu, 2019]. Lasso regression uses the L1 norm, and the model that uses the L2 norm is called ridge regression. Elastic Net regression, also called elastic network regression, combines ridge regression and lasso regression. Signoretto et al. extends the matrix norm to tensor data and uses it for supervised tensor learning to find low-rank projection matrices [Signoretto, De Lathauwer, & Suykens, 2010]. The success of the matrix trace norm inspires Lacroix et al. and they propose a tensor p-norm regularization term [Lacroix, Usunier, & Obozinski, 2018] [Candès & Recht, 2009]. The ridge regression is a popular regularization technique applied to the tensor decomposition modes [Nickel, Tresp, & Kriegel, 2011].

The ridge regression is mainly used to prevent overfitting when all features are extracted from the sample dataset [Zhang, Han, & Jiang, 2016]. However, experiments show that ridge regression might reduce performance for sparse data while the lasso regression has higher efficiency in the sparse dataset [Ruffinelli, Broscheit, & Gemulla, 2019]. Since the web service dataset is sparse, it is desirable to consider a suitable regularization term to avoid overfitting and support the TTM method.

In recent studies, we also note that there are researchers who use the $N3$ method to calculate the norm [(Lacroix, Usunier, & Obozinski, 2018]. It is not suitable for a more general model. We have also conducted corresponding experiments, and the experimental results show no significant difference between those who use the $N3$ method to compute paradigms and our method.

5.3 Regularization techniques

This section introduces the loss function based on ridge regression, lasso regression, and Elastic Net regression method.

The commonly used regularization terms are L1 norm and L2 norm.

- Least absolute shrinkage and selection operator regression (Lasso)

Lasso regression is a regression model that uses the L1 norm $||W||_1$. It is defined to be the sum of the absolute values of each element of the W. In feature selection, the L1 norm helps us minimize the objective function by making W equal to zero to remove these useless features and reduce the interference with the prediction of the sample.

The object function for the lasso regression uses the L1 norm as follows.

$$\|\mathcal{X}\|_1 = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} |x_{i_1 i_2 i_3}|. \quad (5.1)$$

- Ridge regression

Ridge regression is a regression model that uses the L2 norm $||W||_2$. The L2 norm is a square root of the sum of the squares of the values W. The L2 norm makes each element of W small and close to zero. The smaller the parameter, the simpler the model, and the simpler the model is, the less likely it is to produce overfitting. The ridge regression solves the objective function, which is altered by adding a penalty equivalent to the square of the coefficients as follows,

$$\|\mathcal{X}\|_2 = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} x_{i_1 i_2 i_3}^2} = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}. \quad (5.2)$$

- Elastic Net regression

Elastic Net regression is a model that combines ridge regression and lasso regression.

5.4 A modified regularization term

In this section, the solution for optimizing the objective function is introduced. An alternating optimization algorithm is applied in that one of the decomposition elements is optimized at each iteration when other elements are kept fixed.

Regular tensor decomposition commonly uses ridge regression as the regularization term. The essential regularization term $\Omega(\bar{\boldsymbol{\mathcal{X}}})$ is shown in the following formula,

$$\Omega(\bar{\boldsymbol{\mathcal{X}}}) = \frac{1}{2}\lambda \left(\left\| U^{(1)} \right\|^2 + \left\| U^{(2)} \right\|^2 + \cdots + \left\| U^{(N)} \right\|^2 \right) \quad (5.3)$$

where $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximate tensor. $U^{(1)}$ is the factor matrix for user i, $U^{(2)}$ is the factor matrix for service j, and $U^{(3)}$ is the factor matrix for time period k. λ are parameters of the factor matrix in the regularization term.

Motivated by the Elastic Net regression, we propose a modified regularization term that benefited the advantages of both lasso and ridge regressions:

$$\begin{aligned} \Omega(\bar{\boldsymbol{\mathcal{X}}}) &= \lambda \left(\frac{1-p}{2} \|\bar{\boldsymbol{\mathcal{X}}}\|_2^2 + p\|\bar{\boldsymbol{\mathcal{X}}}\|_1 \right) \\ &= \lambda \left(\frac{1-p}{2} \left\| \left[\!\left[U_{\boldsymbol{new}}^{(\mathbf{1})}, U_{\boldsymbol{new}}^{(\mathbf{2})}, U_{\boldsymbol{new}}^{(\mathbf{3})} \right]\!\right] \right\|_2^2 + p \left\| \left[\!\left[U_{\boldsymbol{new}}^{(\mathbf{1})}, U_{\boldsymbol{new}}^{(\mathbf{2})}, U_{\boldsymbol{new}}^{(\mathbf{3})} \right]\!\right] \right\|_1 \right) \end{aligned} \quad (5.4)$$

where $\boldsymbol{\lambda}$ denotes the regularization parameter, and its default value is 35. The parameter $p = 0$ corresponds to the ridge method $||W||_2$ and $p = 1$ to the lasso method $||W||_1$. $\bar{\mathcal{X}}$ denotes an approximation tensor. $U_{new}^{(n)} = U^{(\mathrm{n})} + \Delta U^{(\mathrm{n})}$ $\mathrm{n} = 1,2,3$ denotes the new factor matrices.

We perform an optimization task for the objective function $L(\mathcal{X}, \bar{\mathcal{X}}) = \ell(\mathcal{X}, \bar{\mathcal{X}}) + \Omega(\mathcal{X})$ as follows,

$$\min\left(\ell(\mathcal{X}, \bar{\mathcal{X}})\right) + \Omega(\bar{\mathcal{X}})$$

$$\Rightarrow \min\left(\ell(\mathcal{X}, \bar{\mathcal{X}})\right) + \lambda\left(\frac{1-p}{2}\,||\bar{\mathcal{X}}||_2^2 + p||\bar{\mathcal{X}}||_1\right)$$

$$\Rightarrow \min_{U_{new}^{(n)}, n=1,2,3}\left(\left\|\mathcal{X} - \left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]\right\|\right) + \lambda\left(\frac{1-p}{2}\left\|\left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]\right\|_2^2 + p\left\|\left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]\right\|_1\right)$$

$$\Rightarrow \min_{U_{new}^{(n)}, n=1,2,3}\left(\left\|\mathcal{X} - \left[\!\left[U^{(\mathrm{n})} + \Delta U^{(\mathrm{n})}\right]\!\right]\right\|\right) + \lambda\left(\frac{1-p}{2}\sum_{n=1}^{3}\left|U^{(\mathrm{n})} + \Delta U^{(\mathrm{n})}\right|^2 + p\sum_{n=1}^{3}\left|U^{(\mathrm{n})} + \Delta U^{(\mathrm{n})}\right|\right) \quad (5.5)$$

where the feature factor matrices $\Delta U^{(\mathrm{n})}$ $\mathrm{n} = 1,2,3$ are required to solve in the least squares sense the overdetermined the above equation. The new factor matrices denote as $U_{new}^{(n)} = U^{(\mathrm{n})} + \Delta U^{(\mathrm{n})}$ $\mathrm{n} = 1,2,3$. $\lambda > 0$ is the regularization parameter.

The regularization can be treated as a compromise between finding a small penalty and minimizing the loss function $\ell(\mathcal{X}, \bar{\mathcal{X}}) = \|\mathcal{X} - \bar{\mathcal{X}}\|^2$. The regularization parameter λ controls the compromise: the smaller the λ., the more it minimizes the loss function, and conversely, the smaller the penalty.

The setting of the regularization parameters is related to the size of the dataset. Usually, the regularization parameters are set larger for large datasets and smaller for small datasets. To facilitate comparison with other methods, we use the default value $\lambda = 35$ in the experiment.

The following is the updated algorithm of TTM with the regularization term method (TTMwR), which is the iteration process optimization of $\left[\!\left[U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}\right]\!\right]$ until

convergence. Each iteration computing is performed in two steps: computing the feature factor matrices and updating the iteration.

Algorithm 5.1 TTM with a regularization term

Input: an original tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the regularization term parameter λ, the weight parameter p

Output: the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$, the factor matrices are the index of users, services, and time, respectively.

Step 1. Initialize regular factor matrices $U^{(2)}, U^{(3)}$ and slices $X_{(1)}, X_{(2)}, X_{(3)}$.

Step 2a. Fixing the $U^{(2)}$ and $U^{(3)}$ to estimate the factor matrices $U^{(1)}$, $\overline{U^{(1)}}$.

Step 2b. Compute the corresponding error ε_{new} and ε.

Step 2c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(1)}$.

Step 2d. Compute the corresponding regularization $\lambda\Omega(\overline{\boldsymbol{\mathcal{X}}})$, update the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$

Step 3a. Fixing the $U_{new}^{(1)}$ and $U^{(3)}$ to estimate the factor matrices $U^{(2)}$, $\overline{U^{(2)}}$.

Step 3b. Compute the corresponding error ε_{new} and ε.

Step 3c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(2)}$.

Step 3d. Compute the corresponding regularization $\lambda\Omega(\overline{\boldsymbol{\mathcal{X}}})$, update the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$

Step 4a. Fixing the $U_{new}^{(1)}$ and $U_{new}^{(2)}$ to estimate the factor matrices $U^{(3)}$, $\overline{U^{(3)}}$.

Step 4b. Compute the corresponding error ε_{new} and ε.

Step 4c. Compare the ε_{new} and ε, and set new factor matrix $U_{new}^{(3)}$.

Step 4d. Compute the corresponding regularization $\lambda\Omega(\overline{\boldsymbol{\mathcal{X}}})$, update the approximate tensor $\overline{\boldsymbol{\mathcal{X}}}$

Step 5. Repeat step 2a to step 4d, update the approximate tensor $\overline{\boldsymbol{\mathcal{X}}} = [\![U_{new}^{(1)}, U_{new}^{(2)}, U_{new}^{(3)}]\!]$.

Step 6. Reduce the objective function $L(\boldsymbol{\mathcal{X}}, \overline{\boldsymbol{\mathcal{X}}}) = \|\boldsymbol{\mathcal{X}} - \overline{\boldsymbol{\mathcal{X}}}\|^2 + \lambda\Omega(\overline{\boldsymbol{\mathcal{X}}})$ until convergence is exhausted.

Step 7. Return the final prediction tensor $\overline{\boldsymbol{\mathcal{X}}}$.

5.5 Experiment

In this subsection, we implement the prediction experiments on the web service dataset to evaluate the novel regularization term for tensor decomposition based on TTM.

5.5.1 Experimental setup

To evaluate the proposed QoS attribute prediction method, we use the web service dataset offered by Zheng et al. [Zheng, Ma, Lyu, & King, 2010]. This dataset describes real-world QoS attribute prediction results from 142 users on 4,500 web services over 64 different time slices. This experiment focused on the response time and proposed a method to predict missing QoS attribute values.

The experiment is conducted on a Lenovo ThinkCentre M58 desktop with a 3.0 GHz Intel Core™ 2 Duo *CP*U and an 8 GB RAM, running *Ubuntu* operation system. The program is implemented with Python 3.4 and Microsoft C++.

We use the standard mean absolute error (*MAE*), and root mean square error (*RMSE*) to compare the quality of our prediction. The calculation formula or *MAE* and *RMSE* are

MAE is defined as follows:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|}{N}$$

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \hat{r}_{i,j})^2}{N}}$$

where $r_{i,j}$ denotes the expected QoS attribute of web service j observed by user i, $\hat{r}_{i,j}$ is the predicted QoS attribute, and N is the number of the predicted value.

We verify the effectiveness of the proposed TTMwR method, and the comparison is based on service collaboration with the following other methods.

- Web service QoS attribute prediction framework (WSPred): As a benchmark method, this is a tensor factorization-based recommendation with a time-aware personalized QoS attribute prediction service for different service users [Zhang, Zheng, & Lyu, 2011]. $\boldsymbol{\lambda} = 35$ denotes the default value.
- TTM with regularization term method (TTMwR): This tensor-based method combines lasso and ridge regressions based on TTM. $\boldsymbol{\lambda} = 35$ denotes the default value.
- TTM when $\boldsymbol{\lambda} = 0$: This is a traversal-tensor method without the regularization term.

The above methods predict the response time and compute the *MAE* and *RMSE* values. The smaller value means the method has high performance.

Since a user does not revoke all web services, the dataset is usually sparse in the real world. The implement will randomly remove QoS attribute with different density from 5%, 10%,15%, 20%, 25%, and 30%. The 5% density means that 5% of the data is used for training, and 95% of the data is used for testing. We randomly set the parameter p in TTMwR corresponding to the lasso and ridge. For example, the parameter $p = 0$ is to the ridge, $p = 1$ to the lasso, and $p = 0.25$ means that result is generated by combing 25% by ridge and 75% by lasso regression.

5.5.2 Experimental results and discussion

We examine the prediction performance of three methods.

We set the different parameter $\boldsymbol{\lambda}$ value: when setting $\boldsymbol{\lambda}$ default value is 35, the TTM has the regularization term. When $\boldsymbol{\lambda} = 0$, it means that TTM has no regularization term. TTMwR shows better predictive performance than TTM without regularization and WSPred methods in Table 5.1 and Table 5.2.

Table 5.1 Performance comparison in *MAE*

Methods	$\boldsymbol{\lambda}$	***P***	**Density 5%**	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
WSPred	**35**		0.7913	0.7603	0.7535	0.7629	0.7520	0.7687
TTM	**0**		0.8183	0.7745	0.7417	0.7415	0.7345	0.7382
		0	0.6850	0.6806	0.6723	0.6683	0.6604	0.6693
TTMwR	**35**	**0.25**	0.6892	0.6721	0.6663	0.6672	0.6594	0.6763
		0.5	0.6859	0.6695	0.6721	0.6665	0.6635	0.6629
		0.75	0.6872	0.6711	0.6680	0.6682	0.6667	0.6619
		1	0.6884	0.6679	0.6675	0.6676	0.6647	0.6670

Table 5.2 Performance comparison in *RMSE*

Methods	λ	P	**Density 5%**	**Density 10%**	**Density 15%**	**Density 20%**	**Density 25%**	**Density 30%**
WSPred	**35**		1.8006	1.7741	1.7695	1.7764	1.7780	1.7823
TTM	**0**		1.8569	1.7852	1.7408	1.7366	1.7226	1.7214
		0	1.5891	1.5788	1.5687	1.5680	1.5609	1.5645
TTMwR	**35**	**0.25**	1.5989	1.5710	1.5669	1.5637	1.5617	1.5637
		0.5	1.5959	1.5721	1.5681	1.5635	1.5614	1.5590
		0.75	1.5950	1.5738	1.5673	1.5643	1.5633	1.5595
		1	1.6010	1.5725	1.5666	1.5631	1.5601	1.5621

(1) Accuracy with different methods

The TTMwR method has smaller *MAE* and *RMSE* values for all densities than the other methods in Figures 5.1 and 5.2. The prediction accuracy can also be improved with the training matrix density increase from 5% to 30%. The total average *MAE* of the TTMwR method (0.67) has 14% more than the WSPred method (0.7648). Thus, the TTMwR method can significantly improve the accuracy result.

We also illustrate the evaluation results in the different values of the parameter p separately in Figures 5.3 and 5.4 for the TTMwR method. The result shows that the method has the worst prediction accuracy for 5% density. With density increasing, the accuracy performance curve drop-down during 5% to 25% density. The best accuracy result appears when the density is 25% for both *MAE* and *RMSE*, and the *MAE* result generates a sharp decline curve at the point. Then the curve rises slightly after 25% density, and the accuracy decrease for 30% density.

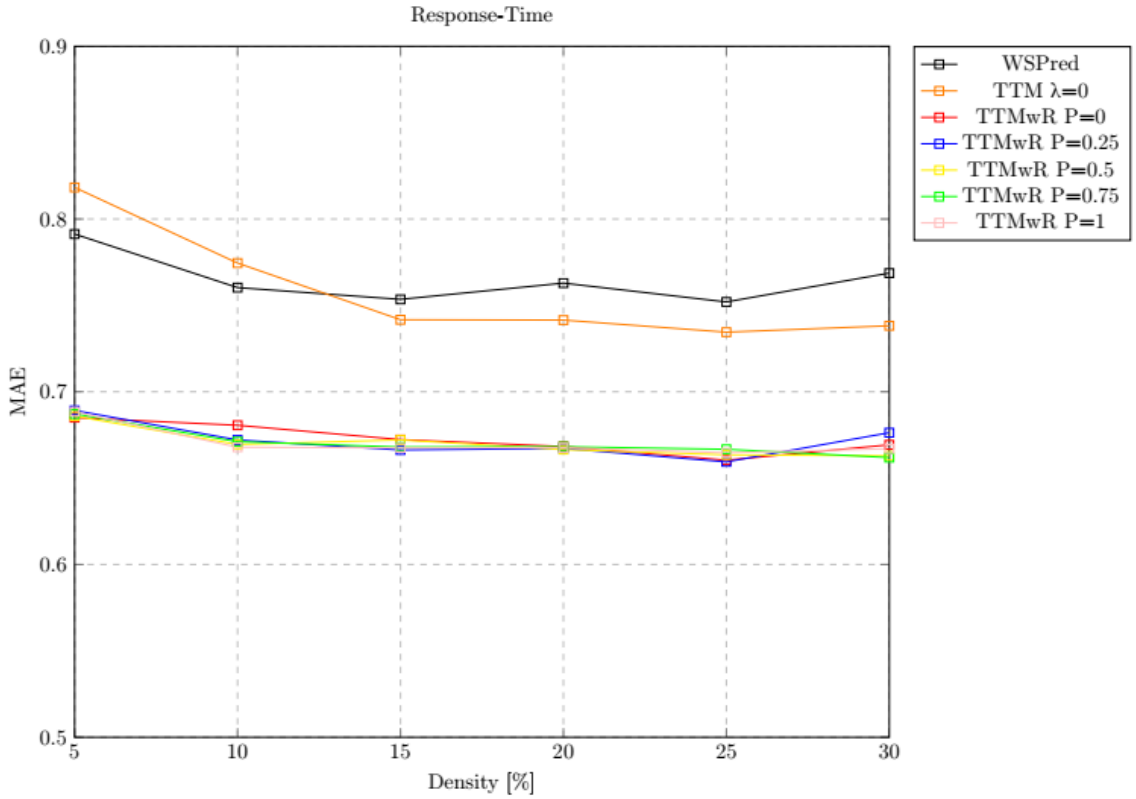


Figure 5.1 Impact of density on prediction accuracy MAE

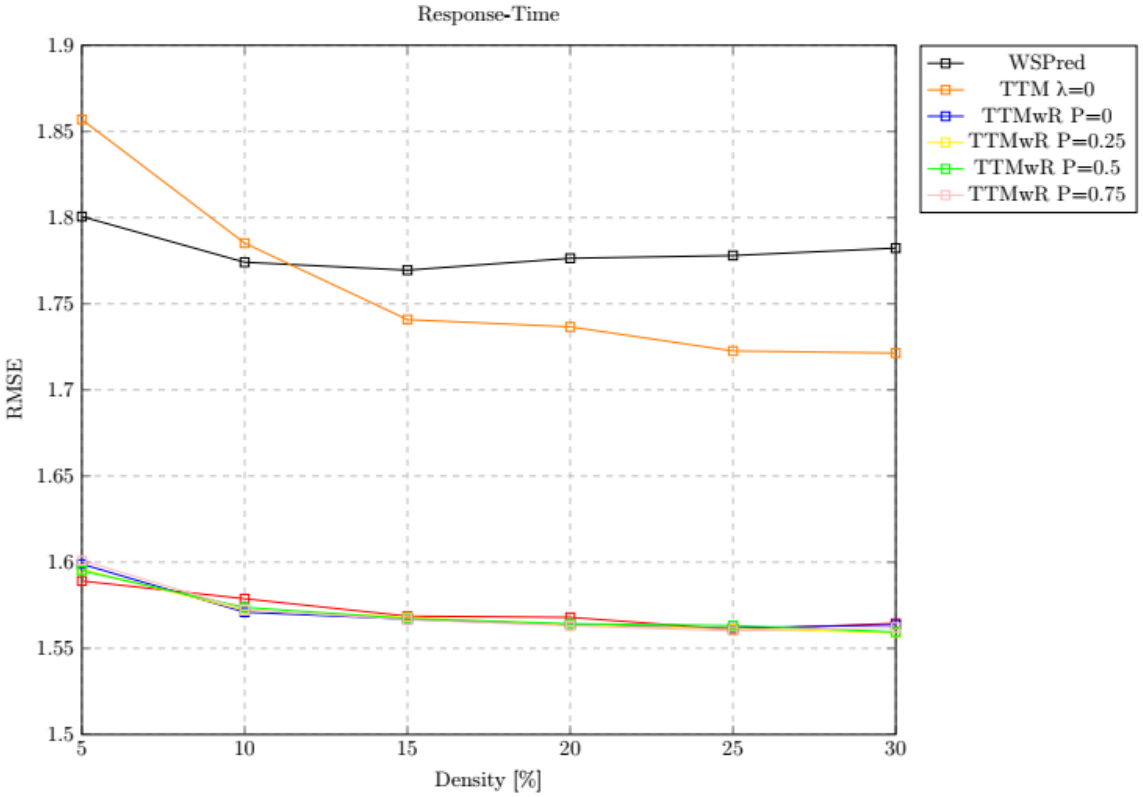


Figure 5.2 Impact of density on prediction accuracy *RMSE*

(2) Impact of the weight parameter p

We focus on the impact of the regression parameter p for the TTMwR method in Figures 5.3 and 5.4. In Figure 5.3, the high *MAE* result is shown for density 10% when $p = 1$, which means the lasso regularization help to achieve a better *MAE* result. As well as the best *RMSE* result appears when $p = 0.25$. Conversely, the minimum *MAE* result is shown when $p = 0.25$, and the minimum *RMSE* result is shown when $p = 1$ from density 15% to 25%. For density 30%, the best *MAE* result is shown when $p = 0.75$, and the best *RMSE* result is shown when $p = 0.5$ *or* 0.75.

For the lower data densities (less than 25%), with the parameter p increases, the accuracy value rises. The ridge regularization contributes more to improve the performance of the method. However, when data density is higher, the performance depends on both lasso and ridge regularization contributes. The best regularization ratio is 25% by the lasso and 75% by the ridge, or an equal split.

Thus, from the experimental results, it can be observed that the impact of the parameter p is not significant. Nevertheless, some subtle differences are observed.

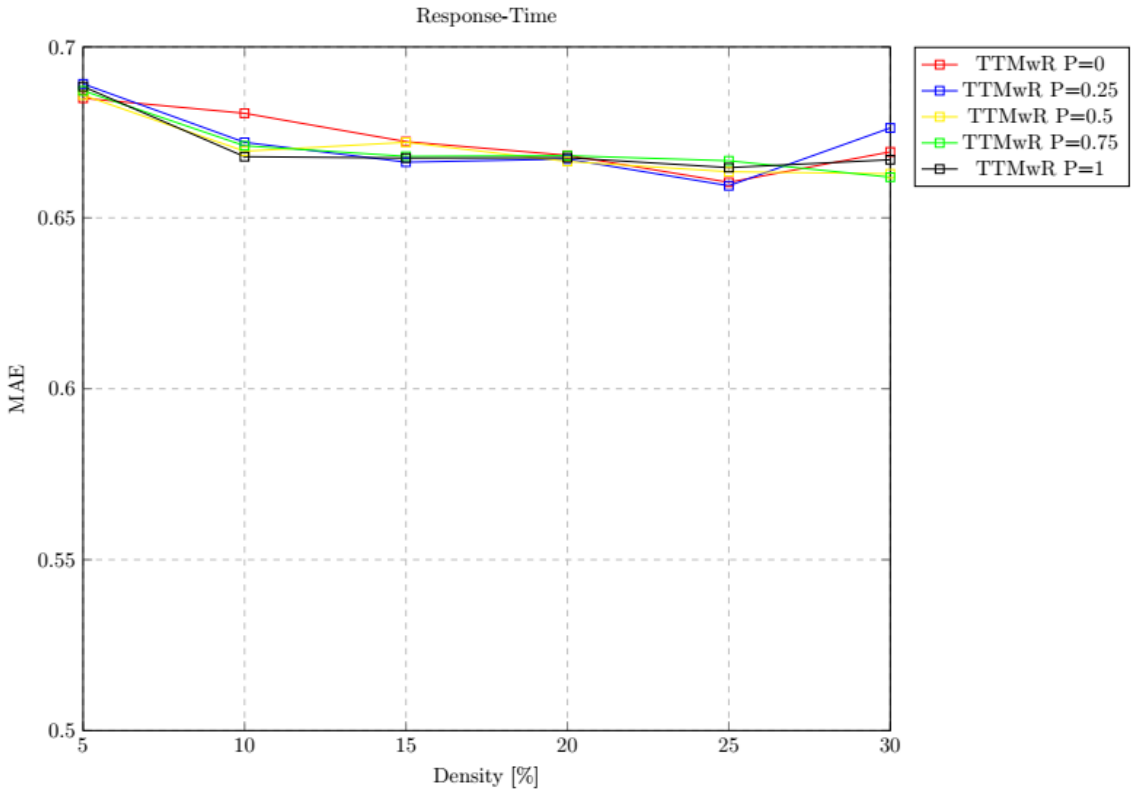


Figure 5.3 Impact of the parameter p on *MAE*

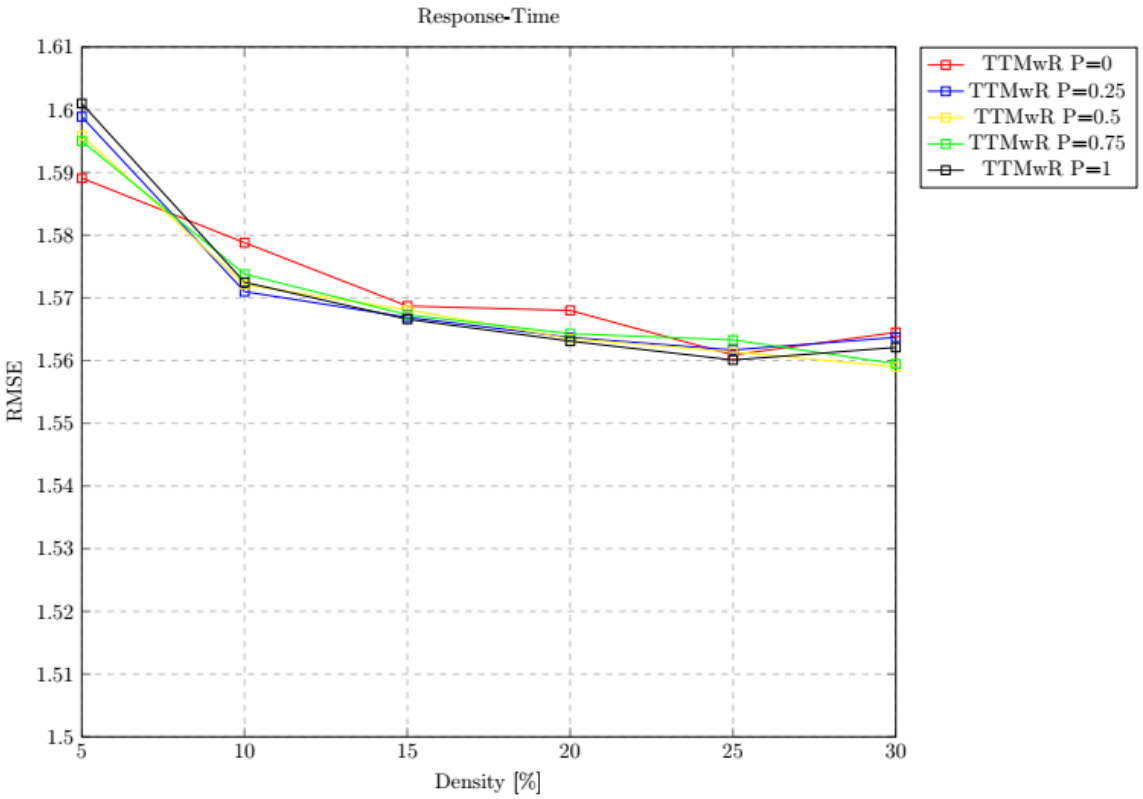


Figure 5.4 Impact of the parameter *p* on *RMSE*

5.6 Summary

A modified regularization term for tensor decomposition based on TTM is proposed. This method aims to reduce the possibility of overfitting and increase the method's robustness. Based on TTM, a combination method for estimating prediction results for the web service recommendation is established, exploiting lasso and ridge regression simultaneously. The experimental results show that the proposed tensor method can effectively improve the estimation performance.

Chapter 6

TTM with K-means Method for Recommendation

This chapter focuses on a two-step strategy based on the K-means algorithm and TTM to deal with the initial unorganized data.

6.1 Introduction

We implement a two-step strategy combining clustering and tensor analysis, which computes the K-means algorithm and TTM tensor decomposition for the initial data preprocessing.

(1) Propose a two-step strategy method.

A two-step strategy method is given as following Figure 6.1.

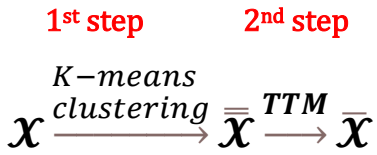


Figure 6.1 Two-step strategy method

- In the first step, the dataset is clustered using the K-means algorithm technique.

  The goal is to find a pre-processing way to deal with the initial unorganized data. We selected two different distance computations, Euclidean and cosine similarity, respectively. The cosine similarity was chosen because it is suitable for calculating angles between two vectors and is insensitive to the absolute length of the label vectors.

- In the second step, after clustering the dataset, the tensor decomposition is performed by applying the TTM.

  The tensor decomposition is used to remove the empty parts of the model, and the approximate tensor is obtained by reconstruction to generate the corresponding recommended values. This method can remove the vacant parts in the dataset to reduce data sparsity.

(2) Analyze the issues such as selecting initial K values, computational performance, and distance calculation are discussed by experimenting on the web service dataset.

The recommendation system's effectiveness based on the clustering technique and tensor decomposition algorithm proposed in this paper is compared with the recommendation effect of similar algorithms.

The chapter is structured as follows. Section 6.2 introduces the motivation and reviews our previous research work. Section 6.3 introduces the K-means algorithm and data clustering process for web service recommendation. In Section 6.4, the TTM with the K-means algorithm for recommendation is proposed. Section 6.5 introduces that the algorithm's performance was evaluated by experimentally discussing two impacts. Finally, Section 6.6 summarizes the entire chapter.

6.2 Motivation

Integrating tensor and clustering algorithms is one of the research topics to use high dimensional data to provide an accurate web service recommendation.

The clustering methods look for hidden valuable information in datasets [Dubey & Choubey, 2017]. The generally used clustering analysis algorithm is the K-means algorithm. The K-means algorithm is currently the most widely used clustering algorithm, which MacQueen first proposed in 1967 [MacQueen, 1967], and Hartigan optimized and implemented an efficient K-means algorithm based on Fortran [Hartigan, 1975]. The K-means algorithm is particularly suitable for classification applications with high-dimensional data [Liu, Hu, Ge, & Xiong, 2012].

The first one to introduce the K-means algorithm into the tensor model is Symeonidis, who applied over the user-tag cluster-item tensor instead of the user-tag-item tensor to represent the ternary relationship in social tagging systems [Symeonidis, 2015]. Silic etc. proposes a K-means algorithm method, CLUStering (CLUS), based on a similar tensor structure using two-dimensional feature vectors for users and services, respectively [Silic, Delac, & Srbljic, 2014]. Shang et al. propose K-means and the time-context-based tensor decomposition method. The initial clustering of datasets is carried out through K-means to improve the data aggregation and algorithm efficiency [Shang, Wang, & Huang, 2018].

When TTM starts running and predicts the QoS attribute, the initial dataset is unprocessed, and the records are sparse and unclassified. If pre-processing of the dataset is implemented, it might improve the prediction performance of TTM. The reason for used the K-means

algorithm is that it is currently the most widely used clustering algorithm, and it is suitable for applications with tensor data. Motivated by the above research, we chose a K-means algorithm to find a pre-processing way to deal with the initial unorganized data. Due to time limitations, we will continue to complete the evaluation based on this study for the other clustering methods in the future.

6.3 K-means algorithm

This section focuses on the K-means algorithm and how K-means is handled in a web service recommendation.

(1) K-means algorithm

K-means algorithm classifies a group of objects into a K number clusters based on the object's attributes or features. First, it requires data points (objects), K number of clusters, and the earliest randomly selected centroid in the dataset. Clustering is determined by two factors: calculating the distance between members and the degree of association with the nearest centroid. The algorithm keeps calculating the distance and the centroid of the partition until the stop condition is met. Euclidean distance is the most used distance measurement method. The stopping condition is that the recalculated centroid no longer changes with iterations, and no members in the result are reassigned.

The K-means algorithm is as follows, for N data objects located in a continuous dimensional space and K number of clusters P, all clusters are independent and have a compact interior.

Algorithm 6.1 K-means algorithm

Input: N dataset, K number of clusters

Output: K clusters P

Step l. Randomly select one object as the centroid q of a cluster.

Step 2. For all m objects, calculate the distance D from the centroid and divide it into the nearest centroid cluster.

Step 3. Recalculate the centroids of K clusters, whose centroid is the average of all object values in the cluster.

Step 4. Repeat steps 2 and 3 until the distance between the objects in the cluster and each centroid is the smallest.

(2) Clustering process

Given a service invocation dataset X which contain N samples,

$$\mathrm{X(u,s,t)} = [x_1, x_2, \dots, x_N]^T \epsilon R^{N\times D}, x_i \epsilon R^D \tag{6.1}$$

where u is the user executing the invocation, s is the service invoked, and t is the actual time of the service invocation. $X(u,s,t)$ is the service response time as QoS attribute value. x_i is a sample data containing N samples.

The goal of clustering is to divide these N samples with the sample similarity degree into K clusters $P = [P_1, P_2, \dots, P_K]$. The K-means algorithm objective function is following [Li & Ding, 2013],

$$\min_{q_k|_{k=1}^K} \sum_{k=1}^K \sum_{x_i \epsilon \mathrm{P}_K} \|x_i - q_k\|^2 \tag{6.2}$$

where q_k is the center of a cluster P_K, let $Q = [q_1, q_2, \dots, q_K] \epsilon R^{D\times K}, V = [v_1, v_2, \dots, v_N]^T \epsilon R^{N\times K}$, where v_i is the category of indicator vector. If $x_i \epsilon \mathrm{P}_K$, then $v_{ik} = 1$, otherwise $v_{ik} = 0$. The K-means algorithm in equation (6.2) can be expressed in the form of matrix decomposition:

$$\begin{cases} \min\limits_{U,V} \|X - QV^T\|^2 \\ s.t. \quad Q^T Q = I \end{cases}. \tag{6.3}$$

K-means algorithm is based on the $X(user, service, time)$ is as follows.

Algorithm 6.2 K-means algorithm

Input: N user-service-time dataset $X(user, service, time)$, K number of clusters

Output: K clusters P

Step l. Retrieve all $user$ items, which is denoted as user cluster $P_{user} = \{u_1, u_2, \dots, u_m, \}, m\ is\ nubmer\ of\ users.$

Step 2. Retrieve all $service$ items, which is denoted as service cluster $P_{service} = \{s_1, s_2, \dots, s_n, \}, n\ is\ nubmer\ of\ services.$

Step 3. Retrieve all $time$ items, which is denoted as time cluster $P_{time} = \{t_1, t_2, \dots, t_l, \}, l\ is\ nubmer\ of\ time.$

Step 4. Select the K number of the cluster as the initial cluster centers $Q = \{q_{user}, q_{service}, q_{time}\}$ for each cluster $P_{user}, P_{service}, and\ P_{time}$ respectively.

Step 5. Calculate the distances D between items of responding cluster and each cluster center $q_{user}, q_{service}, q_{time}$ as follows,

$D_{user}(u_m, q_{user}), D_{service}(s_n, q_{service}), and\ D_{time}(t_l, q_{time})$

Step 6. Set the item belongs to cluster P if the distance D is minimum.

Step 7. Calculate the cluster's average in the same cluster to generate a new cluster center.

Step 8. if the clustering center no longer changes, exit; otherwise, go to step 5 and recalculate.

This clustering algorithm aims to make the generated clusters denser and more independent by dividing items into K clusters. This algorithm is suitable for the scenario that the clusters are relatively dense, and the distinction between clusters is obvious. The K-means algorithm can be used as the basic algorithm in the system analysis as a pavement. Although it is inefficient in handling systems with large datasets due to the increased complexity of the system, the data preprocessing with K-means clustering can make the main algorithm of the system more efficient. This chapter uses TTM as the main algorithm to further process the data after the clustering process.

6.4 TTM with K-means method

This section describes the TTM with the K-means method and algorithm.

Considering a K-means algorithm as preprocessing of TTM for prediction QoS attribute provides an idea to further refine the web service recommendation. Thus, our proposed two-step strategy method, TTM with K-means method, is applied to fit the QoS attribute prediction scenario as follows,

$$\boldsymbol{\mathcal{X}} \xrightarrow[\boldsymbol{clustering}]{\boldsymbol{K-means}} \bar{\bar{\boldsymbol{\mathcal{X}}}} \xrightarrow{\boldsymbol{TTM}} \bar{\boldsymbol{\mathcal{X}}} \tag{6.4}$$

where $\boldsymbol{\mathcal{X}}$ is an original tensor which consists of QoS attribute value, $\bar{\bar{\boldsymbol{\mathcal{X}}}}$ is a clustered tensor after K-means clustering, $\bar{\boldsymbol{\mathcal{X}}}$ denotes an approximation tensor.

First, we compute the mean value in the same context cluster $P_{user}, P_{service}, and\ P_{time}$ for prediction $user, service, time$ items separately with an original tensor $\boldsymbol{\mathcal{X}}$.

Then we apply the $user, service, time$ items to generate a clustered tensor $\bar{\bar{\mathcal{X}}} = [\![U^{(\text{user})}, U^{(\text{service})}, U^{(time)}]\!]$, where $U^{(\text{user})}, U^{(\text{service})}, and\ U^{(time)}$ are the factor matrices after clustering.

Second, this clustered tensor $\bar{\bar{\mathcal{X}}}$ is used as an input tensor of TTM. As same as decomposition processing in Chapter 4, we find the optimal that minimizes the objective function $L(\mathcal{X}, \bar{\mathcal{X}})$ in every iteration step as follows,

$$L(\mathcal{X}, \bar{\mathcal{X}}) = \ell(\mathcal{X}, \bar{\mathcal{X}}) + \lambda\Omega(\bar{\mathcal{X}}) \tag{6.5}$$

where $\mathcal{X}$ is an original tensor which consists of QoS attribute value, $\bar{\mathcal{X}}$ denotes an approximation tensor, $\lambda > 0$ is the regularization parameter, $\ell(\mathcal{X}, \bar{\mathcal{X}})$ denotes as $\ell(\mathcal{X}, \bar{\mathcal{X}}) = \|\mathcal{X} - \bar{\mathcal{X}}\|^2$, $\Omega(\bar{\mathcal{X}})$ denotes a regularization term.

To verify the tensor clustering decomposition method under unsupervised learning, the updated algorithm proposed in this chapter is applied to the QoS attribute prediction and compared with CLUS. The algorithm is a tensor decomposition based on K-means clustering, updating the distance measure with cosine similarity. The algorithm is as follows:

Algorithm 6.3 TTM with K-means Clustering

Input: *N* user-service-time dataset $\mathcal{X}(user, service, time)$, Response time value, the regularization parameter λ, *K* Numbers of clusters.

Output: the approximate tensor $\bar{\mathcal{X}}$, the factor matrices are the index of users, services, and time, respectively.

Step l. K-means clustering process in dataset $\mathcal{X}(user, service, time)$ based on *K* Numbers of clusters.

Step 2. Compute mean value in the same context cluster P_{user} for predicting $user$ items.

Step 3. Compute mean value in the same context cluster $P_{service}$ for predicting $service$ items.

Step 4. Compute mean value in the same context cluster P_{time} for predicting $time$ items.

Step 5. Generate a clustered tensor $\bar{\bar{\mathcal{X}}}$ based on $user, service, time$ items clustered.

Step 6. Inputting the clustered tensor $\bar{\bar{\mathcal{X}}}$, apply the TTM**.**

Step 7. Reduce the loss function until convergence is exhausted.

Step 8. Return the final prediction tensor $\bar{\boldsymbol{\mathcal{X}}}$.

6.5 Experiment

In this subsection, we implement the QoS attribute prediction experiments to evaluate the two-step strategy. This section describes the experiment setup, including methods compared, dataset, hardware system, and evaluation standard. Section 6.3.3 analyses and discusses the number of cluster impactions with accuracy and prediction time. Section 6.3.4 analyses and discusses the distance metrics impaction with accuracy and prediction time.

6.5.1 Experiment setup

We use the web service dataset WSDream, which describes real-world QoS attribute prediction 30,287,611 results from 142 users on 4,500 web services over 64 different time slices. This section focused on the response time and proposed a method to predict the missing QoS attribute.

The experiment hardware is conducted on a Lenovo THINKCENTRE M58 desktop with a 3.0 GHz Intel Core™ 2 Duo CPU and an 8 GB RAM, running Ubuntu operation system. The program is implemented with Python 3.4 and Microsoft C++.

We use the standard mean absolute error (*MAE*), and root mean square error (*RMSE*) to compare our prediction quality. The calculation formulas are

MAE is defined as follows:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|}{N}$$

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \hat{r}_{i,j})^2}{N}}$$

where $r_{i,j}$ denotes the expected QoS attribute of web service j observed by user i, $\hat{r}_{i,j}$ is the predicted QoS attribute, and N is the number of the predicted value.

The above methods predict the response time and compute the *RMSE* values. The smaller the RMSE value, the better the prediction performance. The implement will randomly

remove the QoS attribute with different densities from 5% to 55%. The 5% density means that 5% of the data is used for training, and 95% of the data is used for testing.

6.5.2 Prediction performance

This section analyzes the impacts of the number of clusters on the prediction performance. The number of clusters K of the K-means algorithm is difficult to determine. If the k value is too small, it will lead to significant differences between data objects within the same cluster, and if the K value is too large, it will lead to a close distance between different clusters. Simultaneously, the improper value of K can also lead the final clustering results into local optimum, which is often the most criticized aspect of using the traditional K-means algorithm. As early as 1998, Rezaee et al. [Rezaee, Lelieveldt, & Reiber, 1998] proposed that the optimal K-value is in the range of (1, $\sqrt{n}$), with n being the size of the dataset, which also provided the direction for the later improvement of the traditional K-means algorithm.

We compare our method with the CLUStering (CLUS), a method for predicting web services based on the K-means algorithm. TTM is also compared in the experiment.

- CLUStering (CLUS): This method incorporates user-service parameters and aggregates past data using the K-means algorithm [Silic, Delac, & Srbljic, 2014].
- TTM with K-means method (K+TTM): This is proposed in this chapter. This method is a tensor decomposition TTM based on K-means clustering.
- TTM: This is a traversal-tensor method proposed in Chapter 4.

The above methods predict the response time and compute the *MAE* and *RMSE* values. The smaller value indicates the method has high performance.

To compare the effects of the initial K values, we use different initial K value selection criteria separately. According to the maximum value of optimal K is $\sqrt{n}$ [Rezaee, Lelieveldt, & Reiber, 1998], we have chosen different initial values.

For the dataset which have 142 users on 4,500 web services over 64 different time slices, we set n=142 users, n=4,500 web services, and n=64 time slices separately, and the maximum value of optimal K is obtained as follows,

$K_{user} = 12, K_{service} = 67,$ and $K_{time} = 8$

We have also chosen the same initial values for $K_{user} = K_{service} = K_{time} = 10$ based on the default value [Silic, Delac, & Srbljic, 2014] in Table 6.2.

Table 6.1 Prediction performance comparison in *RMSE*

for $K_{user} = 12, K_{service} = 67$, and $K_{time} = 8$

Methods	***RMSE***					
	Density 5%	**Density 15%**	**Density 25%**	**Density 35%**	**Density 45%**	**Density 55%**
CLUS	2.2015	2.2188	2.1208	2.0068	1.9939	1.9513
K+TTM	2.0150	2.0222	1.9995	1.9488	1.9127	1.8897
TTM	1.5908	1.5675	1.5150	1.5569	1.5589	1.5584

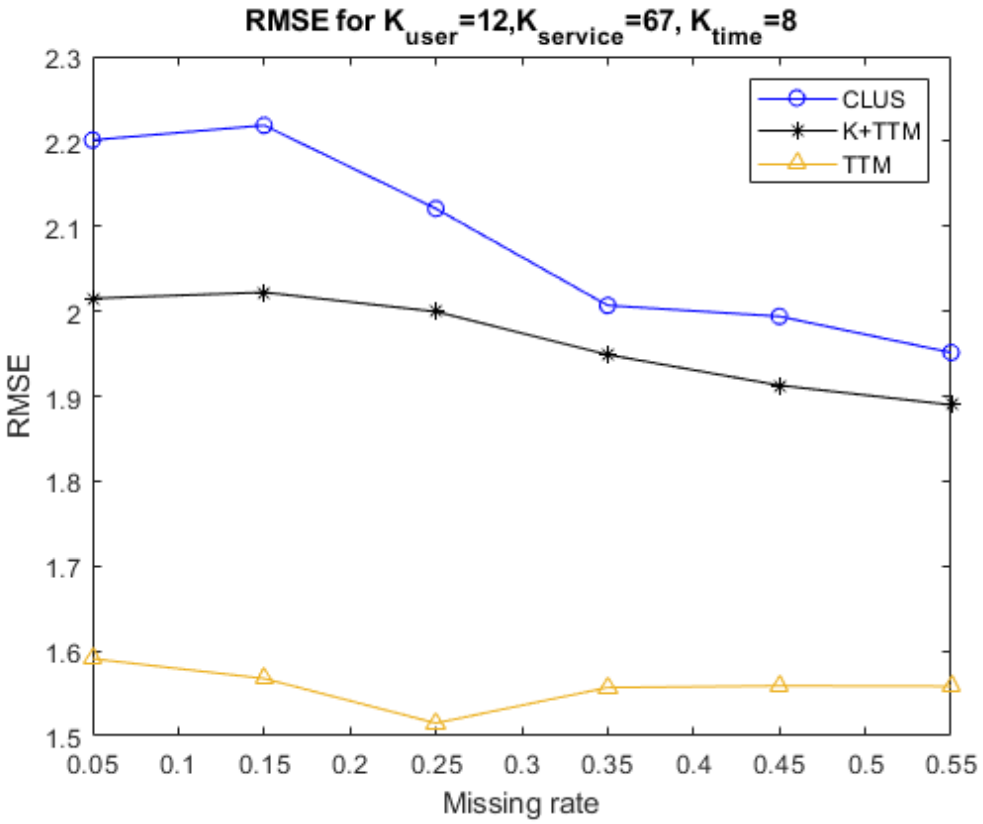


Figure 6.2 Prediction performance comparison in *RMSE*

for $K_{user} = 12, K_{service} = 67$, and $K_{time} = 8$

Table 6.2 Prediction performance comparison in *RMSE*

for $K_{user} = K_{service} = K_{time} = 10$

Methods	***RMSE***					
	Density 5%	**Density 15%**	**Density 25%**	**Density 35%**	**Density 45%**	**Density 55%**
CLUS	2.2320	2.2560	2.1890	2.1118	2.0498	1.9924
K+TTM	2.0192	1.9150	1.8969	1.8597	1.8472	1.8225
TTM	1.5908	1.5675	1.5150	1.5569	1.5589	1.5584

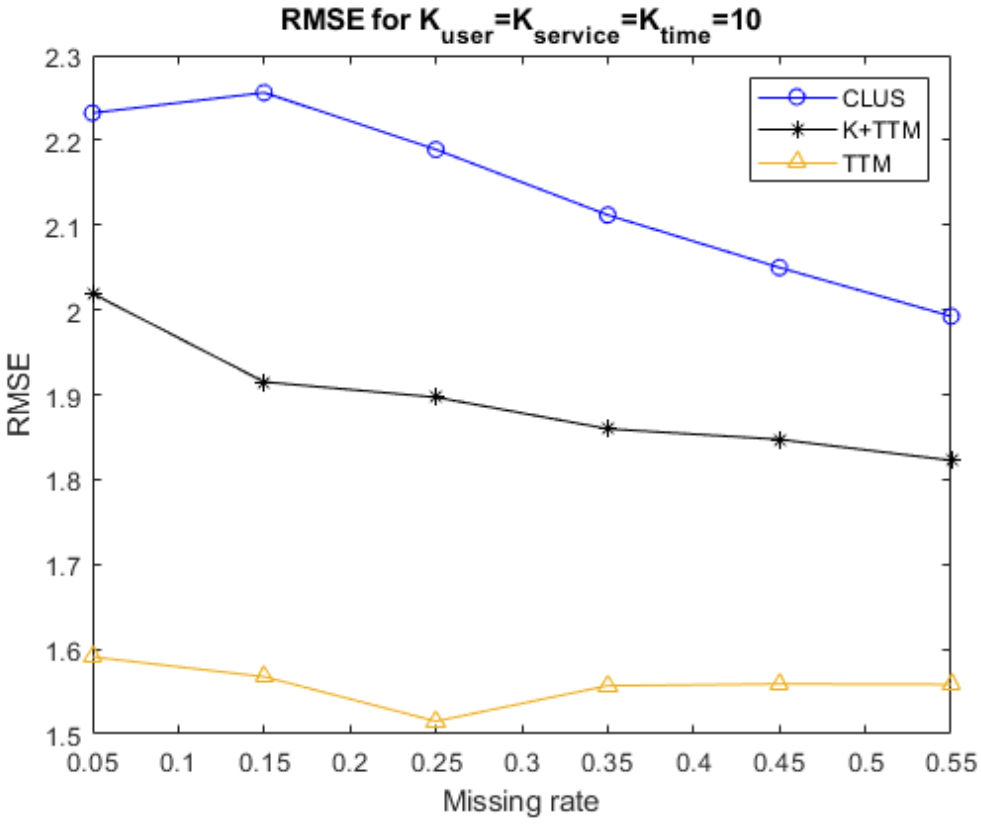


Figure 6.3 Prediction performance comparison in *RMSE*

for $K_{user} = K_{service} = K_{time} = 10$

The K+TTM method has a higher prediction accuracy than the CLUS method. The prediction accuracy results show that accuracy depends on the different data densities in Figure 6.2 and Figure 6.3. For example, for the CLUS method, the *RMSE* value is 2.0068 at 35% density, while the K+TTM's *RMSE* value is 1.9488 at the same density in Table 6.1. It is evident from the presented figures at higher density K+TTM improves more performance *RMSE* value from 2.0150 to 1.8897 while CLUS is in (2.2015, 1.9513) for $K_{user} = 12, K_{service} = 67$, and $K_{user} = 8$ in Table 6.1. The same result is obtained when the number of clusters in $K_{user} = K_{service} = K_{user} = 10$ in Table 6.2.

We find that obtaining the same initial value yields better results for different initial values of the number of clusters than giving different initial values in both methods. *RMSE* value is 1.8897 at 55% density for $K_{user} = 12, K_{service} = 67$, and $K_{time} = 8$, while the K+TTM's *RMSE* value is 1.8225 at the same density for $K_{user} = K_{service} = K_{time} = 10$ in Table 6.2.

The silhouette result also is evident that showed in Figure 6.4.

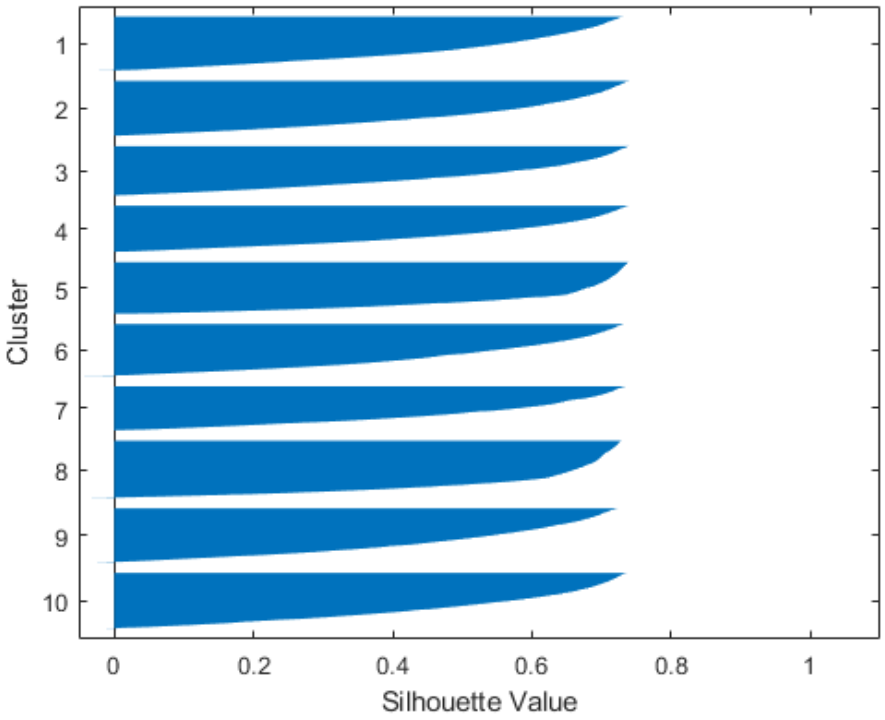


Figure 6.4 Silhouette value comparison

for $K_{user} = K_{service} = K_{time} = 10$

6.5.3 Computational performance

The execution time of the methods is evaluated for computational performances is presented in Table 6.3.

Table 6.3 Execution time comparison

Methods	$\boldsymbol{K_{user} = 12, K_{service} = 67,}$ $\boldsymbol{K_{user} = 8}$	$\boldsymbol{K_{user} = 10, K_{service} = 10,}$ $\boldsymbol{K_{user} = 10}$
CLUS	32 min	27min
K+TTM	11 hours 7 min	11 hours 54 min

The CLUS method has the shortest execution time than the K+TTM method. The CLUS method is not influenced by altering the number of clusters. The reason is that the tensor decomposition needs more complex calculations against CLUS.

The outstanding advantage of clustering is that it is fast, but the accuracy is very low. It is worth the cost of using tensor decomposition in order to get higher prediction values, especially when the hardware computational performance is already high.

6.5.4 Impact of number of clusters

The number of clusters is a parameter that can be adjusted to a specific environment. To evaluate the impact of the number of clusters, we consider the missing rate conditions with various cluster numbers. In the evaluation process, we vary the number of users, service, and time clusters. The initial value of 2 for the number of users, service, and time clusters is chosen. Moreover, we increase the number of clusters from 2 to 67 to calculate the *RMSE* values. Finally, we evaluate the impact of the number of clusters for the data densities of 5% to 55% separately, as shown in Figure 6.5 and Figure 6.6.

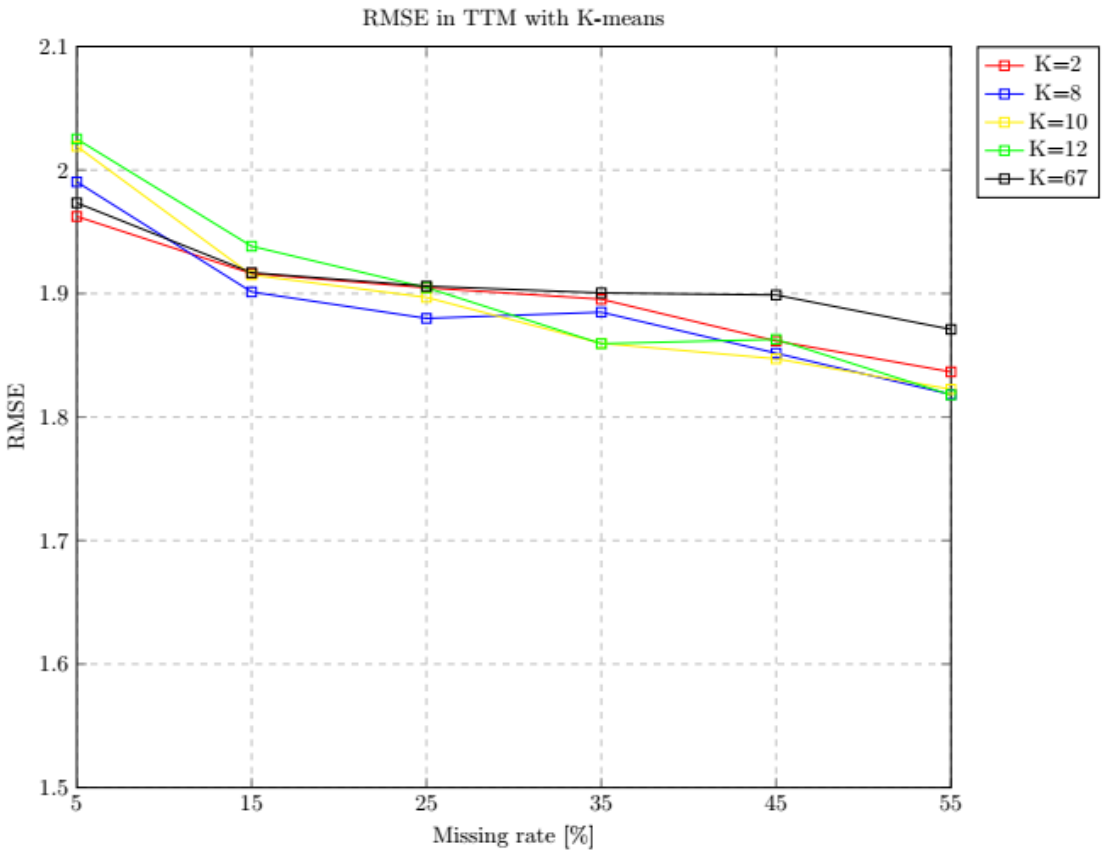


Figure 6.5 Impact of different number of clusters in TTM with K-means clustering

Figure 6.5 depicts the *RMSE* values concerning the number of clusters for the data density range (5%, 55%) in TTM with the K-means method.

For the TTM with the K-means method, with the same number of clusters, the value of *RMSE* decreases as the data density increases, which means that the prediction accuracy increases. On the other hand, when having the same data density, the *RMSE* value decreases with the increase of the number of clusters, but there is a difference: starting from about 12% to 55% of the data density, the *RMSE* value for 8 clusters is the lowest, while the *RMSE* value for 2 clusters is the lowest value when the data density is less than 12%.

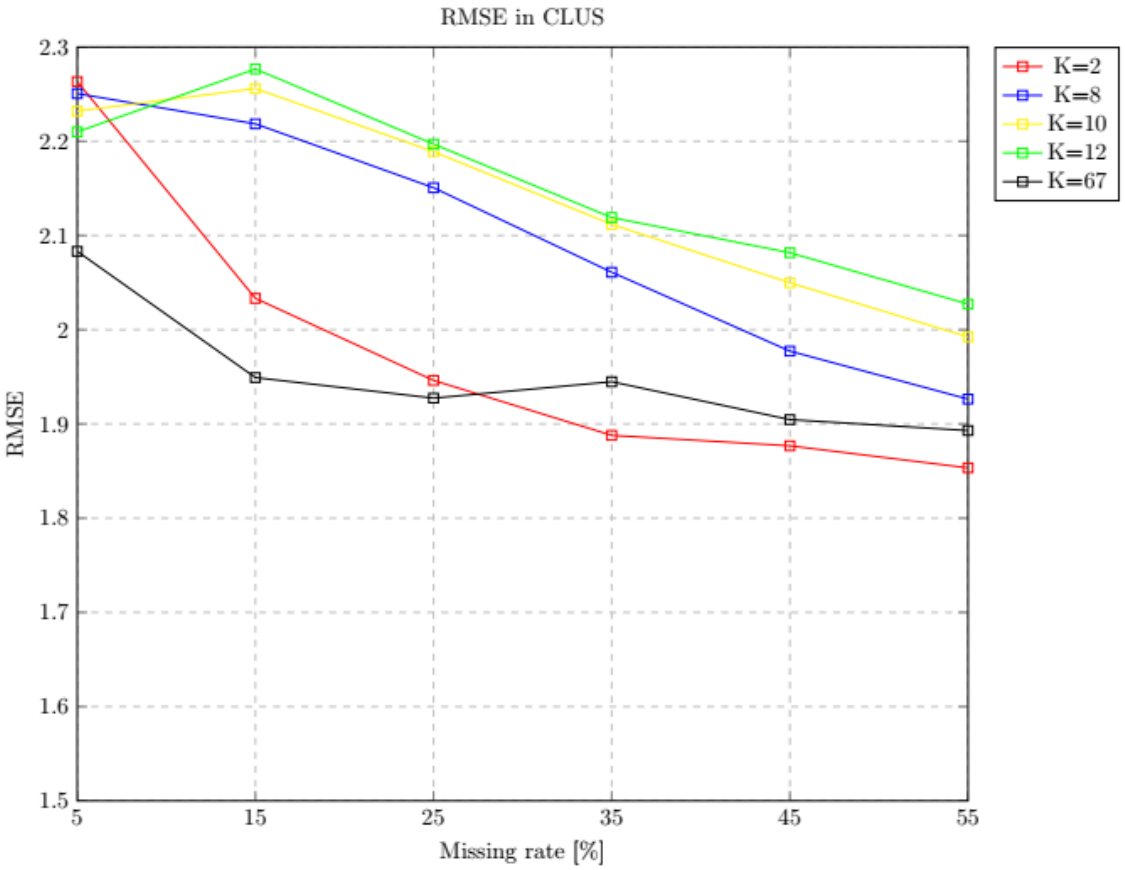


Figure 6.6 Impact of different number of clusters in CLUS

For the CLUS method, Figure 6.6 depicts the relationship between the different *RMSE* values and the number of clusters. Compared to the TTM with the K-means method, the distribution of the *RMSE* value curves is scattered. From the start at 5% data density, with the number of clusters increases, the value of the *RMSE* increases for 8 clusters and 10 clusters until reaching the highest value at the data density of 15%. After 15% data density, the value of the *RMSE* increases for 8 clusters and 10 clusters. In contrast, the other three curves for 10, 12, and 67 clusters keep downward.

Overall, the values of the *RMSE* for 2 clusters and 67 clusters are lower than the other curves. The *RMSE* value for 67 clusters is the lowest when the data density is less than 27%. When the data density is greater than 27%, the *RMSE* value for 2 clusters is the lowest. As shown, we summarize the following,

- Compared with the TTM using the K-means method, the distribution of the RMSE value curve of CLUS is a little scattered.

- With the value of K increases, the prediction accuracy increases.
- When having the same missing rate, the higher the K value, the higher the prediction accuracy.

A higher number of clusters means less clustering, which improves the prediction accuracy. The reason is that clustering reduces redundancy, and the tensor decomposition method enhances the prediction performance.

6.5.5 Impact of distance metrics

Most clustering algorithms calculate data objects' similarity using distance metrics to group related data objects [Silic, Delac, & Srbljic, 2014]. Since each algorithm uses a different distance metric, the similarity between two data objects can be calculated differently using different algorithms, so choosing the appropriate distance metric plays a crucial role in any clustering algorithm. In the traditional K-means algorithm, the Euclidean distance is used to measure the similarity between data objects, and the Euclidean distance assumes that all attribute values of data objects are equally crucial by default in real life. This indiscriminate treatment of attribute importance will likely lead to distance distortion of data objects in Euclidean space: although two points in the space are close on essential attributes. However, due to distance amplification by other irrelevant attributes, these two points are likely to be measured as farthest in Euclidean space [Li & Man, 2013].

In particular, for high-dimensional data, the cosine similarity between two data points $x(x_1, x_2, \dots, x_n)\ and\ y(y_1, y_2, \dots, y_n)$ sometimes is better in clustering than the Euclidean distance in such a spherical space [Liu, Hu, Ge, & Xiong, 2012]. The angle between two vectors indicates the cosine similarity,

$$\cos(\mathrm{x}, \mathrm{y}) = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}} \tag{6.6}$$

where $x_i, y_i\ (i = 1,2, \dots, n)$ are matrix variants in the cluster assignment.

We compare the cosine and the Euclidean distance formula. Figures 6.7 to 6.10 illustrate the evaluation results in the different data densities separately while applying the two distance formulas.

In Figure 6.7, it is evident that the cosine-based method provides improved prediction accuracy than the Euclidean-based method at all numbers of clusters in 5% density. With

the increase of clusters from 70 to 140, the *MAE* value increases, and the prediction accuracy worsens. The worst *MAE* value of the Euclidean-based method appears at 140 clusters in Figure 6.7. As shown in Figure 6.8, for 5% density, the curve still rises slightly after 70 clusters. For 20% and 30% density, the *MAE* values show that the accuracy performance curve drop-down when crossing 70 clusters in Figure 6.9 and Figure 6.10 separately.

As shown, we summarize the following,

The *MAE* values obtained by the two methods are not significantly different in 10%, 20%, and 30% density separately.

We summarize as follows.

- Compared with the Euclidean method, the cosine method improves the correct prediction accuracy rate when having the same K value.

  The reason is that the vector data are more sensitive to angle calculation and less sensitive to absolute length calculation. The distance between members can be measured more accurately by angle calculation.

- The prediction accuracy rate obtained by both methods is higher when the K value is smaller.

  If the category size of users or services sample data may not be high, the lower cluster number fits exactly the sample categories. We will evaluate different datasets to analyze this issue in future studies.

- The prediction accuracy rate obtained by the two methods is almost the same in K=5,10, and 70 separately.

  We note that these three values are almost the same as the initial set of K values. This problem shows that the choice of the initial cluster number is important in the clustering method, and it determines the performance of the clustering.

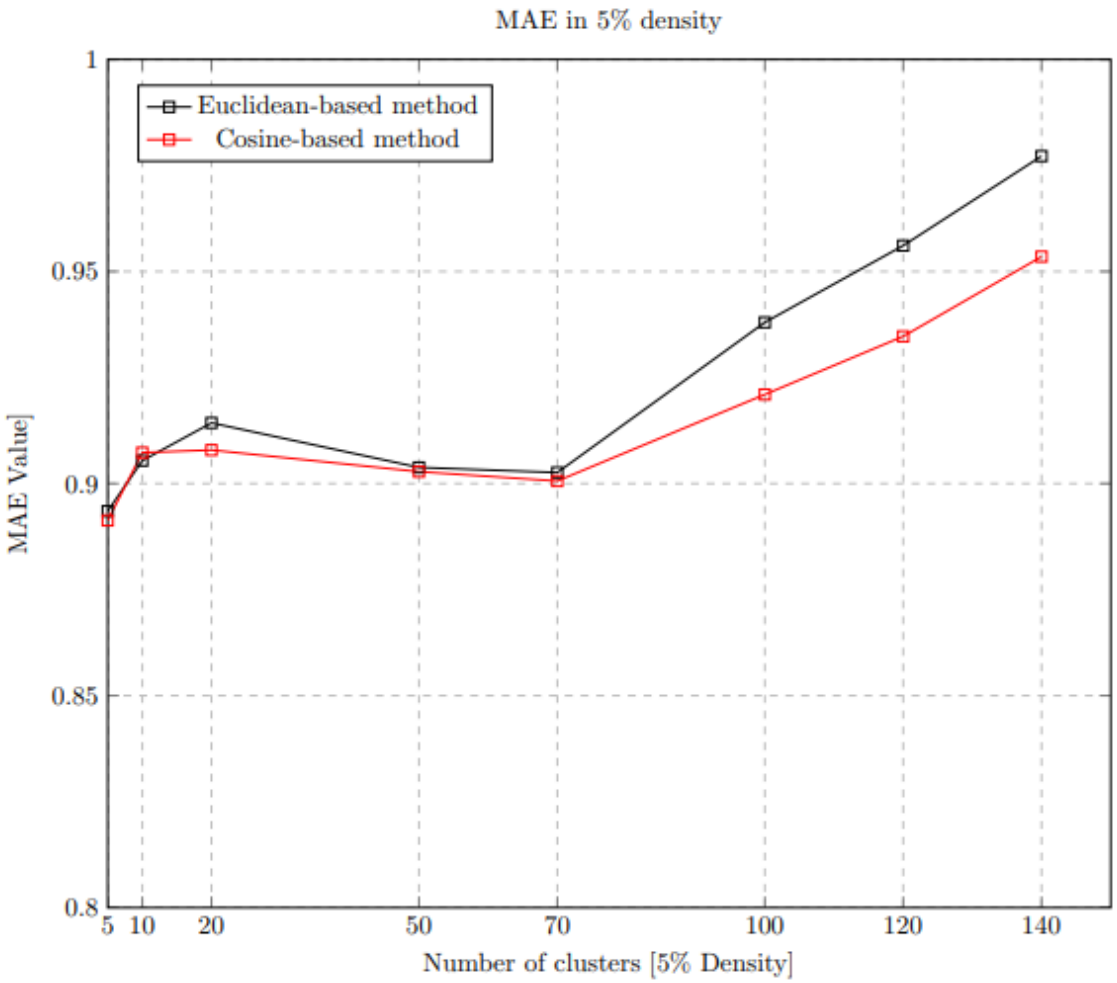


Figure 6.7 Impact of the distance metrics on *MAE* in 5% density

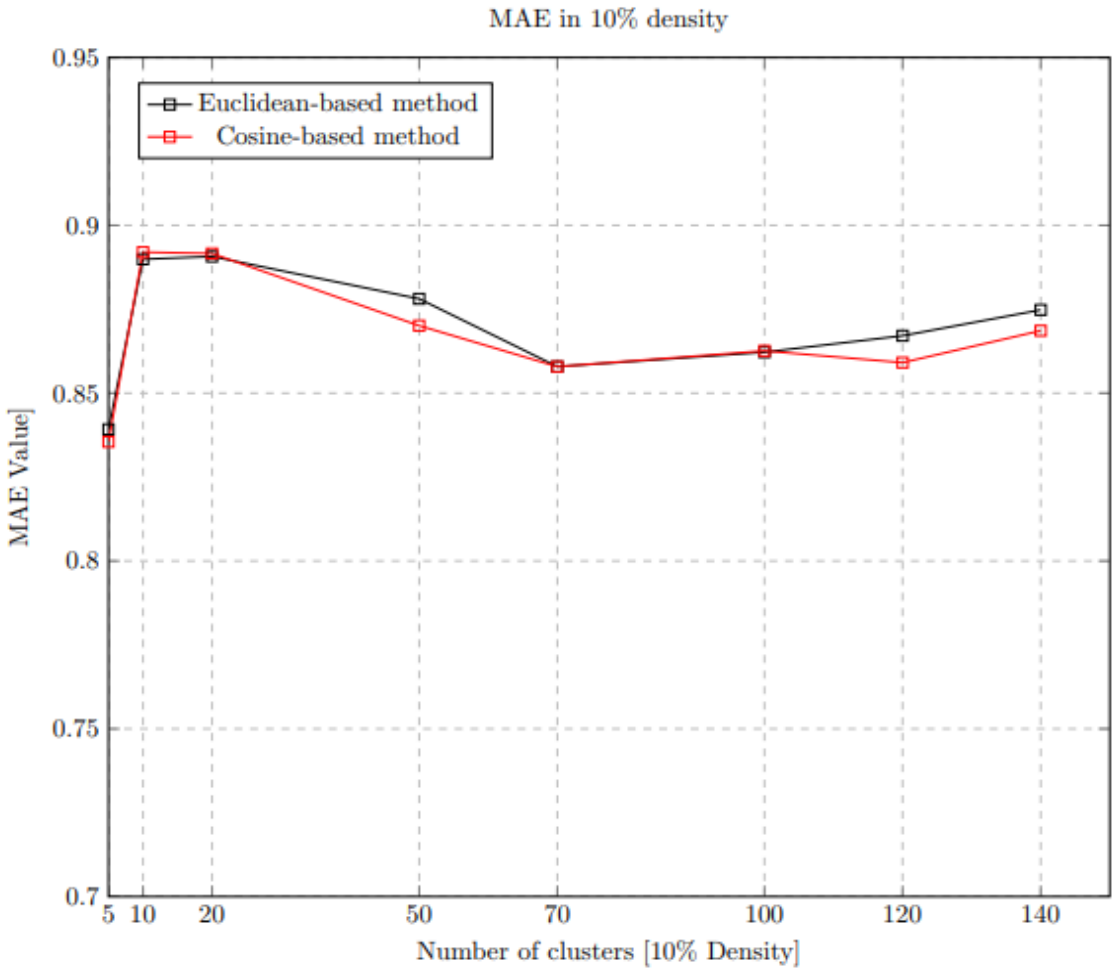


Figure 6.8 Impact of the distance metrics on *MAE* in 10% density

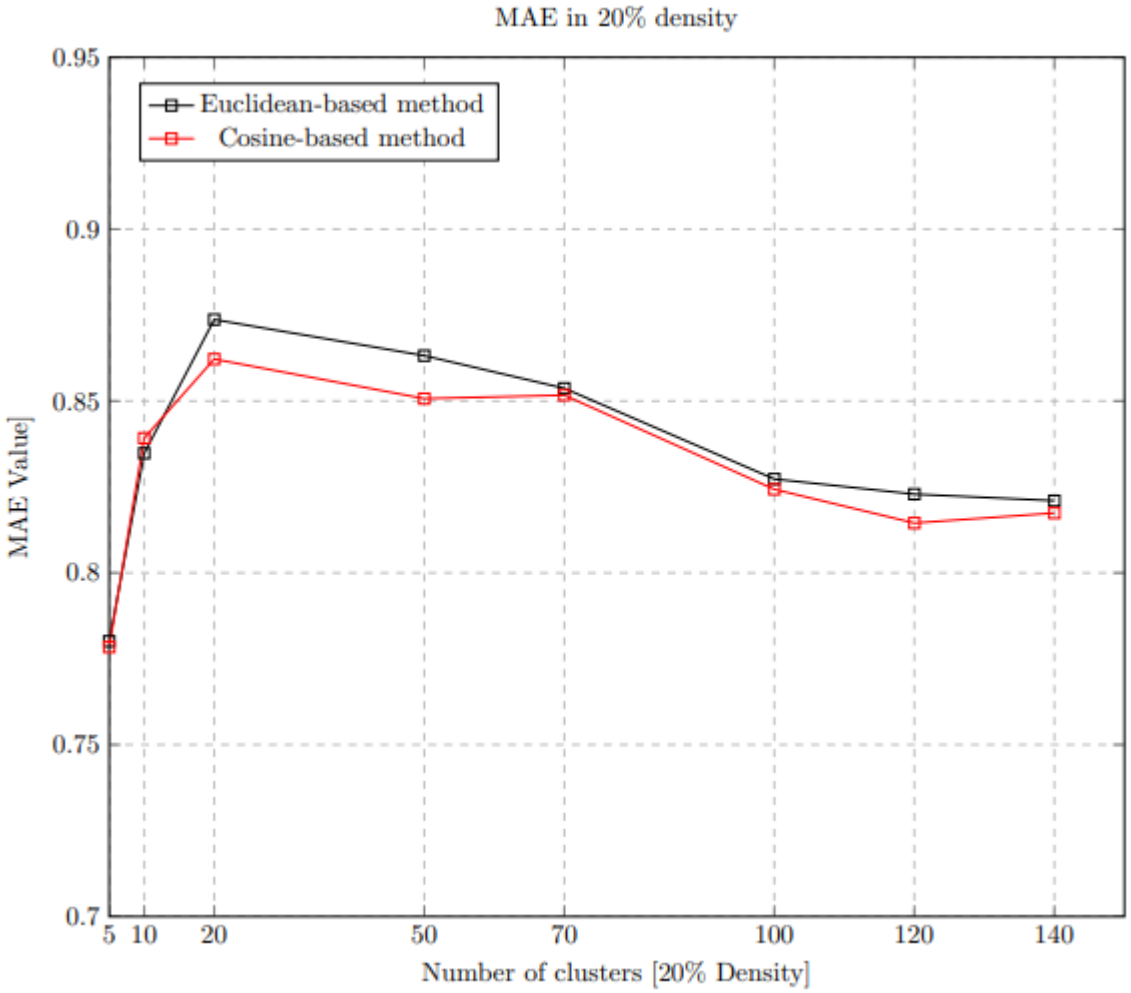


Figure 6.9 Impact of the distance metrics on *MAE* in 20% density

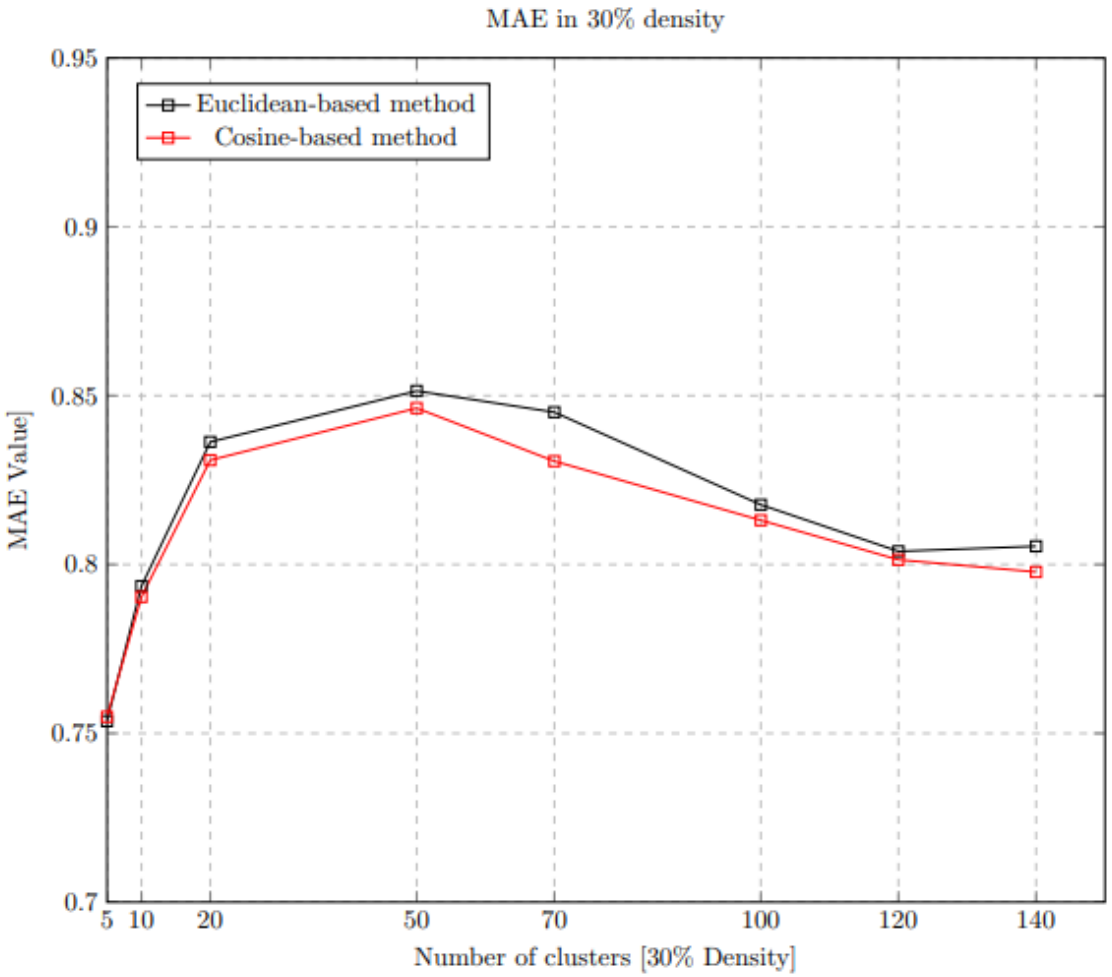


Figure 6.10 Impact of the distance metrics on *MAE* in 30% density

6.6 Summary

We propose a two-step strategy based on the K-means algorithm and TTM to deal with the initial unorganized data. The TTM with the K-means method removes the empty parts of the model, and the approximate tensor is obtained by reconstruction to generate the corresponding recommended values.

A series of experiments have been conducted to evaluate the effectiveness of the recommendation system combining the clustering technique and tensor decomposition algorithm. We discuss the issues such as selecting initial K values, computational performance, and distance calculation. The TTM with the K-means method improves the

prediction accuracy in a recommendation with a lower *RMSE* value than the K-mean clustering algorithm.

Chapter 7

Conclusions and Future Directions

QoS attribute prediction has become a popular research topic in service recommendation. Most of the existing prediction methods are based on a large sample and have been successfully applied to the web service recommendation system. These web recommendation methods are not satisfactory when the user-service-time web recommendation data is sparse. The reason is that the samples in the initial recommendation dataset are only the corresponding values of the existing web services used by the existing users. The QoS attribute prediction is performed based on available historical data. When facing an increase of new users or new web services, no new invocation records are recorded due to the limitations of some conditions. Although the traditional tensor decomposition model is a powerful prediction tool in the web service recommendation system, it also does not avoid the shortcoming of low accuracy prediction rate due to small sample data. It is challenged to construct a suitable method to address the small sample problem based on the traditional tensor decomposition model. Thus, we addressed these issues and proposed a modified tensor decomposition method for QoS attribute prediction to improve the recommendation accuracy.

7.1 Summary

This thesis conducts a new tensor decomposition method to enhance QoS attribute prediction performance.

(1) A tensor-based modeling for web service recommendation is proposed to meet the added sample data and preprocess demand. Data analysis reveals the observation that the number of samples of user-service-time data is insufficient, and the preprocessing of initial data affects the QoS recommendations. Based on these two observations, the methods add an enhanced sample scheme and a K-means clustering preprocessing compared to the regular tensor decomposition method to solve low prediction accuracy due to sample data lack. In the tensor decomposition stage, a new two regularization term is implemented to avoid overfitting. Analysis of experimental results on real datasets shows that the proposed method improves the quality of prediction.

(2) An improved tensor decomposition method, TTM, is proposed to address the shortcomings of traditional recommendation methods with low accuracy due to insufficient initial samples. The data analysis reveals the observation that constructing new samples from existing samples can compensate for the insufficient samples in the user-service-time data. Based on this observation, the feature factor matrices are designed based on the current feature sample data to compensate for the missing items in the factor matrix of different time slices. Then, it is applied to the tensor decomposition as a way to improve the prediction rate. Finally, the experiments are conducted on the web service WSDream and the traffic prediction datasets. Experimental results validate the effectiveness of the proposed TTM recommendation method.

(3) A regularization term incorporating two regular models is proposed to support a web service recommendation TTM method. The different roles of these two different regularization models are analyzed to solve the overfitting problem of web service recommendation systems. A modified regularization term is designed and applied to the TTM method by integrating the lasso and ridge models. Experimental validation is conducted on the web service WSDream dataset to discuss and evaluate the weights of the different regularization models. Experimental results show that the modified regularization term can help increase the correct rate of predicting QoS attributes and better support the web service recommendation TTM method.

(4) A two-step strategy approach is proposed, which includes clustering and TTM methods. Data analysis revealed the observation that the preprocessing of the initial unorganized data may impact the service recommendation performance. Based on this observation, first, a preprocessing process based on K-means clustering was designed to cluster the initial data. In the second step, the clustered data objects are used as input and applied to the TTM method to complete the QoS attribute prediction. Experimental results on the relevant dataset show the evaluation between our methods and the clustering method.

7.2 Future Directions

Research on the theory and technology of QoS attribute prediction-based is constantly evolving, and new requirements are emerging as web services are widely used and application scenarios continue to be extended. Therefore, the following issues in web service recommendations need to continue to be studied in depth.

(1) *Real-time update technology for web service recommendation methods.*

Whether it is an easy-to-understand collaborative filtering recommendation method or a tensor decomposition model with better recommendation results, the computational complexity of these methods is high. In the application scenario of real-time update of QoS attribute, to capture the user's interest and web service performance changes in the latest context, repeated training of prediction methods should be avoided, and incremental updates based on the current training models should be performed. Therefore, the design of prediction models supporting update mechanisms has significant theoretical research and application value.

(2) *Research on web service recommendation algorithms in different network environments.*

In the actual application scenarios of web services, the network environment in which the services are located may differ, such as the difference between fixed and mobile data network environments. This difference will lead to a significant difference in QoS attribute prediction performance among users. Therefore, studying the service recommendation methods under various network environment scenarios is essential, especially in mobile internet environments.

(3) *Investigate efficient and parallelized tensor decomposition algorithms.*

Tensor decomposition helps estimate and recover missing data from a sample dataset; however, the decomposition is the most time-costing in the whole process, especially for higher-dimensional data, even four-dimensional and five-dimensional data, the algorithm complexity can be a very high degree. The TTM method proposed in this thesis has a high accuracy prediction rate, but this follows at the high complexity cost. Thus, it is practical to consider algorithmic parallelization to investigate decomposition algorithms to reduce processing time effectively.

(4) *Higher dimensions data*

For large-scale recommender systems, millions of users send tens of thousands of service requests to the system at each moment. The huge QoS data brings us complete service information and the challenge of higher dimensions data analysis. In the past decades, deep learning techniques have been widely studied and applied in recommender systems. However, less research has been done on QoS prediction, which deserves future research. The unique challenges need to be addressed: processing large amounts of QoS data brings a significant challenge to the efficiency of traditional QoS prediction techniques. It is expected that additional research efforts will address this unique challenge and further advance current QoS prediction techniques.

Bibliography

BAĞIRÖZ, B., Güzel, M., YAVANOĞLU, U., & Özdemir, S. (2019). QoS Prediction Methods in IoT A Survey. Paper presented at the 2019 *IEEE International Conference on Big Data* (Big Data).

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6), 717-772.

Carroll, J. D., Pruzansky, S., & Kruskal, J. B. (1980). CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1), 3-24.

Chai, S., Feng, W., & Hassanein, H. S. (2016). Tensor decomposition-based web service QoS prediction. *Journal of Coupled Systems and Multiscale Dynamics*, 4(2), 113-118.

Chandrasekaran, V., Recht, B., Parrilo, P. A., & Willsky, A. S. (2012). The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6), 805-849.

Cheng, T., Wen, J., Xiong, Q., Zeng, J., Zhou, W., & Cai, X. (2019). Personalized Web service recommendation based on QoS prediction and hierarchical tensor decomposition. *IEEE Access*, 7, 62221-62230.

Cichocki, A., Zdunek, R., Phan, A. H., & Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation.* John Wiley & Sons.

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4), 1324-1342.

Dubey, A., & Choubey, A. (2017). A systematic review on k-means clustering techniques. *International Journal of Scientific Research Engineering & Technology* (IJSRET, ISSN 2278–0882), 6(6).

Fan, X., Hu, Y., Zhang, R., Chen, W., & Brézillon, P. (2015). Modeling temporal effectiveness for context-aware web services recommendation. Paper presented at the 2015 *IEEE International Conference on Web Services*.

Ghafouri, S. H., Hashemi, S. M., & Hung, P. C. (2020). A survey on Web service QoS prediction methods. *IEEE Transactions on Services Computing*.

Middleware Systems Research Group. (2020). One-ITS Toronto Traffic Dataset. Retrieved from http://msrg.org/datasets/traffic.

Hao, L., Liang, S., Ye, J., & Xu, Z. (2018). TensorD: A tensor decomposition library in TensorFlow. *Neurocomputing*, 318, 196-200.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16, 1- 84. (University Microfilms, Ann Arbor, Michigan, No. 10,085).

Hartigan, J. A. (1975). Clustering algorithms: John Wiley & Sons, Inc.

Hasnain, Muhammad; Pasha, Muhammad Fermi; Ghani, Imran; Mehboob, Bilal; Imran, Muhammad; Ali, Aitizaz. (2020). Benchmark Dataset Selection of Web Services Technologies: A Factor Analysis. *IEEE Access*, vol. 8, pp. 53649-53665.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1), 1-12.

He, P., Zhu, J., Zheng, Z., Xu, J., & Lyu, M. R. (2014). Location-based hierarchical matrix factorization for web service recommendation. Paper presented at the 2014 *IEEE International Conference on Web Services*.

Hillar, C. J., & Lim, L.-H. (2013). Most tensor problems are NP-hard. *Journal of the ACM*, 60(6), 1-39.

Jain, Prateek, Oh, Sewoong. (2014). Provable tensor factorization with missing data. *Advances in Neural Information Processing Systems*, pp. 1431-1439.

Ji, Y., Wang, Q., Li, X., & Liu, J. (2019). A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7, 162950-162990.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. Paper presented at the *Proceedings of the 4th ACM Conference on Recommender Systems*.

Kiers, H. A. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3), 105-122.

Kolda, T. G. (2006). *Multilinear operators for higher-order decompositions*. (No. SAND2006-2081). Sandia National Laboratories.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *Journal of SIAM Review*, 51(3), 455-500.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30-37.

Lacroix, T., Usunier, N., & Obozinski, G. (2018). Canonical tensor decomposition for knowledge base completion. Paper presented at the *International Conference on Machine Learning*.

Li, S., & Man, Z. (2013). K-means clustering algorithm based on adaptive feature weighted. *Computer Technology and Development*, 23(6), 98-105.

Liu, C., Hu, T., Ge, Y., & Xiong, H. (2012). Which distance metric is right: An evolutionary k-means view. Paper presented at the *Proceedings of the 2012 SIAM International Conference on Data Mining*.

Liu, X., & Fulia, I. (2015). Incorporating user, topic, and service-related latent factors into web service recommendation. Paper presented at the 2015 *IEEE International Conference on Web Services*.

Ma, Y., Wang, S., Yang, F., & Chang, R. N. (2015). Predicting QoS values via multi-dimensional QoS data for Web service recommendations. Paper presented at the 2015 *IEEE International Conference on Web Services*.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Paper presented at the *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1967 (Vol. 1, pp. 281-297). University of California Press.

Mnih, A., & Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, 20, 1257-1264.

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. Paper presented at the *International Conference on Machine Learning*.

O'Sullivan, J., Edmond, D., & ter Hofstede, A. (2002). What is in a Service? *Distributed and Parallel Databases*, 12(2), 117-133.

Ogutu, J. O., Schulz-Streeck, T., & Piepho, H.-P. (2012). Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. In *BMC Proceedings* (Vol. 6, pp. 1-6). BioMed Central.

Paatero, P. (1997). A weighted non-negative least squares algorithm for three-way 'PARAFAC'factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2), 223-242.

Pandharbale, P. B., Mohanty, Sachi Nandan, & Jagadev, Alok Kumar. (2021). Recent web service recommendation methods: A review. *Materials Today: Proceedings*.

Programmableweb. (2021). API Directory. Retrieved from https://www.programmableweb.com/category/all/apis

Ragnarsson, Stefan, & Van Loan, Charles F. (2012). Block tensor unfoldings. *SIAM Journal on Matrix Analysis and Applications*, 33(1), 149-169.

Rajih, M., Comon, P., & Harshman, R. A. (2008). Enhanced line search: A novel method to accelerate PARAFAC. *SIAM Journal on Matrix Analysis and Applications*, 30(3), 1128-1147.

Rao, J., & Su, X. (2005). A Survey of Automated Web Service Composition Methods, In *International Workshop on Semantic Web Services and Web Process Composition* (pp. 43-54). Springer, Berlin, Heidelberg.

Rezaee, M. R., Lelieveldt, B. P., & Reiber, J. H. (1998). A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3-4), 237-246.

Ruffinelli, D., Broscheit, S., & Gemulla, R. (2019). You can teach an old dog new trick! on training knowledge graph embeddings. Paper presented at the *International Conference on Learning Representations*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Paper presented at the *Proceedings of the 10th International Conference on World Wide Web*.

Shang, W., Wang, K., & Huang, J. (2018). An Improved Tensor Decomposition Model for Recommendation System. *International Journal of Performability Engineering*, 14(9), 2116.

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., & Mei, H. (2007). Personalized QoS prediction for web services via collaborative filtering. Paper presented at the *IEEE International Conference on Web Services*.

Shashua, A., & Levin, A. (2001). Linear image coding for regression and classification using the tensor-rank principle. Paper presented at the Proceedings of the 2001 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Shi, M., Li, D., & Zhang, J. Q. (2019). Matrix Polynomial Predictive Model: A New Approach to Accelerating the PARAFAC Decomposition. *IEEE Access*, 7, 91872-91884.

Shi, Q., Cheung, Y.-M., Zhao, Q., & Lu, H. (2018). Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6), 1803-1817.

Signoretto, M., De Lathauwer, L., & Suykens, J. A. (2010). Nuclear norms for tensors and their use for convex multilinear estimation. *Journal of Linear Algebra and Its Applications*.

Silic, M., Delac, G., & Srbljic, S. (2014). Prediction of atomic web services reliability for QoS-aware recommendation. *IEEE Transactions on Services Computing*, 8(3), 425-438.

Song, Qingquan, Ge, Hancheng, Caverlee, James, & Hu, Xia. (2019). Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1), 1-48.

Stork, D. G., Duda, R. O., Hart, P. E., & Stork, D. (2001). *Pattern classification*. A Wiley-Interscience Publication.

Symeonidis, P. (2015). ClustHOSVD: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(9), 1240-1251.

Tao, D., Li, X., Hu, W., Maybank, S., & Wu, X. (2005). Supervised tensor learning. Paper presented at the *Fifth IEEE International Conference on Data Mining*.

Thomas, L., & Immanuel, A. (2017). Web Service Composition: A Survey on the Various Methods used for Web Service Composition. *International Journal of Advanced Research in Computer Science*, 8(3).

Wolf, L., Jhuang, H., & Hazan, T. (2007). Modeling appearances with low-rank SVM. Paper presented at the *2007 IEEE Conference on Computer Vision and Pattern Recognition*.

Wu, H., Zhang, Z., Luo, J., Yue, K., & Hsu, C. H. (2018). Multiple attributes QoS prediction via deep neural model with contexts. *IEEE Transactions on Services Computing*.

Xiong, W., Wu, Z., Li, B., & Gu, Q. (2017, June). A learning approach to QoS prediction via multi-dimensional context. In *2017 IEEE International Conference on Web Services (ICWS)*, IEEE, 164-171.

Yu, C., & Huang, L. (2017). CluCF: a clustering CF algorithm to address data sparsity problem. *Service-Oriented Computing and Applications*, 11(1), 33-45.

Zhang, J., Han, Y., & Jiang, J. (2016). Tucker decomposition-based tensor learning for human action recognition. *Multimedia Systems*, 22(3), 343-353.

Zhang, W., Sun, H., Liu, X., & Guo, X. (2014a). Incorporating invocation time in predicting web service QoS via triadic factorization. Paper presented at the *2014 IEEE International Conference on Web Services*.

Zhang, W., Sun, H., Liu, X., & Guo, X. (2014b). Temporal QoS-aware web service recommendation via non-negative tensor factorization. Paper presented at the *International Conference on World Wide Web*.

Zhang, Y., Zheng, Z., & Lyu, M. R. (2011). WSPred: A time-aware personalized QoS prediction framework for Web services. Paper presented at the *2011 IEEE 22nd International Symposium on Software Reliability Engineering*.

Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2009). Wsrec: A collaborative filtering-based web service recommender system. Paper presented at the 2*009 IEEE International Conference on Web Services*.

Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2010). QoS-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing*, 4(2), 140-152.

Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2012). Collaborative web service QoS prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 6(3), 289-299.

Zhou, Q., Wu, H., Yue, K., & Hsu, C. H. (2019). Spatio-temporal context-aware collaborative QoS prediction. *Future Generation Computer Systems*,100, 46-57.

Appendix A

Matrix factorization

Given a dataset of m-dimensional data vectors, the vectors are placed in the columns of a $m \times n$ matrix $\boldsymbol{V} \in \mathbb{R}^{m \times n}$, n is the number of examples of the dataset.

The matrix $\boldsymbol{V}$ is approximately equal to an approximation data matrix $\overline{\boldsymbol{V}} \in \mathbb{R}^{m \times n}$. The approximation data matrix $\overline{\boldsymbol{V}}$ is factorized into a matrix $W \in \mathbb{R}^{m \times r}$ and a matrix $H \in \mathbb{R}^{r \times n}$ as shown as follows,

$$\boldsymbol{V} \approx \overline{\boldsymbol{V}} = WH$$

Set the data vector v_r is the corresponding columns of $\overline{\boldsymbol{V}} = \left[v_{1,}\, v_{2,} \ldots, v_{r,}\right] \in \mathbb{R}^{m \times n}$, each data vector $v_r \approx Wh$, where h is the corresponding columns of H, and the r usually is smaller than m or n.

To find an approximate factorization $\overline{\boldsymbol{V}} = WH$, the matrix factorization (MF) method seeks a decomposition of the data matrix $\overline{\boldsymbol{V}}$ with matrices W and H. We consider the objective function $E(W, H)$ which is given by

$$E(W, H) = ||\boldsymbol{V} - \overline{\boldsymbol{V}}||^2 = ||\boldsymbol{V} - WH||^2 = \sum_{i,j} \left(\boldsymbol{V}_{ij} - [WH]_{ij}\right)^2$$

MF involves the following optimization problem:

$$\begin{aligned} &\arg\min_{W,H} E(W, H) \\ &= \arg\min_{W,H} \left|\left|\boldsymbol{V} - WH\right|\right|^2 \\ &= \arg\min_{W,H} \sum_{i,j} \left(\boldsymbol{V}_{ij} - [WH]_{ij}\right)^2 \end{aligned}$$

Appendix B

Regular tensor decomposition

The tensor decomposition method applies in the high dimensional dataset from the matrix factorization in Kolda's literature. We focus on a 3-way tensor with rank-one $\boldsymbol{\mathcal{X}}$, and an approximation tensor with rank-one $\bar{\boldsymbol{\mathcal{X}}} = [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]$ where the factor matrix $U^{(1)}, U^{(2)}, and\ U^{(3)}$, $\mathrm{X}_{(1)}$ is the *1st* frontal slice of the tensor $\mathcal{X}$, $\mathrm{X}_{(2)}$ is the *2nd* frontal slice of the tensor $\mathcal{X}$, and $\mathrm{X}_{(3)}$ is the *3rd* frontal slice of the tensor $\mathcal{X}$. The frontal slices are generated as the following equations,

$$\mathrm{X}_{(1)} = U^{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T$$

$$\mathrm{X}_{(2)} = U^{(2)}\left(U^{(1)} \odot U^{(3)}\right)^T$$

$$\mathrm{X}_{(3)} = U^{(3)}\left(U^{(2)} \odot U^{(1)}\right)^T$$

where the symbol $\odot$ denotes the Khatri-Rao product, T denotes the matrix transpose.

Given a loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$, the loss value is calculated between two tensors $\boldsymbol{\mathcal{X}}$ and $\bar{\boldsymbol{\mathcal{X}}}$. where $||\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}||^2$ is the L2 norm. When minimizing the function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ and it becomes zero, a decomposition of the tensor $\boldsymbol{\mathcal{X}}$ is completed, and the components are generated.

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = ||\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}||^2 = ||\boldsymbol{\mathcal{X}} - [\![\boldsymbol{U^{(1)}}, \boldsymbol{U^{(2)}}, \boldsymbol{U^{(3)}}]\!]||^2.$$

Following the Alternating Least Squares (ALS) algorithm, we obtain the regular factor matrix $\mathrm{U}^{(\mathrm{n})}$ $n = 1,2, and\ 3$ as follows equations,

$$U^{(1)} = \mathbf{X}_{(1)}\left(U^{(2)} \odot U^{(3)}\right)^T \Rightarrow U^{(1)} = \mathbf{X}_{(1)}\left[U^{(2)} \odot U^{(3)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(3)}\right)^T U^{(3)}\right]^+$$

$$U^{(2)} = \mathbf{X}_{(2)}\left(U^{(1)} \odot U^{(3)}\right)^T \Rightarrow U^{(2)} = \mathbf{X}_{(2)}\left[U^{(1)} \odot U^{(3)}\right]\left[\left(U^{(1)}\right)^T U^{(1)} * \left(U^{(3)}\right)^T U^{(3)}\right]^+$$

$$U^{(3)} = \mathbf{X}_{(3)}\left(U^{(2)} \odot U^{(1)}\right)^T \Rightarrow U^{(3)} = \mathbf{X}_{(3)}\left[U^{(2)} \odot U^{(1)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(1)}\right)^T U^{(1)}\right]^+$$

where the symbol $\odot$ denotes the Khatri-Rao product, T denotes the matrix transpose, and the symbol $+$ denotes the pseudoinverse, and * denotes the Hadamard product.

Substituting the above equations, the function $f(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ is written as follows,

$$f(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \left|\left|\boldsymbol{\mathcal{X}} - \bar{\boldsymbol{\mathcal{X}}}\right|\right|^2 = \left|\left|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{U^{(1)}}, \boldsymbol{U^{(2)}}, \boldsymbol{U^{(3)}}]\!]\right|\right|^2$$

$$= ||\boldsymbol{\mathcal{X}} - (\mathbf{X}_{(1)}\left[U^{(2)} \odot U^{(3)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(3)}\right)^T U^{(3)}\right]^+)\circ(\mathbf{X}_{(2)}\left[U^{(1)}\right.$$
$$\left.\odot U^{(3)}\right]\left[\left(U^{(1)}\right)^T U^{(1)} * \left(U^{(3)}\right)^T U^{(3)}\right]^+)\circ(\mathbf{X}_{(3)}\left[U^{(2)}\right.$$
$$\left.\odot U^{(1)}\right]\left[\left(U^{(2)}\right)^T U^{(2)} * \left(U^{(1)}\right)^T U^{(1)}\right]^+)||^2$$

where the symbol $\odot$ denotes the Khatri-Rao product, T denotes the matrix transpose, and the symbol $+$ denotes the pseudoinverse, and * denotes the Hadamard product.

There is another representation of this algorithm. Set the $u^{(1)}$ is the corresponding vector of the factor matrix $U^{(1)} = \left[u^{(1)}\right]$, $u^{(2)}$ is the corresponding vector of the factor matrix $U^{(2)} = \left[u^{(2)}\right]$, and $u^{(3)}$ is the corresponding vector of the factor matrix $U^{(3)} = \left[u^{(3)}\right]$. The *3*-way original tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is a result of the outer product of *three* vectors set as follows,

$$\mathcal{X} \approx \bar{\mathcal{X}} = \left[u^{(1)}\right] \circ \left[u^{(2)}\right] \circ \left[u^{(3)}\right]$$

where the symbol "$\circ$" represents the vector outer product, $[\bullet]$ denotes the vector set.

We rewrite the loss function $\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}})$ as the outer product of three vectors,

$$\ell(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = \left\|\mathcal{X} - \left[u^{(1)}\right] \circ \left[u^{(2)}\right] \circ \left[u^{(3)}\right]\right\|^2 .$$

The regular tensor decomposition method involves the following optimization problem:

$$\underset{u^{(1)},u^{(2)},u^{(3)}}{\arg\min} \left\|\mathcal{X} - \left[u^{(1)}\right] \circ \left[u^{(2)}\right] \circ \left[u^{(3)}\right]\right\|^2.$$