Graph Regularized Tensor Decomposition for Recommender Systems

Rohan Chandrashekar

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Graph Regularized Tensor Decomposition for Recommender Systems

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

by

Rohan Chandrashekar

November 21, 2022

Thesis supervisors:

Dr.ir. K. Batselier, Faculty 3mE, DCSC, TU Delft
Dr. E. Isufi, Faculty EEMCS, TU Delft
E.M. Memmel, Faculty 3mE, DCSC, TU Delft

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Humans make decisions when presented with choices based on influences. The Internet today presents people with abundant choices to choose from. Recommending choices with an emphasis on people's preferences has become increasingly sought. Grundy (1979), the first computer librarian Recommender System (RS), provided users with book recommendations. Growing volumes of user data in the '90s saw increased usage of commercially available RS for e-commerce, music, movies, books, and social networking services. Due to their effectiveness in providing recommendations, Collaborative Filtering (CF) algorithms are predominantly used to build these RS. However, traditional CF algorithms adopting Matrix Factorization (MF) and Nearest Neighbor (NN) methods suffer from handling sparse data or model scalability. With exponentially increasing sparse data, building scalable and accurate RS models is of focus.

This thesis uses tensors and graphs to represent available data. Emphasis is given to capturing higher-order interactions present between the data. The use of tensors is motivated as matrices cannot capture data with higher-order relations, such as variation of user ratings to items with time. The transition to using tensors has been promising with the development of efficient tensor decomposition methods and powerful machines. Graphs can capture the correlation between different entities, providing additional information intrinsic to the underlying graph structure. A Graph Regularized CANDECOMP/PARAFAC (GRCP) tensor decomposition model framework is proposed in this thesis. The thesis highlights how to graph Laplacian regularizers (GLRs) benefit CP tensor decomposition methods to build RS. The model framework is evaluated with the MovieLens data set. The model records lesser Normalized Mean Squared Error (NMSE) values than those reported in the literature. The combination of varied data sources notably aids in overcoming the drawbacks of current RS models, offering scalability with computational efficiency in linear time.

Table of Contents

Acknowledgements **ix**

1 Introduction **1**

1-1 Recommender System Modelling . . . 3

1-1-1 Benefits and Challenges . . . 4

1-2 Thesis Motivation and Research Aims . . . 5

1-3 Thesis Outline . . . 7

2 Tensors and Graphs: A Mathematical Background **8**

2-1 Motivation . . . 8

2-2 Tensor Theory . . . 8

2-2-1 Notations and Basics . . . 8

2-2-2 Tensor Operations . . . 10

2-2-3 CANDECOMP/PARAFAC (CP) Tensor Decomposition . . . 10

2-2-3-1 Alternating Least Squares (ALS) Algorithm . . . 11

2-3 Graph Theory . . . 12

2-3-1 Notations and Basics . . . 12

2-3-2 Graph Signals and Spectra . . . 13

2-3-3 Generating Similarity Measures: Graph Kernels . . . 14

2-3-4 Regularization with Graph Laplacian . . . 14

2-4 Discussions . . . 15

2-5 Conclusion . . . 16

3 Graph Regularized (GRCP) Tensor Decomposition **17**
3-1 Motivation . . . 17
3-2 Literature Review . . . 17
3-2-1 Tensors for RS . . . 17
3-2-2 Graphs for RS . . . 18
3-2-3 Tensors and Graphs for RS . . . 19
3-2-4 Discussions . . . 20
3-3 Graph Regularized CP (GRCP) Tensor Decomposition Model . . . 21
3-3-1 GRCP Alternating Least Squares (ALS) Algorithm . . . 23
3-3-2 Conjugate Gradient Solver . . . 23
3-3-3 Convergence Analysis . . . 24
3-3-4 Performance Evaluation Metrics . . . 27
3-4 GRCP for Recommender Systems . . . 28
3-5 Discussions . . . 28
3-6 Conclusion . . . 29

4 Model Validation using Synthetic Dataset **30**
4-1 Motivation . . . 30
4-2 Experiment Setup . . . 30
4-2-1 Algorithm Initialization . . . 31
4-3 Results and Discussions . . . 32
4-3-1 Influence of CP Rank . . . 32
4-3-2 Influence of Regularization . . . 33
4-3-3 Influence of Noise . . . 34
4-3-4 Performance Evaluation Metrics . . . 35
4-4 Conclusions . . . 38

5 Model Validation using MovieLens Dataset **39**
5-1 Motivation . . . 39
5-2 Experiment Setup . . . 39
5-2-1 Algorithm Initialization . . . 40
5-3 Results and Discussions . . . 41
5-3-1 Influence of CP Rank . . . 41
5-3-2 Influence of Regularization . . . 43
5-3-3 Performance Evaluation Metrics . . . 45
5-4 Scalability . . . 48
5-4-1 Experiment Setup . . . 49
5-4-2 Results and Discussions . . . 49
5-5 Conclusions . . . 50

6 Conclusions and Future Scope of Work **51**

A Graph Regularized (GRCP) Tensor Decomposition **54**
A-1 Graph Regularized CP Tensor Decomposition Model 54
A-2 Alternating Least Squares (ALS) algorithm 55
A-3 Convergence Analysis 55
A-3-1 Lipschitz Boundedness 55
A-3-2 KL Inequality for Real Analytic Functions 56

B Model Validation Using Synthetic Dataset **58**
B-1 Performance Evaluation Metrics 58
B-1-1 Algorithm Run Times 58

C Model Validation Using MovieLens Dataset **59**
C-1 Influence of CP Rank 59
C-2 Influence of Regularization 62
C-3 Performance Evaluation Metrics 64
C-3-1 Relative Reconstruction Error (RRE) and POF 64
C-4 Scalability 66

Bibliography **68**

Glossary **74**
List of Acronyms 74

List of Figures

1-1 Collaborative Filtering Techniques 3

2-1 CP Decomposition of a 3-way tensor into its factor approximations [1] . . . 12

2-2 Graph structures depicted with nodes as circles and edges connecting nodes as lines 13

4-1 Image baboon.jpg 31

4-2 Effect of varying CP Rank on Image Reconstruction with kNN = [1 1 1]. Top Row: $\lambda_l = 0$, Bottom Row: $\lambda_l = 0.001$, Left to Right: CP Rank $R =$ [20 40 60 80 100 120 140] 32

4-3 Effect of varying kNN members and regularization on Image Reconstruction with CP rank = 60. Top to Bottom: kNN members [1 1 1], [1 1 2] and [2 2 2], Left to Right: Laplacian Regularization $\lambda_l = [0, 0.1, 0.01, 0.001]$ . . . 34

4-4 Effect of varying with CP rank = 40, kNN = [1 1 1] and $\lambda_l = 0.001$. (a) SNR = 0.1 dB, (b) SNR = 0.5 dB, (c) SNR = 1 dB 35

4-5 Influence of NMSE with varying CP Rank, noise SNR (dB), and regularization λ_l 36

5-1 Influence of varying CP ranks on Training NMSE for $\rho = 0.6\%$ 42

5-2 Influence of varying CP ranks on Training NMSE for $\rho = 2.5\%$ 43

5-3 Influence of varying Laplacian Regularization λ_l on Training NMSE for $\rho = 0.6\%$ 44

5-4 Influence of varying Laplacian Regularization λ_l on Training NMSE for $\rho = 2.5\%$ 44

5-5 Influence of varying hyperparameters on Training NMSE for $\rho = 0.6\%$ . . . 47

5-6 Influence of varying hyperparameters on Training NMSE for $\rho = 2.5\%$ . . . 47

C-1 Influence of varying CP ranks for $\rho = 0.6\%$ 59

C-2 Influence of varying CP ranks for $\rho = 2.5\%$ 60

C-3 Influence of varying CP ranks on Testing NMSE for $\rho = 0.6\%$ 60

C-4 Influence of varying CP ranks on Testing NMSE for $\rho = 2.5\%$ 61
C-5 Influence of varying Laplacian Regularization λ_l on Testing NMSE for $\rho = 0.6\%$ 62
C-6 Influence of varying Laplacian Regularization λ_l on Testing NMSE for $\rho = 2.5\%$ 63
C-7 Algorithm Run Times with varying CP rank for $\rho = 0.3\%$ 66
C-8 Algorithm Run Times with varying CP rank for $\rho = 0.6\%$ 66
C-9 Algorithm Run Times with varying CP rank for $\rho = 0.8\%$ 67
C-10 Algorithm Run Times with varying CP rank for $\rho = 2.5\%$ 67

List of Tables

1-1 Example User-Item Rating matrix for CF-based RS frameworks 2
1-2 Table of Symbols Used 7

4-1 Experiment Setup Parameters: Synthetic Dataset 32
4-2 Aggregate NMSE for varying CP-Rank 33
4-3 Aggregate NMSE for varying regularization λ_l 34
4-4 Aggregate NMSE for varying SNR (dB) 35
4-5 Aggregate runtimes for varying CP-Rank 37
4-6 Code Profile for Worst Case Algorithm Run Time with Total Average Time = 200.62s 37
4-7 Code Profile for BEST NMSE Algorithm Run Time with Total Average Time = 7.3s 37

5-1 Experiment Setup Parameters: MovieLens ML100K 41
5-2 Aggregate NMSE with varying CP Rank for $\rho = 0.6\%$ 42
5-3 Aggregate NMSE with varying CP Rank for $\rho = 2.5\%$ 43
5-4 Aggregate NMSE with varying Laplacian Regularization λ_l for $\rho = 0.6\%$ . . 45
5-5 Aggregate NMSE with varying Laplacian Regularization λ_l for $\rho = 2.5\%$ . . 45
5-6 Aggregate POF with varying CP Rank for $\rho = 0.6\%$ 46
5-7 Aggregate POF with varying CP Rank for $\rho = 2.5\%$ 46
5-8 Aggregate POF with varying λ_l for $\rho = 0.6\%$ and $\rho = 2.5\%$ 46
5-9 k-Cross Validation of the chosen models 48
5-10 Benchmarking with other CP tensor decomposition methods 48
5-11 Aggregate NMSE and Algorithm Run Times for varying ρ 49
5-12 Computational scalability for varying ρ 50

B-1 Aggregate runtimes for varying regularization λ_l 58
B-2 Aggregate runtimes for varying SNR (dB) 58

C-1 Aggregate Training NMSE with varying CP Ranks for $\rho = 0.6\%$ 61
C-2 Aggregate Testing NMSE with varying CP Ranks for $\rho = 0.6\%$ 61
C-3 Aggregate Training NMSE with varying CP Ranks for $\rho = 2.5\%$ 62
C-4 Aggregate Testing NMSE with varying CP Ranks for $\rho = 2.5\%$ 62
C-5 Aggregate Training NMSE with varying regularization λ_l for $\rho = 0.6\%$ . . . 63
C-6 Aggregate Testing NMSE with varying regularization λ_l for $\rho = 0.6\%$ . . . 63
C-7 Aggregate Training NMSE with varying regularization λ_l for $\rho = 2.5\%$ . . . 64
C-8 Aggregate Testing NMSE with varying regularization λ_l for $\rho = 2.5\%$ . . . 64
C-9 Aggregate POF with varying CP Rank for $\rho = 0.6\%$ 64
C-10 Aggregate POF with varying CP-Rank for $\rho = 2.5\%$ 65
C-11 Aggregate POF with varying regularization λ_l for $\rho = 0.6\%$ 65
C-12 Aggregate POF with varying regularization λ_l for $\rho = 2.5\%$ 65

Acknowledgements

Starting my journey in Systems and Control with a bachelor's degree in Electrical Engineering, my motivation to do this Master's Thesis was to learn. I take this opportunity to thank those who helped me during this process.

My parents, Chandrashekar Shankar and Krithika Chandrashekar, whose blessings I seek. My younger brother, Dilip Chandrashekar, for his 'elder brother' charisma. This journey would not have been possible without their encouragement, constant support, and sustained belief in me. No words can express my gratitude toward my family.

Curious and unfamiliar with the thesis topic, I cannot imagine the difficulties my professors, Dr.Ir. K. Batselier and Elvin Isufi must have had with me. Starting a year ago, their guidance in grounding my fundamentals was crucial. Helping me mature into what I present today as my thesis, I am grateful for the knowledge they have imparted to me. I also thank my Ph.D. supervisor Eva Memmel for her patience and support in times that were required the most.

I thank all the memories I made along with this thesis. Studying, Video calls, Delft Hyperloop, Music, Fridays, Cooking, TuD Library, Traveling, Celebrating festivities, India. I thank all the people who made them happen.

kāyena vācā manaseṃdriyairvā
budhyātmanā vā prakṛteḥ svabhāvāt
karomi yadyat sakalaṃ parasmai
nārāyaṇāyeti samarpayāmi

Delft University of Technology
November 21, 2022

Rohan Chandrashekar

sarvasya cāhaṁ hṛdi sanniviṣṭo
mattaḥ smṛtir jñānam apohanaṁ ca
vedaiś ca sarvair aham eva vedyo
vedānta-kṛd veda-vid eva cāham

Translation: I am seated in everyone's heart, and from Me come remembrance, knowledge and forgetfulness. By all the Vedas, I am to be known. Indeed, I am the compiler of Vedanta, and I am the knower of the Vedas.

- Shrimad Bhagavad Gita[1] 15:15

[1]Shrimad Bhagavad Gita, also known as 'The Song by God', is one of the oldest scriptures in Hindu literature. Part of the epic Mahabharata is set as a dialogue between Pandava prince Arjuna and his charioteer Krishna, the Supreme Personality of Godhead, addressing human life's ethical and moral struggles.

Chapter 1

Introduction

A surge in the growth of social networks has provided availability and accessibility of user data in this Internet era. Users actively create and share information on the World Wide Web and want recommendations that match their interests. This requires systems that can provide recommendations specific to user preferences. Recommender Systems (RS) have been vital in identifying and providing users with what they want by focusing on information retrieval, data mining, and machine learning [2, 3, 4, 5]. With exponentially increasing user-information data on which these RS frameworks operate, providing accurate and robust recommendations has increasingly been sought. A few applications of RS deployed are search engines (Google, Microsoft Bing, etc.), e-commerce websites (Amazon, Alibaba, etc.), music recommendations (iTunes, Spotify, etc.), and for movies, the famous Netflix Prize problem [6].

Given a set of users and items, RS gives recommendations by considering explicitly or implicitly available data. RS model this data by providing a rating basis for each unique user present. The rating basis comprises users providing ratings to an item or those implied from user actions and other available data. Once modeled, data filtering methods such as Content-based, Collaborative Filtering (CF), and Hybrid approaches are utilized.

Traditional RS incorporates CF, which has proven to be one of the most successful recommendation approaches. CF techniques can help recommend an item to a user based on similarities to other users in the RS framework [7, 8, 9]. Similar users are obtained based on correlations between users, items, or user-item pairs. It considers the ratings previously given and offers recommendations to a user for a particular item based on these ratings. These ratings are represented as a rating matrix formed between each unique user and item, as shown in Table (1-1). The question marks in the matrix are ratings that are to be completed.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	?	?	1	?	5
User 2	1	5	0	2	?
User 3	4	0	?	?	2
User 4	3	?	4	0	?
User 5	?	4	?	1	0

Table 1-1: Example User-Item Rating matrix for CF-based RS frameworks

CF can be classified based on how the data is processed in an RS framework. Memory-based approaches focus on finding similarities between users or items for recommendations. In contrast, model-based approaches use the given rating basis to train a model to predict recommendation ratings.

Memory-based approaches establish similarity measures between users, items, or user-item pairs. They predict the ratings of users based on the ratings of similar users for a user-based CF method or the ratings of users based on the ratings provided to similar items for an item-based CF [10]. Similarity measures are computed between users or items with the available ratings. The computation popularly uses Cosine similarity (Vector Space Similarity) or the Pearson Correlation Coefficient. [11]. Since the similarity measures depend on neighbors, the models are also known as Nearest Neighbor (NN) estimators. The kNN approach that utilizes similarity measures calculated between k-Nearest Neighbors can be interpreted as a **Graph**.

Constructing a graph that best captures the available data requires defining its structural and spectral properties. The choice of a function used to calculate similarity measures over the graph dictates the spectral properties of a kNN graph [12]. The constructed graph can further be used for regularization by including them as additional sources of information. Graph signal processing is utilized to analyze and extract information from the underlying graph structure [13].

Model-based approaches employ an end-to-end model to predict ratings [14]. These methods use dimensionality reduction techniques to make efficient predictions, particularly sparse rating matrices. Dimensionality techniques project a given matrix structure onto lower dimensions, thus, extracting the most fundamental factors that influence the matrix. Popular model-based methods include Probabilistic and Statistical analysis, Neural Networks, and Matrix Factorization techniques. Matrix Factorization (MF) techniques are often adopted where SVD and NMF are widely used [15, 16, 17].

Techniques predominantly based on MF decompose the rating matrix into several smaller matrices, choosing the most contributing factors of the matrix. The efficacy of MF methods in terms of accuracy and computational complexity is known. Multi-dimensional structures represent varied contexts if the data comprises more than two dimensions. Structures that can capture multi-linear interactions within the data while retaining matrix operations' benefits are sought.

Tensors capture relations and interactions between data points in higher-order dimensions. For example, a tensor of three dimensions can incorporate data such as users,

items, and time. The flexibility of the number of dimensions of a tensor enables capturing data beyond the user-item matrix for RS. In addition, tensor-based decomposition methods result in low-rank approximations (or) latent factors of a tensor. The original tensor can be reconstructed using the latent factors obtained from the decomposition. The reconstruction error using tensor decomposition methods is smaller than standard MF techniques [18]. While tackling the tensor decomposition problem, the focus areas are the interpretability of the factor matrices, uniqueness, storage, and run-time complexities as the number of dimensions increases [19].

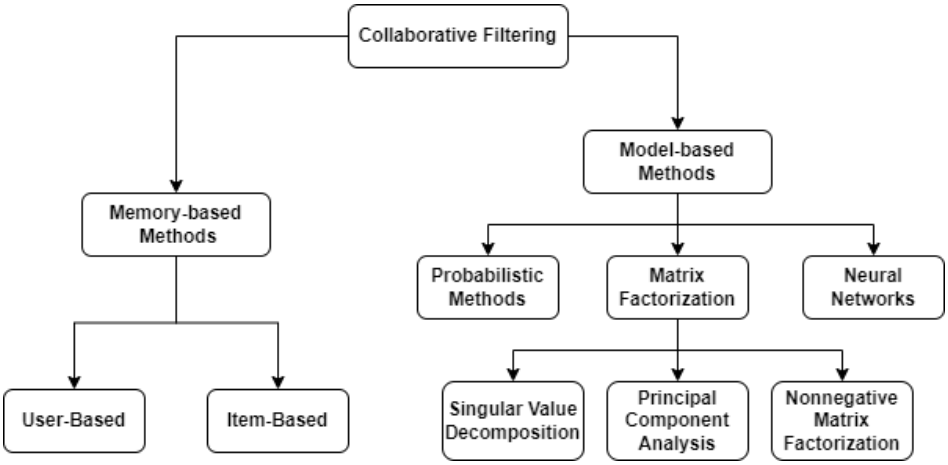

Figure 1-1: Collaborative Filtering Techniques

1-1 Recommender System Modelling

Let the set of users be represented by $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_P]$ and the set of items be represented by $\mathbf{I} = [\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_Q]$. Every user rates a subset of items with a rating score having a lower bound of r_{lb} and an upper bound of r_{ub}. A rating matrix is represented by $\mathbf{R} \in \mathbb{R}^{P \times Q}$ is created between the set of users and items where each entry is a rating score r_{ij} given by the unique user to an item. The objective of the recommender system is to predict the ratings for items that have not been given by users using the known ratings.

A well-known approach is to use a low-rank matrix factorization method that approximates the rating matrix $\mathbf{R} \in \mathbb{R}^{P \times Q}$ into factor matrices that best capture the interactions between user u and item i [16].

$$\mathbf{R} \approx \mathbf{A}^{\mathrm{T}}\mathbf{B}$$

where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_P] \in \mathbb{R}^{L \times P}$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_Q] \in \mathbb{R}^{L \times Q}$ with $L < min(P, Q)$. To learn the factor matrices $\mathbf{A}$ and $\mathbf{B}$, we minimize the squared error on the set of available ratings.

$$\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{Q} w_{ij} \left(r_{ij} - \mathbf{a}_i^{\mathrm{T}} \mathbf{b}_j \right)^2$$

where w_{ij} indicates whether a given user has rated an item (1) or not (0). Introducing regularization terms in the objective function can avoid overfitting the above model.

$$\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{Q} w_{ij} \left(r_{ij} - \mathbf{a}_i^{\mathrm{T}} \mathbf{b}_j \right)^2 + \frac{\lambda_1}{2} ||\mathbf{A}||_F^2 + \frac{\lambda_2}{2} ||\mathbf{B}||_F^2 \tag{1-1}$$

where $\lambda_1,\ \lambda_2 > 0$ and $||\cdot||_F^2$ is the squared Frobenius norm of a matrix.

Extending the rating matrix to a multi-dimensional problem has benefited from using tensors. Using tensor factorization models, incorporating specific contexts with user-item ratings is possible. These contexts could be user attributes, item properties, or temporal contexts. Introducing a low-rank tensor factorization model, the rating tensor $\mathcal{T}$ is approximated to $\hat{\mathcal{X}}$ as $\mathcal{T} \approx \hat{\mathcal{X}} = [[\lambda; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(d)}]]$ where $\mathcal{T} \approx \hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ is the approximated rating tensor with d dimensions. The approximated tensor is factorized into factor matrices $[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(d)}]$ with normalizing coefficient $\lambda \in \mathbb{R}^R$. The dimensions of the factor matrices are $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R}$ with $n \in [1, d],\ R \in \mathbb{Z}^+$. Adapting Equation (1-1) to tensors, the loss function is written as,

$$\min_{\hat{\mathcal{X}}} \|\mathcal{W}_\Omega (\mathcal{T} - \hat{\mathcal{X}})\|^2 + \Psi(\hat{\mathcal{X}}) \tag{1-2}$$

where $\mathcal{T}$ is the ground truth tensor, $\hat{\mathcal{X}}$ is the tensor being reconstructed, $\mathcal{W}_\Omega$ indicates whether a given user has rated an item or not, and $||\cdot||^2$ is the squared tensor norm. $\Psi(\hat{\mathcal{X}})$ is a regularization function that regularizes the factor matrices being minimized. Combining functions that are known to aid predictions for RS while avoiding model overfitting is sought. Having defined the objective function, a combination of sum-of-squared errors with regularization terms, we can incorporate an Alternating Least Squares (ALS) or a gradient-based method to learn the factor matrices.

1-1-1 Benefits and Challenges

Rating matrices used in RS modeling are of significant dimensions and are sparse. As a result of having less data, the recommendation problem becomes challenging when making accurate and robust predictions. Some of the known issues of these types of RS are sparsity of the user-item rating matrix, cold-start issues, user biases that may be inherently present in the data, or time-varying user preferences [20]. While sparsity is prevalent with existing users and items, adding new users or items leads to a cold start where the RS knows very little or nothing to make recommendations. Biases in rating values are assumed to be independent of user-item interactions where the system has users providing higher ratings to a specific subset of items. The issues mentioned above are when the rating matrix is static, i.e., it does not account for temporal effects that might result in user-item ratings drifting to time.

Discussing RS using traditional CF models, memory-based approaches are easier to implement and scale well with correlated items. Graphs inherently capture information that might be intrinsic to the built RS. However, these approaches are not viable when presented with sparse rating matrices. Influenced by user ratings previously given, the

models need to be corrected for any bias or temporal drifts that might be present. Straightforward to implement, limited scalability constrains the use of memory-based models. This is due to the generation of similarity measures between entities with increased data, i.e., nodes to be processed [21].

Model-based approaches better address sparsity and scalability issues with improved prediction capabilities, despite losing information due to dimensionality reduction. The challenges of this approach appear when deploying them for real-time applications. Training end-to-end models using model-based methods requires significant computational resources each time. They can make better and more robust predictions but come at a trade-off between scalability and performance prediction [22].

1-2 Thesis Motivation and Research Aims

Current literature elaborates on how memory and model-based models using traditional CF algorithms capture and process data to make recommendations. The drawbacks of traditional CF algorithms are significant when scaling the algorithm for RS handling sparse data.

The main contribution of this thesis is the Graph Regularized CP (GRCP) tensor decomposition model framework for RS. This thesis formulates a unified model using complementary data sources. Furthermore, we aim to leverage tensors to accommodate multi-contextual data and graph structures to capture underlying intrinsic relationships in the data to build the GRCP model. The main research question that we focus on in this thesis is:

How does the GRCP model make better recommendations for large-scale RS applications?

The research question is broken down into further sub-questions to answer the main research question.

(RQ.1) *How does the GRCP model capture and utilize available data?*

The GRCP model captures available data using tensors and graphs. First, we shall cover the literature on how tensors and graphs can represent data to model latent interactions. Tensors can accommodate data having varied contexts such as users, items, time, etc. Similarly, graphs effectively capture similarity measures between varied contexts in the data. The literature also covers the fundamental mathematical operations on shared underlying data structures to extract and use the information captured.

(RQ.2) *How is the data combined in GRCP to aid recommendations?*

The focus of this research question shall be on the methodology available to utilize the available data effectively. Exploring possibilities of data fusion from varied sources is proposed to build a robust and accurate RS. Having covered sufficient literature highlighting the benefits of combining two sources of information, the GRCP model is ideated. The proposed model consists of an error-fitting term and a regularization

term. We particularly focus on the regularization term in Equation (1-2), aiming to incorporate regularizer functions to alleviate the sparsity problem in RS.

(RQ.3) *How does the GRCP address the drawbacks that current RS are susceptible to?*

The thesis shall address the final question by conducting extensive numerical experiments on synthetic and real datasets. The proposed model converges to a global optimum of the problem by using an Alternating Least Squares (ALS) algorithm with a Conjugate Gradient (CG) solver. Evaluation metrics such as Normalized Mean Squared Error (NMSE), Percentage of Fit (POF), and the computational complexity of the proposed solver evaluate the proposed model. While scaling the model, one must also ensure solvability by utilizing minimal computational resources and run-time.

1-3 Thesis Outline

The rest of the thesis document is structured as follows. **Chapter 2** consolidates the necessary background for this thesis. **Chapter 3** covers the literature and methodology proposed to develop the Graph Regularized Tensor Decomposition model framework. **Chapter 4** tests and investigates the proposed model framework for its applications in image representation and denoising. **Chapter 5** evaluates the proposed model framework on datasets for an RS application and highlights the obtained performance metrics. Finally, **Chapter 6** consolidates the work carried out and discusses this thesis's future scope of work.

The code repositories for the GRCP model framework implemented for the Synthetic Dataset and MovieLens Dataset is made available in the hyperlinks given.

Table 1-2: Table of Symbols Used

Symbol	Significance
x, $\mathbf{x}$, $\mathbf{X}$, $\mathcal{X}$	scalar, vector, matrix, tensor
$\circ$	Outer Product
$\otimes$	Kronecker Product
$\odot$	Khatri-Rao Product
$\times_n$	n-mode Product
$\mathbf{X}_{(n)}$	n-mode matricization of tensor $\mathcal{X}$
$\mathbf{X}^{\mathrm{T}}$	Matrix Transpose
$\mathbf{X}^{\dagger}$	Moore-Penrose Pseudoinverse
$\|\|\mathbf{A}\|\|_F$	Frobenius norm of a matrix
$\mathbb{R}^{I_d}$	Set of all real numbers in the dimension $I_d \in \mathbb{Z}^+$

Chapter 2

Tensors and Graphs: A Mathematical Background

2-1 Motivation

Recommender System (RS) models make recommendations using data available to the system. The information available can be explicit (user ratings, user-item information) or implicit (user behavior, trends, and interactions) [23]. RS frameworks process the data by employing state-of-the-art tools. Filtering algorithms are typically used depending on the context for whom and what recommendations are made.

Unlike matrices that capture data having only binary relations, tensors can capture higher-order relations. These interactions between data allow an RS to incorporate contextual information to make accurate and context-aware recommendations. On the other hand, graphs capture complex interactions between data beyond those given as a matrix. This chapter discusses the fundamentals of tensor theory and graph signal processing.

2-2 Tensor Theory

2-2-1 Notations and Basics

Tensors are generalized representations of multi-dimensional arrays. The order of a tensor is the number of modes or the dimension space it covers. A vector represents a tensor of order one, a matrix represents a tensor of order two, and tensors having order d represent an dth-order or high-order tensors. Each tensor can be represented as an outer product of d vector spaces. Tensor algebra deals with different notations for tensors of various orders. In this thesis, we denote vectors symbolized by boldface

lowercase letters, e.g., $\mathbf{x}$, matrices symbolized by boldface capital letters, e.g., $\mathbf{X}$ and higher-order tensors symbolized by boldface Euler script letters, e.g., $\mathcal{X}$.

Tensor Elements: The elements present in a tensor are indexed by the number of indices the tensor contains. The order of the tensor dictates the number of indices, e.g., a third-order tensor element present in $\mathcal{X}$ is given by the indices (i, j, k) and an element present in the tensor is denoted by x_{ijk}. The two most commonly used representations for tensors are fibers and slices. Fibers are obtained by fixing all indices of the tensor except one, while slices represent two-dimensional sections of a tensor, i.e., all indices are fixed except two.

Definition 1. ***[1] Tensor Norm:*** *The tensor norm of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ is given by the square root of the sum of the square of all its elements $\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_d=1}^{I_d} x_{i_1 i_2 \cdots i_d}^2}$.*

Definition 2. ***[24] Tensor Nuclear Norm:*** *The tensor nuclear norm of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$, $n \in [1, d]$ is defined as,*

$$\|\mathcal{X}\|_* := \text{infmum} \left\{ \sum_{r=1}^{R} \left\| \mathbf{x}_r^{(1)} \right\| \cdots \left\| \mathbf{x}_r^{(d)} \right\| : \mathcal{X} = \sum_{r=1}^{R} \mathbf{x}_r^{(1)} \circ \cdots \circ \mathbf{x}_r^{(d)}, \mathbf{x}_r^{(n)} \in \mathbb{R}^{I_n}, R \in \mathbb{Z}^+ \right\}$$

This can be expressed as,

$$\|\mathcal{X}\|_* = \text{infmum} \left\{ \sum_{r=1}^{R} |\lambda_r| : \mathcal{X} = \sum_{r=1}^{R} \lambda_r \mathbf{x}_r^{(1)} \circ \cdots \circ \mathbf{x}_r^{(d)}, \left\| \mathbf{x}_r^{(n)} \right\| = 1, R \in \mathbb{Z}^+ \right\} \tag{2-1}$$

Definition 3. ***[1] Tensor n-Mode Matricization:*** *Reordering the elements of an d-way tensor to represent it as a matrix is called as matricization of a tensor. It is also known as the unfolding or flattening of a tensor to represent it as a matrix given its n-th mode. This means arranging the mode-n fibers of a given tensor to be columns of the resulting matrix. The n-mode matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ can be denoted as $\mathbf{X}_{(n)}$ who's dimensions are given by $I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_d)$.*

Taking an example, let the frontal slices of tensor $\mathcal{X} \in \mathbb{R}^{3 \times 4 \times 2}$ be,

$$\mathbf{X}(:,:,1) = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{X}(:,:,2) = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

The order of the tensor being three will have three unfoldings of the tensor given by,

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \cdots & 21 & 22 & 23 & 24 \end{bmatrix}$$

2-2-2 Tensor Operations

Definition 4. *[1] **Tensor Inner Product:** Given two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$, the inner product is defined as the sum of the products of their respective entries given by, $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_d=1}^{I_d} x_{i_1 i_2 \cdots i_d} y_{i_1 i_2 \cdots i_d}$. This immediately can be confirmed by computing the square of the norm of the given tensor, i.e., $\langle \mathcal{X}, \mathcal{X} \rangle = ||\mathcal{X}||^2$.*

Definition 5. *[25] **Kronecker Product:** Given two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, the Kronecker product of these matrices is given by $\mathbf{A} \otimes \mathbf{B}$. This resulting matrix is of size $(KI) \times (LJ)$.*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\ \mathrm{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\ \mathrm{B} & a_{I2}\ \mathrm{B} & \cdots & a_{IJ}\ \mathrm{B} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes & \mathbf{b}_2 & \mathbf{a}_1 \otimes \mathbf{b}_3 & \cdots & \mathbf{a}_J \otimes \mathbf{b}_{L-1} & \mathbf{a}_J \otimes \mathbf{b}_L \end{bmatrix}$$

Definition 6. *[25] **Khatri - Rao Product:** Also known as the "matching columnwise" Kronecker product, given two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, the Khatri - Rao product is given by $\mathbf{A} \odot \mathbf{B}$. The resulting matrix is of size $(JI) \times K$.*

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}$$

Definition 7. *[1] **Tensor n-Mode Product** The tensor n-mode product multiplies a tensor by a matrix (or a vector) in its n-th mode. The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n \mathbf{U} \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times J \times I_{n+1} \times \ldots \times I_d}$. Expanding it elementwise, it is $(\mathcal{X} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j\ i_{n+1} \cdots i_d} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_d} u_{j i_n}$.*

Each mode-n fiber is multiplied by the matrix $\mathbf{U}$ and can be expressed in terms of the unfolded tensors as $\mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)}$. Taking an example for the same, we consider the tensor as in Definition 3. Given a matrix $\mathbf{U} \in \mathbb{R}^{2 \times 3}$ where $\mathrm{U} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$. The Tensor n-Mode matrix product $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U} \in \mathbb{R}^{2 \times 2 \times 2}$ is,

$$\mathbf{Y}(:,:,1) = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix}, \quad \mathbf{Y}(:,:,2) = \begin{bmatrix} 130 & 157 & 184 & 211 \\ 172 & 208 & 244 & 280 \end{bmatrix}$$

2-2-3 CANDECOMP/PARAFAC (CP) Tensor Decomposition

The canonical decomposition (CANDECOMP) and parallel factors (PARAFAC), also known as the CP decomposition, breaks down a given tensor into its corresponding component rank-one factors [26, 27]. The sum of these factors gives back the closest approximation to the original tensor.

Definition 8. *[1] **CP Tensor Rank:** The CP rank of a given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ is defined as the sum of minimum number of R rank-one tensors required to generate the original tensor $\mathcal{X}$.*

By combining the rank-one components as columns, the same can represent what is known to be factor matrices given by $\mathcal{X} = [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(d)}]$ with $\mathbf{X}^{(n)} = \sum_{r=1}^{R} \mathbf{x}_r^{(n)}$ where $\mathbf{x}^{(n)} \in \mathbb{R}^{I_n},\ n \in [1, d]$. The given tensor $\mathcal{X}$ is said to be of rank one if it can be expressed as the outer product of d vectors, which is given by $\mathcal{X} = \mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \ldots \circ \mathbf{x}^{(d)}$.

Definition 9. *[1] The CP decomposition of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ is an approximation of the given tensor with the sum of its respective factors given by $\mathcal{X} \approx \hat{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{x}_r^{(1)} \circ \mathbf{x}_r^{(2)} \circ \ldots \circ \mathbf{x}_r^{(d)}, \|\mathcal{X} - \hat{\mathcal{X}}\|^2 < \epsilon$; where $\mathbf{x}^{(n)} \in \mathbb{R}^{I_n}$ for $R \in \mathbb{Z}^+,\ n \in [1, d]$ and $\epsilon > 0$.*

Considering a third-order tensor as seen in Figure (2-1) for further discussion, each of the rank-one components can be combined to form its factor matrices, i.e., $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_R]$ with similar representations for the $\mathbf{B}$ and $\mathbf{C}$ matrices. These factor matrices represent the CP model as $\hat{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$. The columns of the factor matrices are normalized to length one with their weights absorbed into a normalizing coefficient $\lambda \in \mathbb{R}^R$.

$$\hat{\mathcal{X}} = [\![\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

2-2-3-1 Alternating Least Squares (ALS) Algorithm

Straightforward to implement and provide unique decompositions while being memory efficient, the CP-ALS has been a prevailing optimization approach to solving the tensor decomposition problem. The key idea is to fix all the available factor matrices except the one to be optimized. This step is repeated for all the available factor matrices until a suitable stopping criterion has been met. The unconstrained objective function for the ALS algorithm to minimize can be written as $\min_{\hat{\mathcal{X}}} \|\mathcal{X} - \hat{\mathcal{X}}\|^2$ where $\hat{\mathcal{X}} = [\![\lambda, \mathbf{X^{(1)}}, \mathbf{X^{(2)}} \ldots \mathbf{X^{(d)}}]\!]$ with $\mathbf{X}^{(n)} = \sum_{r=1}^{R} \mathbf{x}_r^{(n)},\ n \in [1, d]$ and λ is the normalizing coefficient for the columns of the obtained latent factors.

Algorithm 1 CP-ALS Algorithm for order d Tensor [1]

1: **function** CP-ALS($\mathcal{Y}$,R)
2: initialize $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n \in [1, d]$
3: **repeat**
4: **for** n = 1 ,..., d **do**
5: $\mathbf{A} \leftarrow \mathbf{X}^{(1)\mathrm{T}}\mathbf{X}^{(1)} \ldots \mathbf{X}^{(n-1)\mathrm{T}}\mathbf{X}^{(n-1)}\mathbf{X}^{(n+1)\mathrm{T}}\mathbf{X}^{(n+1)} \ldots \mathbf{X}^{(d)\mathrm{T}}\mathbf{X}^{(d)}$
6: $\mathbf{X}^{(n)} \leftarrow \mathbf{Y}_{(n)}(\mathbf{X}^{(d)} \odot \ldots \odot \mathbf{X}^{(n+1)} \odot \mathbf{X}^{(n-1)} \odot \ldots \odot \mathbf{X}^{(1)})\mathbf{A}^{\dagger}$
7: normalize columns of $\mathbf{X}^{(n)}$ and store norms as λ
8: **end for**
9: **until** max. iterations reached or stopping criteria are met
10: **return** $\lambda, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(d)}$
11: **end function**

The CP-ALS is unique under mild conditions, combined with additional constraints. Computationally, the CP-ALS algorithm's main drawback is the MTTKRP, as given in line 6 of the Algorithm (1). Primarily, it leads to a larger convergence time for the algorithm. It also

includes the potential for the algorithm to be numerical ill-conditioning and non-convergence to the global optimum. Regularization constraints on the factor matrices help impose conditions for the algorithm's numerical stability and global convergence. Another drawback is that there is no defined method to choose the number of rank-one components R for the CP-ALS algorithm. This depends on the available data and is considered a tunable hyperparameter in the optimization process [28, 29].

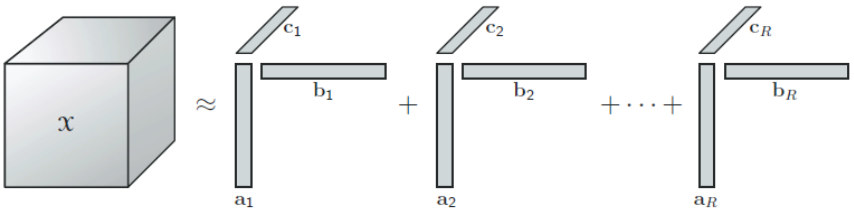

Figure 2-1: CP Decomposition of a 3-way tensor into its factor approximations [1]

2-3 Graph Theory

2-3-1 Notations and Basics

Graphs are nonlinear data structures that contain a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ between connected nodes, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In the case of a weighted graph, it is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{W}$ are the assigned weights to the edges connected between nodes.

Definition 10. ***[30] Undirected Graphs:*** *A graph where the set of connected nodes is bidirectional. Given nodes u and v, then the edge $(u, v) \in \mathcal{E}$ connecting them is equivalent to $(v, u) \in \mathcal{E}$. The neighbourhood set of node u is given as $\mathcal{N} := \{v | (u, v) \in \mathcal{E}\}$. The weight matrix $\mathcal{W}$ for such graphs is symmetric.*

Graph Linear Algebra: The structure of a graph is understood by representing it as algebraic matrices. These matrices depict how the nodes in a graph interact with each other.

- Adjacency Matrix: The adjacency matrix $\mathbf{A}$ of a graph gives the connectivity between its various nodes. Unweighted graphs have entries of the matrix to be binary; 1 if two nodes are connected and 0 if they are not.
- Degree Matrix: The degree of a node u in an undirected graph is equal to the sum of the weights of all edges connected to the particular node, i.e., $d_u := \sum_{v=1}^{N} \mathbf{A}_{uv}$. The matrix is defined to be a diagonal matrix given by $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$.
- Laplacian Matrix: Defined for undirected graphs, the Laplacian matrix $\mathbf{L}$ is the difference between the degree and adjacency matrix, i.e., $\mathbf{L} = \mathbf{D} - \mathbf{A}$. It is a symmetric and positive semidefinite matrix.

(a) Undirected Graph **(b)** Weighted Graph

Figure 2-2: Graph structures depicted with nodes as circles and edges connecting nodes as lines

2-3-2 Graph Signals and Spectra

A graph signal assigns a real signal value to the nodes. Each signal $\mathbf{x}$ maps the node set i to the real set, i.e., $\mathbf{x}_i : i \longrightarrow \mathbb{R}$. Each edge connecting these nodes plays a role in determining how the signal values diffuse over the network.

Definition 11. ***[31] Graph signal shifting:*** *Given a graph shift operator* $\mathbf{S}$*, graph signal shifting is defined as* $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}$ *where,* $\mathbf{x}^{(1)}$ *stands for the one-shift of a graph signal* $\mathbf{x}$*. The shift operator dictates how signal values diffuse to neighboring nodes in a graph.*

Depicting a more general case of the shift operation, the k^{th} *shift of a graph signal over a graph network can be computed as* $\mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{x}) = \mathbf{S}\mathbf{x}^{k-1},\ k \in \mathbb{Z}_0^+$.

Given a signal $\mathbf{x}$ over an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we are interested in finding the variation of the signal over a particular edge $e_{i,j} = (u, v)$ being valued at node u. This is given by the edge derivative [30]. The edge derivative $\frac{\partial \mathbf{x}}{\partial e_{i,j}}\Big|_u = \sqrt{\mathbf{S}_{i,j}}\,(x_i - x_j)$ indicates how much a given signal value changes in two connected nodes. The larger the derivative, the more variation in signal values x_i and x_j. This notion of signal variation can be applied to entire graph networks using the p-Dirichlet form of $\mathbf{x}$.

$$S_p(\mathbf{x}) = \frac{1}{p}\sum_{u_k \in \mathcal{V}} \|\nabla_{u_l}\mathbf{x}\|_2^p = \frac{1}{p}\sum_{u_k \in \mathcal{V}} \left(\sum_{\nu_j \in \mathcal{N}_i} \mathbf{S}_{i,j}\,(x_i - x_j)^2\right)^{\frac{p}{2}}$$

$S_1(\mathbf{x})$ refers to the total signal variation across a graph network while $S_2(\mathbf{x})$ refers to the graph Laplacian quadratic form given as $S_2(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x}$. The latter is used to measure signal variation predominantly in undirected graphs, measuring the smoothness of a signal. The smaller the smoothness measure, the slower the signal changes over the graph. Similarly, the larger the smoothness measure, the more rapidly the signal changes over the graph.

This particular shift operator could be either of the graph matrices discussed previously. For $\mathbf{S}$=$\mathbf{A}$, the graph shift operation takes the signals from the neighbors and combines them with the edge weights. For $\mathbf{S}$=$\mathbf{L}$, the operation takes information from the neighboring nodes and merges them with the current node's state.

Graph Spectra: Studying the eigenvalue decomposition of the graph algebraic matrices, as mentioned in Section (2-3-1), has important applications in graph spectral theory [12]. The

spectral analysis requires a symmetric matrix, which can be carried out on a given graph's Laplacian matrix $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$; where, $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ is a diagonal matrix with the eigenvalues increasing along the diagonal ($0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$) and eigenvectors $\mathbf{U}= [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$. Certain properties of the above Laplacian eigendecomposition are:

- Given the eigenvalues of Λ are arranged in increasing order, the constant unit energy eigenvector $\mathbf{u}_1$ corresponding to the first eigenvalue $\lambda_1 = 0$ signifies the DC energy component of the graph. This is given by $1/\sqrt{N}$.
- The eigenvectors $\mathbf{u}_k$ associated with lower magnitude λ_k correspond to elements varying slowly across the graph. Similarly, those eigenvectors $\mathbf{u}_k$ associated with higher magnitude λ_k correspond to elements varying quickly across the graph.

2-3-3 Generating Similarity Measures: Graph Kernels

Kernels map a particular input set Ω to a set of real numbers $\mathbb{R}$. These mapping functions $k(\cdot, \cdot) : \Omega \times \Omega \longrightarrow \mathbb{R}$ measure the similarity between the entities in the input set.

Kernel functions give meaning to measures such as the distance between two graph nodes. One such application is to find similar entities in a graph defined by this distance. Furthermore, they are extensively used to statistically learn an adequate similarity measure to analyze nonlinear interactions between entities in the graph network [32]. Therefore, constructing a valid kernel that can capture information inherent to a graph structure is essential.

Given a set of entities $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Omega$ of rank r, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{n \times r}$, a $n \times n$ symmetric kernel matrix $\mathbf{K}$ can be generated. The entries of $\mathbf{K}$ are inner products of the set of entities i.e., $[\mathbf{K}]_{ij} = k_{ij} = \mathbf{x_i}^{\mathrm{T}}\mathbf{x_j}$. Although simple to evaluate, the inner product is not always a preferred similarity measure as the kernel matrix should satisfy the property of being positive semidefinite. Real symmetric matrices not meeting the requirement of being positive semidefinite can be made so with the addition of an identity matrix multiplied by a scalar constant $\alpha \geq |\lambda_{min}|$ such that all the eigenvalues of the kernel matrix are non-negative and $(\mathbf{K} + \alpha\mathbf{I})$ is positive semidefinite [33].

One kernel that has shown consistent performance in information retrieval and collaborative filtering applications is the regularized graph Laplacian kernel. Based on graph spectral theory, the kernel has been widely used to capture information propagation in social networks. The kernel matrix can thus be computed as $\mathbf{K} = \sum_{k=0}^{\infty} \alpha^k (-\mathbf{L})^k = (\mathbf{I} + \alpha\mathbf{L})^{-1}$ where, $0 < \alpha < \rho(\mathbf{L})^{-1}$ and $\rho(\mathbf{L})$ is the spectral radius of $\mathbf{L}$. The kernel variables for calculating the similarities are obtained by attributing spectral information with each node attribute.

2-3-4 Regularization with Graph Laplacian

The prior knowledge about how a particular signal varies is fundamental in regularizing and retrieving signals of interest from a given graph network. To estimate the signal of interest $\mathbf{x}$ from noisy observations $\mathbf{y}$, we need to solve the following optimization problem,

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{R}^N}{\mathrm{argmin}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \gamma f(\mathbf{x}; \mathbf{S}) \tag{2-2}$$

where, $\hat{\mathbf{x}}$ is the optimal estimated signal, $\|\mathbf{y} - \mathbf{x}\|_2^2$ is the quadratic optimization objective function which we are looking to minimize between the signal $\mathbf{x}$ and noisy observations $\mathbf{y}$. $f(\mathbf{x}; \mathbf{S})$ is the regularisation term which acts as a penalty function for the estimated signal using the graph shift operator S. This incorporates prior information about the graph signal $\mathbf{x}$ while estimating the optimal signal $\hat{\mathbf{x}}$.

Using the graph Laplacian $\mathbf{L}$ as the shift operator, the most commonly used regularizer is given by $f(\mathbf{x}; \mathbf{S}) = \mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x}$, which gives the signal smoothness measure over the set of observations guarantees smoothness of the estimated signals over the graph. This is also known as the graph Laplacian regularizer (GLR). The variable $\gamma > 0$ dictates the trade-off between the objective function and the regularisation term. $\gamma \longrightarrow 0$ results in the optimization problem using little prior information from the graph structure. $\gamma \longrightarrow \infty$ gives more weight to the prior information and less to the observations, which is used in the cases where the observations $\mathbf{y}$ are excessively corrupted by noise. Using a penalized regularizer equally has its limitations. The global smoothness measure is penalized rather than the variability in a node's surroundings.

Thus, we can rewrite the Equation (2-2) by imposing different penalties on different nodes. A local regularisation problem can be formulated as follows, $\hat{\mathbf{x}}(\omega) = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{x}\|_2^2 + (\mathrm{diag}(\omega)\mathbf{x})^{\mathrm{T}}\mathbf{L}(\mathrm{diag}(\omega)\mathbf{x})$ where $\mathrm{diag}(\omega)$ is a diagonal matrix with $\omega = [\omega_1, \omega_2, \ldots, \omega_N] \in \mathbb{R}^N$ being the different penalty weights imposed on the regularization term [34].

2-4 Discussions

Several efficient and state-of-the-art tensor decomposition methods are available in the literature. The CP is an explanatory model, i.e., the factor matrices of the decomposition offer insights into the variation of contextual data. Unlike matrix decompositions, the CP decomposition method using an ALS method is unique under much weaker conditions. The effects of scaling indeterminacy and matrix degeneracy leading to numerical instability must be considered. Improper scaling could lead the algorithm to converge at a local minimum of a fixed iteration. Methods to normalize and apply low-rank constraints to each factor matrix have been found to solve these problems [1, 35].

The ALS method is the "workhorse" algorithm to solve a CP decomposition [1]. Although it takes more iterations to converge, the solution quality is better for a chosen CP rank R than other popular tensor decomposition methods [36]. Linearizing the nonlinear block Gauss-Seidel algorithm makes it computationally advantageous [35]. The minimizer globally converges, provided the function is coercive and real analytic [37]. Derivative-based algorithms require more memory and computational time and no guarantee of the minimizer converging globally [38]. A trade-off between the dataset dimensions, algorithm convergence properties, and computational capability is made, and the ALS method is chosen for this thesis.

Building relevant graph structures for diverse data, considering its source and characteristics, is essential while building RS. This data can be classified as interaction data (user-item interaction matrix) or side information data (user preferences and item attributes varying over time) [39]. Each data class has a specific graph structure to capture the data to the best possible extent. kNN graphs are undirected graphs representing similar entities or nodes in

a graph. The interactions between each unique node are edges that connect two nodes. The interactions can be mapped to similarity coefficients and represented as edge weights using graph kernel functions.

Graphs have several advantages in RS applications in mitigating issues of sparsity [40, 41]. The choice of graph Laplacians is well motivated to be used as a regularizer. In this thesis, the GLR is utilized to maintain any structure in the data. Treating the data as graph signals, graph signal processing is used to build the graph Laplacians [30]. Its symmetric and positive semi-definite property makes it favorable for optimization algorithms that require such matrices.

2-5 Conclusion

This chapter reviews the necessary background for the research topics to be discussed in this thesis. It highlights the theory behind the structures and mathematical operations used in the thesis. First, we discuss basic tensor notations, tensor products, and the CP tensor decomposition method. Core to model training, we also highlight the standard optimization algorithm for CP tensor decomposition methods. Next, we review graph basics and how graph structures are represented as matrices. Finally, we discuss how to compute graph spectra on graph networks and the algebra behind graph signal processing. Focus is given to graph regularization using graph Laplacians and its efficacy in information retrieval.

Chapter 3

Graph Regularized (GRCP) Tensor Decomposition

3-1 Motivation

Capturing data with higher-order relations while exploiting its underlying geometric structure is key to building accurate and efficient RS models. A literature review highlighting these is provided. Improving prediction metrics such as NMSE with model scalability has been core to tensor decomposition methods. Combining heterogeneous data sources has proven beneficial in building accurate, scalable, and computationally efficient RS frameworks.

This chapter proposes a Graph Regularized CP (GRCP) tensor decomposition model framework. The objective function is a combination of real analytic and semialgebraic functions. An ALS method solves the objective function by keeping all but one of the factor matrices constant. These factor matrices are locally minimized using a Polak Ribiere Conjugate Gradient (CG) solver. The convergence and computational complexity of the proposed method are also discussed.

3-2 Literature Review

3-2-1 Tensors for RS

Tensors are being used for RS applications to build context-aware models. For example, users assigning ratings to items have several factors, such as a particular tag relating to a user and item or time-changing user preferences and item relevance. These contexts highlight data with higher-order relations that a standard matrix cannot capture.

Using GPS data, [42] develops a tensor-based recommender system considering activities carried out by users at a given location. The objective contains graph Laplacian regularized users and activity factor matrices. A CP decomposition is carried out, solving the factor matrices

using a gradient-based solution. However, they do not discuss the initialization methods of the algorithm, given that a gradient-based approach does not provide a global solution. The algorithm's scalability is questionable given each gradient iteration takes $\mathcal{O}(mnr+m^2+r^2)$ operations where m, n are the tensor dimensions, and r is the CP rank of the model. This scales quadratically with an increase in tensor dimensions, becoming significant for $r \ll n < m$.

Context-aware Point of Interest recommender systems developed in [43] use information obtained from check-in data that are location specific. They construct a user-location-time frame tensor $\mathcal{X}$ and apply the CP decomposition to reconstruct the given tensor as $\hat{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{x}_r^{(1)} \circ \mathbf{x}_r^{(2)} \circ \mathbf{x}_r^{(3)}$. They additionally impose the user's social connections $\mathbf{L}$ as additional regularization to improve the accuracy of predictions.

$$\min_{\mathbf{X}^{(1)},\mathbf{X}^{(2)},\mathbf{X}^{(3)}} \frac{1}{2}\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 + \frac{1}{2}\,\mathrm{Tr}\left\{\mathbf{X}^{(1)^{\mathrm{T}}}(\lambda\mathbf{I} + \alpha\mathbf{L})\mathbf{X}^{(1)}\right\} + \frac{\lambda}{2}\,\mathrm{Tr}\left\{\mathbf{X}^{(2)}\mathbf{X}^{(2)^{\mathrm{T}}} + \mathbf{X}^{(3)}\mathbf{X}^{(3)^{\mathrm{T}}}\right\}$$

where $\{\mathrm{Tr}\}$ is the matrix trace operator. Successfully capturing the ternary relationships present within the data obtained from Location-Based Social Networks, the suggested method shows better Precision and Recall metrics compared to other baseline algorithms. The ALS algorithm provides no information on how the gradients are evaluated.

With the focus shifting to incorporate data with multiple contexts, students' grades are predicted for new courses in [44], investigating a CP tensor decomposition method. They propose a model allowing side information integration without increasing the required tensor rank. The authors conclude that models incorporating additional contextual information have better prediction metrics than matrix factorization methods. They, however, do not impose a low-rank solution in their decomposition models, and the convergence of their solution is poorly motivated.

The introduction of the nuclear norm to matrix factorization for RS applications is discussed in [45]. The article provides an overview of applications extended to CP tensor decompositions. Coupled matrix-tensor decomposition methods incorporating smoothness measures and rank regularization such as $\|\mathbf{X}\|_* = \min_{\mathbf{A},\mathbf{B}|\mathbf{X}=\mathbf{A}\mathbf{B}^{\mathrm{T}}} \frac{1}{2}\left(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2\right)$ are proposed. Important to note that given the model's rank is to be penalized, one cannot impose orthogonality constraints of the nature $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I}$. Several possible design choices to formulate the objective function are discussed, some of which are used in this thesis.

Recently, authors in [46] propose a tensor factorization model that incorporates users' trust and implicit feedback in their model. The trust model consists of two types, where the weighting used to model the trust factors uses constant and similarity functions. The model incorporates various biases and factor matrix norm regularization. A gradient descent approach is adopted to minimize the loss function. The same model is extended to a time-varying bias factorization model in [47] with the contexts provided to the tensor dimensions. Interestingly, varying the number of time bins in a user-item-time context model provides different results for the same dataset.

3-2-2 Graphs for RS

Graph regularization has been applied to matrix and tensor factorization methods. This aids methods in a matrix or tensor completion problem by appending additional information [48]. The choice of penalty penalization and spectral properties of the regularizer have been studied

and have found uses in decomposition-based methods. As a result, the obtained models are robust, scalable, and better at predicting values providing low reconstruction error measures.

One of the most common penalization methods to solve a least-squares problem is the Tikhonov regularization [49]. Two well-known regularizer functions of this form are the nuclear norm and GLRs. Minimizing objective functions of this form to obtain direct numerical solutions is possible. The data-fitting term is commonly penalized with a GLR and has various applications in data classification, semi-supervised learning, and graph signal denoising [50, 34]. For building RS model frameworks, the end objective is to balance the data-fitting term with the regularization term [51].

It was first noted in [52] that matrix factorization techniques could significantly improve the learning performance of data points having similar embeddings. They considered the local invariance and exploited the underlying geometric structure in the data. The factor matrices are solved using a gradient descent-based method. The model is evaluated on parameters such as the number of nearest neighbors and the weighting scheme selected to capture the geometric structure in the data. Discussions on parameters for datasets containing images are carried out. The proposed model is better than the baselines included. However, algorithm run-time and scalability comments for the proposed model are insufficiently provided.

Following this, [53] highlights tackling the low-rank matrix completion problem by incorporating pairwise relationships among variables known via graphs $\mathbf{L}_{(a)}$ and $\mathbf{L}_{(b)}$. Casting a generalized version of a weighted nuclear norm as a graph regularizer was unique in this literature. They also propose an efficient Graph Regularized ALS algorithm to optimize the issue at hand, thus making the proposed method highly scalable.

$$\equiv \min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \left\| \mathcal{W}_{\Omega} \left(\mathbf{R} - \mathbf{A}\mathbf{B}^{\mathrm{T}} \right) \right\|_F^2 + \frac{1}{2} \left\{ \mathrm{Tr} \left(\mathbf{A}^{\mathrm{T}} \mathbf{L}_{(a)} \mathbf{A} \right) + \mathrm{Tr} \left(\mathbf{B}^{\mathrm{T}} \mathbf{L}_{(b)} \mathbf{B} \right) \right\} \tag{3-1}$$

Furthermore, this thesis explores their methodology for tensor decomposition methods using an ALS algorithm. Notably, the algorithm ensures global convergence in linear time.

As observed in [34], it is necessary for nodes with piecewise-smooth or piecewise-constant signals to be penalized differently from a global GLR. The paper provides scalable methods for solving regularized linear systems using a CG method. In adaptation to tensor decomposition, this translates to solving for each factor matrix with a global Laplacian regularizer. Investigating a bias-variance trade-off was essential in determining the regularization parameter's efficacy. As proposed, penalizing local variability instead of global smoothness improves regularization.

3-2-3 Tensors and Graphs for RS

Motivating the need to use tensors and graphs, we now review literature presented exclusively for tensors and graph regularization for RS applications. The sparse literature on the intersection of the three topics forms the basis for this thesis. The available literature is studied, and a unified model is proposed.

A graph regularized low-rank tensor representation is proposed in [41] for feature selection using a graph embedding-based approach. The method uses an ALS method to solve the objective function by evaluating the derivative of the objective for the factor matrix being solved. Various image datasets are used to test and validate the model. The downsampling

of the pixels allows feasibility in computing the inverse of the graph Laplacian. However, this questions the model's scalability for higher-resolution images and would proportionately increase the computational burden. Furthermore, adapting the objective as defined in the paper would not be feasible for RS as the tensor dimensions exponentially grow. The same discussions apply to the model proposed in [54] that incorporates a graph Laplacian-based tensor decomposition method.

A novel method is developed in [55] to decompose a tensor $\mathcal{X}$ coupled with graph data $\mathbf{G}^{(n)}$ as regularizers. The joint analysis method demonstrates the latent structure of related heterogeneous data. Their proposed method consists of a CP decomposition $\hat{\mathcal{X}} = [[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}]]$ for the tensor with an ADMM method to obtain the latent factor matrices.

$$\min_{\mathcal{X}, \left\{\mathbf{X}^{(n)}, \mathbf{d}^{(n)}, \mathbf{G}^{(n)}\right\}_{n=1}^{3}} \left\| \mathcal{X} - [[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}]] \right\|_F^2 + \mu \sum_{n=1}^{3} \left\| \mathbf{G}^{(n)} - \mathbf{X}^{(n)} \operatorname{diag}\left(\mathbf{d}^{(n)}\right) \mathbf{X}^{(n)^{\mathrm{T}}} \right\|_F^2$$

The first term $||\cdot||_F^2$ is the squared Frobenius norm of the model error and the second term accounts for the Symmetric Nonnegative Matrix Factorization model. Nonnegative constraints on the factor matrices $\mathbf{X}^{(n)} > 0$ and diagonal matrices $\mathbf{d}^{(n)} > 0$ are imposed. Their method is compared with the results obtained using CP, nonnegative tensor decomposition methods, and other decomposition algorithms. The proposed algorithms achieve more accurate predictions and perform better than the alternatives mentioned in the paper.

A graph signal processing framework is made in [56] to propose a Tensor Robust PCA on graphs. The model consists of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and graphs built on each tensor n-mode matricization $\mathbf{X}_{(n)},\ n \in [1, 2, 3]$. The graph regularizers $\mathbf{L}^{(n)} = \mathbf{P}^{(n)} \mathbf{\Lambda}^{(n)} \mathbf{P}^{(n)\top}$ with $\mathbf{P}^{(n)} \in \mathbb{R}^{I_n \times R}$, $\mathbf{\Lambda}^{(n)} \in \mathbb{R}^{R \times R}$ incorporate the geometrical structure information with which the data intrinsically is present. They use a low-frequency power concentration method to determine the rank R of the tensor decomposition resulting in $\operatorname{vec}(\mathcal{X}) = \left(\mathbf{P}^1 \otimes \mathbf{P}^2 \otimes \mathbf{P}^3\right) \operatorname{vec}(\mathcal{Z})$ with $\mathcal{Z} \in \mathbb{R}^{R \times R \times R}$. The objective function is given by

$$\min_{\mathbf{Z}_{(n)}} \left\| \mathbf{X}_{(n)} - \mathbf{P}^{(n)} \mathbf{Z}_{(n)} \mathbf{P}^{(i)^{\odot i \neq n \top}} \right\|_1 + \gamma \left\| \sqrt{\mathbf{\Lambda}^{(n)}} \mathbf{Z}_{(n)} \sqrt{\mathbf{\Lambda}^{(i)^{\odot i \neq n}}} \right\|_*$$

with $||\cdot||_1$ is the l_1 norm, $\gamma > 0$ and $\odot i \neq n$ is the Khatri-Rao product of all matrices in $(\cdot)^{(i)^{\odot i \neq n}}$ except n. Using an ALS method to solve the factor matrices, the algorithm is convex and globally converging, overcoming the computational burden of Tensor Robust PCA using a GPU. The proposed model is observed to be robust while improving the model's prediction metrics, showing promising scalability.

3-2-4 Discussions

The key focus of the literature review was aggregating information sources along with tensor decomposition methods and how they aid in making accurate and robust recommendations. The reviewed literature suggests that carrying out data aggregation and incorporating it simultaneously with the factorization technique is advantageous for building RS models. Extracting information from graph structures and utilizing them in conjecture with tensors for the factorization methods was highlighted in the review. Providing better prediction models, the reviewed models highlight how information from graphs can be used as regularization terms while formulating the optimization objective.

The literature review does not cover data fusion methods between tensor and graph structures for RS applications. Discussions on nuclear norm regularized decompositions and their influence on CP rank is limited. The selection of similarity measures between entities in a graph is discussed in graph signal processing methods. The use of graph signal processing is lacking while addressing the regularization of each tensor mode. The importance of GLR for RS model frameworks has not been explored much. Scalability and computational complexity are significant bottlenecks while formulating and solving RS application problems.

This thesis will use data fusion from two data structures, tensors, and graphs, to train RS models. The notion is that data with higher-order relations are fundamental to building accurate and robust RS. We propose a Graph Regularized CP (GRCP) tensor decomposition model framework that will use graph signal processed information and incorporate it into a CP decomposition method. The CP tensor decomposition is constrained by the underlying structural information that a graph can preserve. The constraints will be beneficial in making more accurate and robust recommendations and help identify data trends not restricted to binary relations. The proposed model aims to solve RS frameworks' much-discussed sparsity issues to make recommendations. Finally, the proposed model adds to the existing literature with experiments with tensor nuclear norm and graph Laplacian regularization values on the optimization problem. Another drawback observed in the literature is the scalability of models to larger datasets. Using a linear solver for a system of equations makes the proposed model unique and scalable with reduced computational complexity.

3-3 Graph Regularized CP (GRCP) Tensor Decomposition Model

Formulating the objective function of the problem to be minimized, we begin by combining Equations (1-2) and (3-1) and formulate it as a tensor decomposition problem.

$$\min_{\hat{\mathcal{X}}} \|\mathcal{W}_{\Omega}(\mathcal{T} - \hat{\mathcal{X}})\|^2 + \Psi(\hat{\mathcal{X}}, \mathbf{L}_{(n)})$$

where $\mathcal{T}$ is the ground truth tensor, $\hat{\mathcal{X}}$ is the tensor being reconstructed, and $\Psi(\hat{\mathcal{X}}, \mathbf{L}_{(n)})$ is the regularization function of the tensor $\hat{\mathcal{X}}$ and graph Laplacians $\mathbf{L}_{(n)}$. The regularization function in our proposed model will be the main contribution to existing literature. The function will consist of the nuclear norm and GLR.

The CP decomposition of $\mathcal{T} \approx \hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ with $\hat{\mathcal{X}} = [[\lambda; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots \mathbf{X}^{(d)}]]$ can be bounded such that $||\hat{\mathcal{X}}||^2 < \infty$. However, individual factor matrices $\mathbf{X}^{(n)}$, $n \in [1, d]$ can have norms that are either tending to infinity or zero. It is essential to lower bound the factor matrices such that they do not degenerate while handling sparse tensors, i.e, $||\mathbf{X}^{(n)}||_F \to 0$. The motivation to use the nuclear norm is to obtain well-posed low CP rank solutions to the problem. The nuclear norm lower bounds the factor matrices while solving the objective function [35, 57, 58]. The direct relation between the influence of nuclear norm regularization on CP rank is unknown and shall be investigated in this thesis.

As discussed in Section (2-3-4), regularization using graph Laplacians gives a measure of smoothness to the reconstructed factor matrices. Proven in [53], the GLR is a generalized version of the weighted nuclear norm. This allows us to regularize each factor matrix separately while adopting the ALS algorithm. The graph Laplacians incorporate similarity

relationships that tensors cannot capture. As an additional source of information, they are utilized to solve sparsity issues that RS models encounter.

Introducing the GRCP model framework, we propose the following objective function to minimize,

$$\min_{\mathbf{X}^{(n)}} \frac{1}{2} \left\| \mathcal{W}_{\Omega^{(n)}} \left(\mathcal{T}_{(n)} - \mathbf{X}^{(n)} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \right) \right\|_F^2 + \lambda_n \left\| \mathbf{X}_{(n)} \right\|_* + \frac{\lambda_l}{2} \left\{ \mathrm{Tr} \left({\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{L}_{(n)} \mathbf{X}^{(n)} \right) \right\} \quad (3\text{-}2)$$

where at a given iteration, we minimize $\hat{\mathcal{X}} = [[\lambda; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots \mathbf{X}^{(d)}]]$. This is done by minimizing $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R}, n \in [1, d]$ by keeping all other factor matrices constant. $\mathcal{W}_{\Omega^{(n)}} \in \mathbb{R}^{I_n \times I_1 I_2 \dots I_d}$ is the observable tensor of the ground truth tensor $\mathcal{T}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \dots I_d}$. $\left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}}$ is the Khatri-Rao product of all $\mathbf{X}^{(i)}$ excluding $\mathbf{X}^{(n)}$. The graph Laplacian $\mathbf{L}_{(n)} \in \mathbb{R}^{I_n \times I_n}$ is constructed over every n-mode of the ground truth tensor and is known. λ_n and λ_l are trade-off parameters between the error and regularization terms comprising tensor nuclear norm and graph Laplacian regularization respectively. $|| \cdot ||_F^2$ is the squared Frobenius norm of a matrix, and $\{\mathrm{Tr}\}$ is the matrix trace operator.

The tensor nuclear norm is lower bounded by its tensor n-mode matricization [57], i.e., $\|\mathbf{X}_{(n)}\|_* \leq \|\hat{\mathcal{X}}\|_*$. Thus, we introduce $\| \cdot \|_*$ as the nuclear norm of $\mathbf{X}_{(n)}$ in the objective function being minimized. Writing the nuclear norm as a function of its factor matrices [45],

$$\left\| \mathbf{X}_{(n)} \right\|_* = \min_{\mathbf{X}_{(n)} = \mathbf{X}^{(n)} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}}} \frac{1}{2} \left\{ \left\| \mathbf{X}^{(n)} \right\|_F^2 + \left\| \left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right\|_F^2 \right\} \quad (3\text{-}3)$$

All but one-factor matrix $\mathbf{X}^{(n)}$ is kept constant. The rest of the factor matrices can be neglected from the nuclear norm in the optimization objective function as they do not influence the optimization problem. Rewriting Equation (3-2) by plugging in Equation (3-3) as follows,

$$\min_{\mathbf{X}^{(n)}} \frac{1}{2} \left\| \mathcal{W}_{\Omega^{(n)}} \left(\mathcal{T}_{(n)} - \mathbf{X}^{(n)} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \right) \right\|_F^2 + \frac{\lambda_n}{2} \left\| \mathbf{X}^{(n)} \right\|_F^2 + \frac{\lambda_l}{2} \left\{ \mathrm{Tr} \left({\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{L}_{(n)} \mathbf{X}^{(n)} \right) \right\}$$

As $\|\mathbf{X}^{(n)}\|_{\mathrm{F}}^2 = \mathrm{Tr} \left({\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{I} \mathbf{X}^{(n)} \right)$ and $||\lambda_n \mathbf{I} + \lambda_l \mathbf{L}_{(n)}||_F^2 \leq ||\lambda_n \mathbf{I}||_F^2 + ||\lambda_l \mathbf{L}_{(n)}||_F^2$ using the matrix sub-additive norm property, we sum the norms of a minimization function with $\mathbf{L}^{(n)} = \lambda_n \mathbf{I} + \lambda_l \mathbf{L}_{(n)}$ as follows,

$$\min_{\mathbf{X}^{(n)}} \frac{1}{2} \left\| \mathcal{W}_{\Omega^{(n)}} \left(\mathcal{T}_{(n)} - \mathbf{X}^{(n)} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \right) \right\|_F^2 + \frac{1}{2} ||{\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{L}^{(n)} \mathbf{X}^{(n)}||_F^2 \quad (3\text{-}4)$$

The proposed framework shall be adapted and extended to an RS application in this thesis. The following subsections discuss methods to minimize the objective function using an ALS algorithm with an efficient CG solver. Convergence analysis of the objective function is carried out, ensuring global convergence of the algorithm. Further, model evaluation using standard performance metrics is discussed.

3-3-1 GRCP Alternating Least Squares (ALS) Algorithm

The ALS algorithm minimizes a given factor matrix by keeping the rest fixed at their last update. This is done cyclically until the algorithm meets any of its exit requirements.

$$f_{k+1}\left(\mathbf{X}^{(n)}\right) \triangleq f\left(\mathbf{X}_{k+1}^{(1)}, \ldots, \mathbf{X}_{k+1}^{(n-1)}, \mathbf{X}^{(n)}, \mathbf{X}_{k}^{(n+1)}, \ldots, \mathbf{X}_{k}^{(d)}\right)$$

At iteration $(k+1)$ for $n \in [1, d]$, the objective function to minimize is given by Equation (3-4). Expanding and separating the term to be minimized in Equation (3-4),

$$f(\mathbf{X}^{(n)}) = \min_{\mathbf{X}^{(n)}} \frac{1}{2} \left|\left| \mathbf{X}^{(n)^{\mathrm{T}}} \left[\mathbf{C} + \mathbf{L}^{(n)}\right] \mathbf{X}^{(n)} - 2\mathbf{X}^{(n)^{\mathrm{T}}} \mathbf{Y}^{(n)} \right|\right|_F^2 \tag{3-5}$$

with $\mathbf{Y}^{(n)} = \left(\mathcal{W}_{\Omega^{(n)}} \mathcal{T}_{(n)}\right) \left(\mathbf{X}^{(i)}\right)^{\odot i \neq n} \in \mathbb{R}^{I_n \times R}$, $\mathbf{C} = \left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}} \left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right] \in \mathbb{R}^{R \times R}$.

Refer to Appendix (A-1) for simplification of the objective function. This can be utilized to solve each subproblem by casting Equation (3-5) as a quadratic minimization objective function of the form,

$$g(\mathbf{x}^{(n)}) = \min_{\mathbf{x}^{(n)}} \frac{1}{2} \mathbf{x}^{(n)^{\mathrm{T}}} \mathbf{H}^{(n)} \mathbf{x}^{(n)} - \mathbf{x}^{(n)^{\mathrm{T}}} \mathbf{y}^{(n)} \tag{3-6}$$

with $\mathbf{x^{(n)}} = \mathrm{vec}(\mathbf{X}^{(n)})$ and $\mathbf{y}^{(n)} = \mathrm{vec}(\mathbf{Y}^{(n)}) \in \mathbb{R}^{I_n R}$, the Hessian $\mathbf{H}^{(n)} = \left[\mathbf{C} \otimes \mathbf{I}_{I_n} + \mathbf{I}_R \otimes \mathbf{L}^{(n)}\right] \in \mathbb{R}^{RI_n \times RI_n}$ where $\otimes$ represents the Kronecker product of given matrices. The presence of the GLR requires alterations in the CP-ALS algorithm discussed in Algorithm (1) to solve the minimization problem at each iteration. The Hessian of the given problem is always positive definite at a given iteration $(k+1)$, thus solving a convex optimization problem. The sub-problems of the objective function being solved are convex within the function's domain. Discussions on proof of convexity and algorithm details to solve the objective function are provided in Section (3-3-3) and Appendix (A-2) respectively.

3-3-2 Conjugate Gradient Solver

Solving the given problem as described in Equation (3-6) becomes computationally expensive because Kronecker products of large-sized matrices are present. In addition, the Hessian is not block-diagonal due to the graph Laplacian term. Thus, the bottleneck is computing the Hessian and closed-loop solution of the given quadratic problem.

This can be done efficiently by adapting a matrix-vector product method in a CG solver. Equation (3-6) gives the objective function to be minimized. Taking the gradient to the factor matrix being minimized, the solution of the equation becomes $\nabla g(\mathbf{x}^{(n)}) = \mathbf{H}^{(n)}\mathbf{x}^{(n)} - \mathbf{y}^{(n)} = 0$. Computation of the Hessian matrix-vector product $\mathbf{H}^{(n)}\mathbf{x}^{(n)}$ can be done by using the relation $\mathrm{vec}(\mathbf{A}\mathbf{X}^{(n)}\mathbf{B}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathbf{x}^{(n)}$ given by $\mathbf{H}^{(n)}\mathbf{x}^{(n)} = \mathrm{vec}(\mathbf{X}^{(n)}\mathbf{C} + \mathbf{L}^{(n)}\mathbf{X}^{(n)})$. The same is incorporated in Algorithm (3).

Algorithm 2 GRCP Conjugate Gradient Solver solving for $\hat{\mathbf{X}}^{(n)}$

1: **function** GRCP-ConjGrad($\lambda_n, \lambda_l, \mathbf{L}_{(n)}, \mathbf{C}, \mathbf{X}_0^{(n)}$)
2: initialize $\mathbf{b} = \mathbf{x} = \text{vec}(\mathbf{X}_0^{(n)}) \in \mathbb{R}^{I_n R}$ for $n \in [1, d]$
3: initialize $\mathbf{L}^{(n)} = \lambda_n \mathbf{I} + \lambda_l \mathbf{L}_{(n)} \in \mathbb{R}^{I_n \times I_n}$
4: $\mathbf{r} = \mathbf{b} - \text{Hessian\_Vec}(\mathbf{L}^{(n)}, \mathbf{C}, \mathbf{x})$
5: $\mathbf{p} = \mathbf{r}$
6: **repeat**
7: $\mathbf{r_0} = \mathbf{r}$
8: $\mathbf{v} = \text{Hessian\_Vec}(\mathbf{L}^{(n)}, \mathbf{C}, \mathbf{p})$
9: $\alpha = \frac{\mathbf{r_0}^{\mathrm{T}} \mathbf{r_0}}{\mathbf{p}^{\mathrm{T}} \mathbf{v}}$
10: $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}$
11: $\mathbf{r} = \mathbf{r_0} - \alpha \mathbf{v}$
12: **if** $||\mathbf{r}|| < \text{tol}$ **then**
13: break;
14: **end if**
15: $\beta = \max\left(0, \frac{\mathbf{r}^{\mathrm{T}}(\mathbf{r} - \mathbf{r_0})}{\mathbf{r_0}^{\mathrm{T}} \mathbf{r_0}}\right)$ ▷ Polak-Ribiere method
16: $\mathbf{p} = \mathbf{r} + \beta \mathbf{p}$
17: **until** max. iterations reached
18: **return** $\hat{\mathbf{X}}^{(n)} = \text{unvec}(\hat{\mathbf{x}})$
19: **end function**

Here, the conjugate direction is β, as given in line 15 of Algorithm (2). We use the Polak Ribiere method [59] to calculate β instead of the Fletcher–Reeves method [60]. The Polak Ribiere method has better convergence to a local optimum than the Fletcher-Reeves method, given that line 11 of Algorithm (2) $\rightarrow 0$ as the number of iterations $\rightarrow \infty$. Moreover, it converges equally or faster than the standard steepest descent method. Another criterion for choosing the Polak Ribiere method is that the descent direction is Lipschitz bounded, while it is not definite using the Fletcher Reeves method. The choice of Polak Ribiere over Fletcher Reeves is attributed to numerically stable, and robustness to error-accumulation [61, 62].

Algorithm 3 Hessian Matrix-Vector Mulitplcation

1: **function** Hessian_Vec($\mathbf{L}^{(n)}, \mathbf{C}, \mathbf{x}^{(n)}$)
2: $\mathbf{X}^{(n)} = \text{unvec}(\mathbf{x}^{(n)}) \in \mathbb{R}^{I_n \times R}$
3: **return** $\text{vec}\left(\mathbf{X}^{(n)}\mathbf{C} + \mathbf{L}^{(n)}\mathbf{X}^{(n)}\right)$
4: **end function**

3-3-3 Convergence Analysis

This section discusses the preliminaries required to prove and verify the convergence of iterative optimization algorithms. Solving higher-order decomposition problems as quadratic subproblems favor using linear block Gauss-Seidel iterations. Each iteration is applied to the Hessian matrix to obtain a local solution, ensuring the Hessian matrix is positive and definite. The use of convex real analytic functions with a bounded first derivative allows us to prove the global convergence of the proposed method.

The objective function for each factor matrix minimized at each ALS iteration is given in Equation (3-6). The optimal tensor $\hat{\mathcal{X}} = [[\lambda; \mathbf{X}^{(1)}\mathbf{X}^{(2)} \dots \mathbf{X}^{(d)}]]$ is obtained by evaluating the local optimum of each factor matrix at a given iteration. Given the subproblems are quadratic, the necessary condition for the minimum of the function is the Hessian $\mathbf{H}^{(n)}$ should be positive definite.

Problem Investigation

Investigating the type of problem being solved is important to show the convergence of an algorithm. Here, we prove that the problem minimized contains convex real analytic functions with a bounded first derivative, thus globally converging to a solution.

Let $\mathcal{T}$ be the ground truth tensor with $\text{dom}(\mathcal{T}) \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$. The reconstructed tensor of interest is given by $\hat{\mathcal{X}}$ with $\text{dom}(\hat{\mathcal{X}}) \in \text{dom}(\mathcal{T})$. Considering $\hat{\mathcal{X}}$ to be the best rank approximation of $\mathcal{T}$, the condition $\left\|\mathcal{T} - \hat{\mathcal{X}}\right\| \leq \|\mathcal{E}\|$ applies with $\mathcal{T} = \hat{\mathcal{X}} + \mathcal{E}$, where $\mathcal{E}$ is the residual between the original and reconstructed tensor. This lower bounds our objective function [63]. The following assumptions are made to prove convergence.

Assumption 1. The function g is continuous in its domain, i.e, $\text{dom}(g)$.

Assumption 2. The function g is strongly convex for all $0 < L < \infty$, satisfying the following inequality condition: $g(\mathbf{x}_*^{(n)}) \leq g(\mathbf{x}^{(n)}) + \langle \nabla g(\mathbf{x}^{(n)}), \mathbf{x}_*^{(n)} - \mathbf{x}^{(n)} \rangle + \frac{L}{2}||\mathbf{x}_*^{(n)} - \mathbf{x}^{(n)}||^2$ with Lipschitz constant L, $\forall \mathbf{x}^{(n)}, \mathbf{x}_*^{(n)} \in \mathbf{X}_{(n)}$ where $\mathbf{X}_{(n)}$ is the n-mode matricization of $\hat{\mathcal{X}}$.

Convexity of Objective Function [64]

The function g is strongly convex only if the Hessian $\mathbf{H}^{(n)}$ is positive definite. The Hessian $\mathbf{H}^{(n)}$ is given by $\mathbf{H}^{(n)} = \left[\mathbf{C} \otimes \mathbf{I}_{I_n} + \mathbf{I}_R \otimes \mathbf{L}^{(n)}\right]$. The two terms that are to be investigated are $\mathbf{C}$ and $\mathbf{L}^{(n)}$. The regularization term consists of $\mathbf{L}^{(n)} = \lambda_n \mathbf{I} + \lambda_l \mathbf{L}_{(n)}$. Here, the graph Laplacian $\mathbf{L}_{(n)}$ is a symmetric positive semidefinite matrix. With $\lambda_n, \lambda_l > 0$, the Hessian $\mathbf{H}^{(n)}$ is positive definite. The term $\mathbf{C}_{=} \left[\left(\mathbf{x}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}} \left[\left(\mathbf{x}^{(i)}\right)^{\odot i \neq n}\right] > 0$, i.e, positive definite. Having $\lambda_n, \lambda_l = 0$ still ensures the positive definiteness of the Hessian $\mathbf{H}^{(n)}$ provided $\mathbf{C}$ is full rank.

Lipschitz Boundedness [65]

Let $g : \text{dom}(g) \to \mathbb{R}^{I_n \times R}$, $\text{dom}(g) \in \text{dom}(\mathcal{T})$. g belongs to the C^∞ class of functions where the function is differentiable for all degrees of differentiation. The function g is locally Lipschitz if

$$g_0 : \{\mathbf{x}^{(n)}, \mathbf{x}_0^{(n)} \in \text{dom}(g) : \left\|\mathbf{x}^{(n)} - \mathbf{x}_0^{(n)}\right\| < \epsilon\} \to \text{dom}(\mathcal{T})$$

for some $\epsilon > 0$. Given the exact solution of the minimization function is not possible to obtain and is lower bounded, the gradient function $\left[\partial g(\mathbf{x}^{(n)}) / \partial \mathbf{x}^{(n)}\right]$ is locally Lipschitz continuous and bounded within $\text{dom}(g)$, i.e., $\left\|\left[\frac{\partial g(\mathbf{x}^{(n)})}{\partial \mathbf{x}^{(n)}}\right]\right\| \leq B$ for $B > 0$, $\mathbf{x}^{(n)} \in \text{dom}(g)$ with Lipschitz constant $L = B$. Further proof has been provided in Appendix (A-3-1).

Kurdyka-Lojasiewicz (KL) Inequality

A function g satisfies the Kurdyka-Lojasiewicz (KL) property at a stationary point $\mathbf{x}_*^{(n)} \in dom(\partial g)$, if there exists $\theta \in [0, 1)$ such that

$$\frac{|g(\mathbf{x}^{(n)}) - g(\mathbf{x}_*^{(n)})|^\theta}{\text{dist}(\mathbf{0}, \partial g(\mathbf{x}^{(n)}))}$$

is bounded around $\mathbf{x}_*^{(n)}$, i.e, in a certain neighborhood $\mathcal{U}$ of $\mathbf{x}_*^{(n)}$, there exists $\phi(s) = cs^{1-\theta}$ for $c \geq 0$ and $\theta \in [0, 1)$ such that the KL inequality below holds:

$$\phi'(|g(\mathbf{x}^{(n)}) - g(\mathbf{x}_*^{(n)})| \, \text{dist}(\mathbf{0}, \partial g(\mathbf{x}^{(n)})) \geq 1 \tag{3-7}$$

The property was first introduced for real analytic functions, which later was extended to nonsmooth subanalytic functions [66]. A function is said to be real-analytic α if the function can be represented by a convergent power series on some interval of positive radius centered at α.

$$g(\mathbf{x}^{(n)}) = \sum_{j=0}^{\infty} a_j (\mathbf{x}^{(n)} - \alpha)^j \tag{3-8}$$

The objective function is said to be real analytic on any $V \subseteq dom(g)$ if it is real analytic at each $\alpha \in V$. Proven in Appendix (A-3-2), we show that the g is a sum of real analytic functions and, thus, satisfies the KL inequality as given in Equation (3-7). This also verifies that the objective function is smooth throughout the given domain.

Global Convergence

Global convergence of minimizing an objective function is crucial to obtaining the optimal point after optimization. Having proven $g(\mathbf{x}^{(n)})$ satisfies the KL inequality at $\mathbf{x}_*^{(n)}$ and $\nabla g(\mathbf{x}^{(n)})$ is Lipschitz continuous in $dom(g)$, global convergence and convergence rate of the problem are given by the following theorems. Proofs for the theorems are as provided in [37].

Theorem 1. ***[37] Global Convergence:*** *The sequence $\{\mathbf{x}_k^{(n)}\}$ converges to a critical point $\mathbf{x}_*^{(n)}$ where g satisfies the KL inequality under the assumptions stated in [Lemma 2.6, p1770 of [37]]. The sequence $\{\mathbf{x}_k^{(n)}\}$ is said to converge to $\mathbf{x}_*^{(n)}$, which is a critical point of g.*

Theorem 2. ***[37] Convergence Rate****: Under the assumptions of [Lemma 2.6, p1770 of [37]] and that $\mathbf{x}_k^{(n)}$ converges to a critical point $\mathbf{x}_*^{(n)}$ at which g satisfies the KL inequality with $\phi(s) = cs^{1-\theta}$ for $c > 0$ and $\theta \in [0, 1)$, the following hold:*

- *If $\theta = 0$, $\mathbf{x}_k^{(n)}$ converges to $\mathbf{x}_*^{(n)}$ in finitely many iterations (finite convergence).*
- *If $\theta \in (0, \frac{1}{2}]$, $||\mathbf{x}_k^{(n)} - \mathbf{x}_*^{(n)}|| \leq C\tau^k$ for all $k \geq k_0$, for certain $k_0 > 0$, $C > 0$, $\tau \in [0, 1)$ (linear convergence).*
- *If $\theta \in [\frac{1}{2}, 1)$, $||\mathbf{x}_k^{(n)} - \mathbf{x}_*^{(n)}|| \leq Ck^{-(1-\theta)/(2\theta-1)}$ for all $k \geq k_0$, for certain $k_0 > 0$, $C > 0$ (sublinear convergence).*

3-3-4 Performance Evaluation Metrics

Evaluating the proposed model on widely accepted metrics such as Normalized Mean Square Error (NMSE) and Percentage of Fit (POF). These metrics can evaluate the proposed model framework by generalizing a tensor completion problem to a prediction problem.

Normalized Mean Square Error (NMSE)

The NMSE for tensor completion models is evaluated on the observable space of the tensors. This is given as,

$$\text{NMSE} = \frac{||\mathcal{W}_\Omega(\mathcal{X} - \hat{\mathcal{X}})||}{||\mathcal{X}||}$$

Percentage of Fit (POF)

Relative Reconstruction Error is a metric that evaluates the relative error between the reconstructed tensor to the original tensor.

$$\text{RRE} = \frac{||\mathcal{X} - \hat{\mathcal{X}}||}{||\mathcal{X}||}$$

In this thesis, we shall be evaluating the Percentage of Fitness (POF) given by,

$$\text{POF } (\%) = 1 - \text{RRE}$$

Computational Complexity Analysis

Evaluating computational complexity is important for algorithms to ensure they are scalable. We evaluate the number of required operations for the CG solver at each iteration as given in Algorithm (2).

The CG solver requires the matrices $\mathbf{L_{(n)}} \in \mathbb{R}^{I_n \times I_n}$, $\mathbf{C} \in \mathbb{R}^{R \times R}$ and $\mathbf{X_0}^{(n)} \in \mathbb{R}^{I_n \times R}$. The graph Laplacian $\mathbf{L_{(n)}}$ is computed and stored from a predefined kNN graph.

- The MTTKRP is a computationally intensive iteration with computational complexity $\mathcal{O}(|\Omega|R)$. The MTTKRP generates an initial point $\mathbf{X_0}^{(n)}$ for the CG algorithm.
- $\mathbf{C} = \left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}} \left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]$ can be precomputed in $\mathcal{O}(R^3)$ operations at each ALS iteration.
- The Hessian_Vec function as defined in Algorithm (3) is the main computational bottleneck of the solver. The cost of computing $\left[\mathbf{X}^{(n)}\mathbf{C} + \mathbf{L}^{(n)}\mathbf{X}^{(n)}\right]$ is $\mathcal{O}(I_n R^2 + \text{nnz}(\mathbf{L}^{(n)})R)$. nnz evaluates the total number of non-zeros present in the computed graph Laplacian.

Given a single ALS iteration optimizing a factor matrix, the total computational cost becomes,

$$\mathcal{O}(|\Omega|R) + \mathcal{O}(R^3) + \texttt{maxCGiter}(\mathcal{O}(I_n R^2 + \text{nnz}(\mathbf{L}^{(n)})R))$$

where `maxCGiter` is the maximum number of iterations set for the CG solver.

3-4 GRCP for Recommender Systems

This section adapts the proposed generic model framework specific to an RS application. A third-order user-item-time tensor is constructed. The set of users are represented by $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{I_1}]$, set of items by $\mathbf{I} = [\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_{I_2}]$ and time instance during which the rating was provided by $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{I_3}]$. The ground truth tensor is given as $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Beginning with the objective function of minimizing,

$$\underset{\mathbf{U_m}, \mathbf{I_m}, \mathbf{T_m}}{\text{minimize}} \frac{1}{2} \| \mathcal{W}_\Omega \left(\mathcal{T} - \hat{\mathcal{X}} \right) \|^2 + \frac{1}{2} ||{\mathbf{U_m}}^{\mathrm{T}} \mathbf{L}^{(u)} \mathbf{U_m}||_F^2 + \frac{1}{2} ||{\mathbf{I_m}}^{\mathrm{T}} \mathbf{L}^{(i)} \mathbf{I_m}||_F^2 + \frac{1}{2} ||{\mathbf{T_m}}^{\mathrm{T}} \mathbf{L}^{(t)} \mathbf{T_m}||_F^2$$

where the optimal tensor $\hat{\mathcal{X}} \approx [[\lambda_\mathbf{m}; \mathbf{U_m} \mathbf{I_m} \mathbf{T_m}]] \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. $\mathbf{U_m} \in \mathbb{R}^{I_1 \times R}$, $\mathbf{I_m} \in \mathbb{R}^{I_2 \times R}$ and $\mathbf{T_m} \in \mathbb{R}^{I_3 \times R}$ are factor matrices of the CP tensor decomposition, having a normalizing coefficient $\lambda_\mathbf{m} \in \mathbb{R}^R$ and CP rank R.

Each factor matrix is regularized with their respective graph Laplacians; namely, $\mathbf{L}^{(u)} = \lambda_u \mathbf{I} + \lambda_l \mathbf{L}_{(u)} \in \mathbb{R}^{I_1 \times I_1}$ the graph Laplacian between a set of users, $\mathbf{L}^{(i)} = \lambda_i \mathbf{I} + \lambda_l \mathbf{L}_{(i)} \in \mathbb{R}^{I_2 \times I_2}$ the graph Laplacian between a set of items and $\mathbf{L}^{(t)} = \lambda_t \mathbf{I} + \lambda_l \mathbf{L}_{(t)} \in \mathbb{R}^{I_3 \times I_3}$ the graph Laplacian between time instants.

3-5 Discussions

The use of tensors and graphs has been well motivated by literature, highlighting the advantages and drawbacks of available models. Overall, its seen to be beneficial in using structures to accommodate data with higher-order relations and aided information sources to build RS models.

The proposed GRCP model framework is a generic model that, in this case, is applied to an RS. The same model can be adapted to other applications such as image processing and computer vision, data classification, and machine learning tasks. The choice of an explanatory tensor decomposition method in combination with GLRs has been done before and is not new. Regularizing each factor matrix with the combination of the nuclear norm and GLR has not been covered adequately in the literature. The proposed framework is modeled accordingly to provide improved prediction metrics and scalability with reduced computational effort.

The CP-ALS method is globally convergent in the domain of the proposed objective function. Problems related to scaling and matrix degeneracy have been addressed by incorporating low-rank constraints and normalizing the factor matrices after every ALS iteration. The CG solver shows scalability from its computational complexity. Quicker convergence of a factor matrix to its local optimum is ensured by incorporating the Polak Ribere method. To model the data correctly, one must choose tensor dimensions and the Laplacian weighting function. Tuneable free parameters are the CP model rank R, regularizer coefficients λ_n and λ_l, and k-nearest neighbors according to the application.

The proposition paves the way to building multi-context-aware RS models. The projection of data on a lower dimensional space for interpretability and sparsity of data helps compute fast tensor, matrix, and vector operations. The motivation to build an efficient and robust RS framework to help make better recommendations is theoretically feasible and will be tested on various datasets.

3-6 Conclusion

This chapter reviews the literature required to propose a unified model in this thesis. We propose a GRCP tensor decomposition model framework. Solved using an ALS algorithm incorporating a CG solver, we show that the proposed model is globally convergent, with each factor matrix being locally optimum at a given iteration. Evaluation metrics of the algorithm for accuracy and computational complexity have been carried out, showing convergence in linear time.

Chapter 4

Model Validation using Synthetic Dataset

4-1 Motivation

Testing and validation of a proposed model are necessary. This chapter will evaluate the proposed GRCP model framework on a synthetic dataset. The motivation to use this dataset is to evaluate the model on data with similar tensor dimensions that will later be used to validate an RS model. Observations on the synthetic dataset will be made with varying CP ranks and Laplacian regularization.

Literature is ample for regularized tensor decompositions for images. However, they have been observed not to scale well. This is due to using decomposition methods that are computationally intensive or gradient-based methods that are sensitive to initializations. Thus, we shall use an image as our synthetic data set and evaluate the model for image representation.

4-2 Experiment Setup

Setting up the experiment, we select the image `baboon.jpg` as shown in Figure (4-1). The image is an RGB image of pixel dimensions 512×512. The image is loaded onto MATLAB as a tensor $\mathcal{I} \in \mathbb{R}^{I_1 \times I_2 \times I_3} \in \mathbb{R}^{512 \times 512 \times 3}$ of `class(I) = 'double'`. The third dimension of the tensor represents the RGB components of the image. The Laplacians are obtained from a kNN graph built on each tensor mode $\mathbf{I}_{(n)} \in \mathbb{R}^{I_n \times I_n}$ with a Gaussian edge weight kernel defined by,

$$\mathbf{W}_{ij} = \begin{cases} \exp\left(-\frac{\|(x_i - x_j)\|_2^2}{\sigma}\right) & \text{if } x_j \text{ is connected to } x_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4-1}$$

where x_i and x_j are the i^{th} and j^{th} entities in the graph being built and σ is the variance in the tensor mode given by $\sigma = \frac{||x_i - x_j||_2}{N}$, with $N \in \mathbb{R}^{I_n}$. The noise added to the original image

is given by $\mathcal{I}_n = \mathcal{I} + \xi$ where, $\xi \sim (0, \sigma(\mathcal{I}))$ is Gaussian noise with zero mean and standard deviation $\sigma(\mathcal{I})$.

Using the GRCP model framework, the image is reconstructed, and the influence of regularization is studied. The image is also added with noise, and the recovery of the original image using the GRCP model is evaluated. This is done by ensuring that the Laplacians are built on the image with the added noise.

The computations will be carried out on a MacBook Pro 2017 using an Intel Core 64-bit i5 Dual-Core Processor running at 2.3 GHz with 8GB DDR3 RAM. The software used is MATLAB R2021b, and the GSP Toolbox [67] for creating the graph Laplacians.

Figure 4-1: Image baboon.jpg

4-2-1 Algorithm Initialization

Initialization of the factor matrices and parameters for the ALS algorithm and the CG solver is important for convergence and accurate reconstruction. Experiments are carried out to study the effect of the number of kNN, CP rank R, regularization λ_l, and Signal-to-Noise ratio (SNR) added. The model is trained for a combination of these values, and each hyperparameter's effect on image reconstruction and computational time is measured.

Of the different initialization methods discussed in [1] and [68], the factor matrices for the ALS algorithm are initialized with the leading R left singular values of the given tensor's n-mode matricization. If the tensor dimension is smaller than the chosen CP rank $I_n < R$, then the factor matrix is assigned random values between $(0, 1)$. The algorithms used are as given in (2), (3), and (4). Table (4-1) shows the parameters used.

Table 4-1: Experiment Setup Parameters: Synthetic Dataset

Parameter	Values
ALS Maximum Iterations (`maxALSIter`)	250
ALS tolerance (`tolALS`)	$2.22e^{-16}$
Conjugate Gradient Maximum Iterations (`maxCGIter`)	30
Conjugate Gradient tolerance (`tolCG`)	$1e^{-12}$
kNN members (`kNN`)	[1 1 1; 1 1 2; 2 2 2]
CP Rank (`R`)	[40 60 80 100 120 140]
Laplacian Regularization (`LReg`)	$[0, 0.1, 0.01, 0.001]$
Added Noise (dB) (`SNR`)	[0 0.1 0.5 1 4 7]

4-3 Results and Discussions

We analyze and report the findings from the output of the GRCP model. The influence of CP rank, Laplacian regularization, and noise in terms of SNR (dB) on image reconstruction and representation are recorded.

4-3-1 Influence of CP Rank

Identifying the CP rank of the decomposition is important for the correct representation of the reconstructed image. Minimizing the error between the original image tensor and the graph regularized tensor should be ensured. We choose the original noiseless `baboon.jpg`, perform the CP decomposition varying the CP ranks, and record the obtained NMSEs.

Figure 4-2: Effect of varying CP Rank on Image Reconstruction with kNN = [1 1 1]. Top Row: $\lambda_l = 0$, Bottom Row: $\lambda_l = 0.001$, Left to Right: CP Rank R = [20 40 60 80 100 120 140]

Table (4-2) highlights the NMSEs obtained for various CP ranks. Consistent image reconstruction and representation are observed when using lower CP ranks. The average NMSE is observed to increase as the CP rank is increased. Increasing the CP rank results in overfitting of the data and possibly noise. The algorithm's reconstruction for higher CP rank

decompositions is better with regularization, as seen in Figure (4-2). During reconstruction, regularization is observed to preserve the relationships between neighboring pixels in the data.

CP Rank	40	60	80	100	120	140
Best NMSE	0.101	0.096	0.089	0.113	0.136	0.143
Worst NMSE	1.987	4.162	6.809	7.130	8.262	8.914
Average NMSE	0.223	0.287	0.410	0.718	1.111	1.720

Table 4-2: Aggregate NMSE for varying CP-Rank

Elaborating on Table (4-2), experiments with varying kNN members and regularization values for a given CP rank are carried out. The values recorded indicate which CP rank is most suitable for image reconstruction. The average NMSEs indicate the NMSE that a particular CP rank would achieve by varying the other hyperparameters.

The choice of CP rank is an NP-hard problem and is done based on conducted experiments. Image reconstruction and average NMSEs are similar and consistent for CP ranks between 40 and 60. This is the case with and without regularization. Choosing a CP rank of 40 would benefit the algorithm computationally. However, we record better NMSE values using a CP rank of 60. Using CP ranks greater than 60 is not recommended, with the model being observed to overfit the data with an increase in NMSE values. The next section shall discuss in detail the influence of the number of kNN members and regularization for images reconstructed with a CP rank of 60.

4-3-2 Influence of Regularization

Regularization is important for both image representation and retrieving an image that has been corrupted by noise. We first check how varying the number of kNN members and regularization affects image reconstruction on the original noiseless `baboon.jpg`.

Figure (4-3) shows that the choice of kNN members is important for reconstructing the image. Image reconstruction is best for kNN = [1, 1, 1]. Increasing the number of kNN members does not provide accurate reconstruction results. This is important to note as the capturing variation of pixel values over neighboring pixels is preferred and effective over a patch of pixels in an image.

Table (4-3) highlights the NMSEs obtained for various regularization λ_l values. The table records NMSEs for a given regularization value while varying kNN members and CP rank. Compared to $\lambda_l = 0$, the average NMSEs highlight regularization $\lambda_l > 0$ to benefit image reconstruction. As recorded, the average NMSE is the least for $\lambda_l = 0.001$.

Preserving any structural information in the data is carried out with the regularization term. The Laplacian, also known as a spatial high pass filter, is found to detect and preserve the edges in the image. Fundamental in building kernels/masks in Digital Image Processing, the same is applicable here while constructing the Laplacians for regularization [69]. Computationally, the average runtimes with and without regularization are similar, as given in Table (B-1).

Figure 4-3: Effect of varying kNN members and regularization on Image Reconstruction with CP rank = 60. Top to Bottom: kNN members [1 1 1], [1 1 2] and [2 2 2], Left to Right: Laplacian Regularization $\lambda_l = [0, 0.1, 0.01, 0.001]$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**
Best NMSE	0.096	0.097	0.089	0.096
Worst NMSE	7.381	8.734	8.914	8.262
Average NMSE	0.785	0.773	0.764	0.658

Table 4-3: Aggregate NMSE for varying regularization λ_l

The next section shall discuss the influence of adding a certain SNR (dB) noise to the image attempting to perform a denoising operation with kNN = [1, 1, 1] and $\lambda_l = 0.001$.

4-3-3 Influence of Noise

Having studied the effects of varying CP ranks and regularization values λ_l, we investigate the effect of adding a particular SNR (dB) of noise to the image. Attempts to reconstruct the image with minimal NMSE are made. The Laplacians are constructed on the prefiltered image with given noise SNR (dB). Highlighted in [70], Laplacians built on prefiltered images are robust to noise provided the weighting kernel is as given in (4-1). The experiment is repeated ten times with CP rank $R = 60$, kNN = [1, 1, 1] and $\lambda_l = 0.001$.

Figure (4-4) shows the reconstructed image and the corresponding noisy image. The Laplacian is responsible for filtering high-frequency data and providing smoothness during reconstruction. The applied noise within a band of SNR = (0.1 - 1) dB is filtered well, as observed by the average NMSE values shown in Table (4-4). It does not help the decomposition method if the image contains no or low-frequency noise in the data. The NMSEs are consistent when the criterion for the noise to be bounded is met.

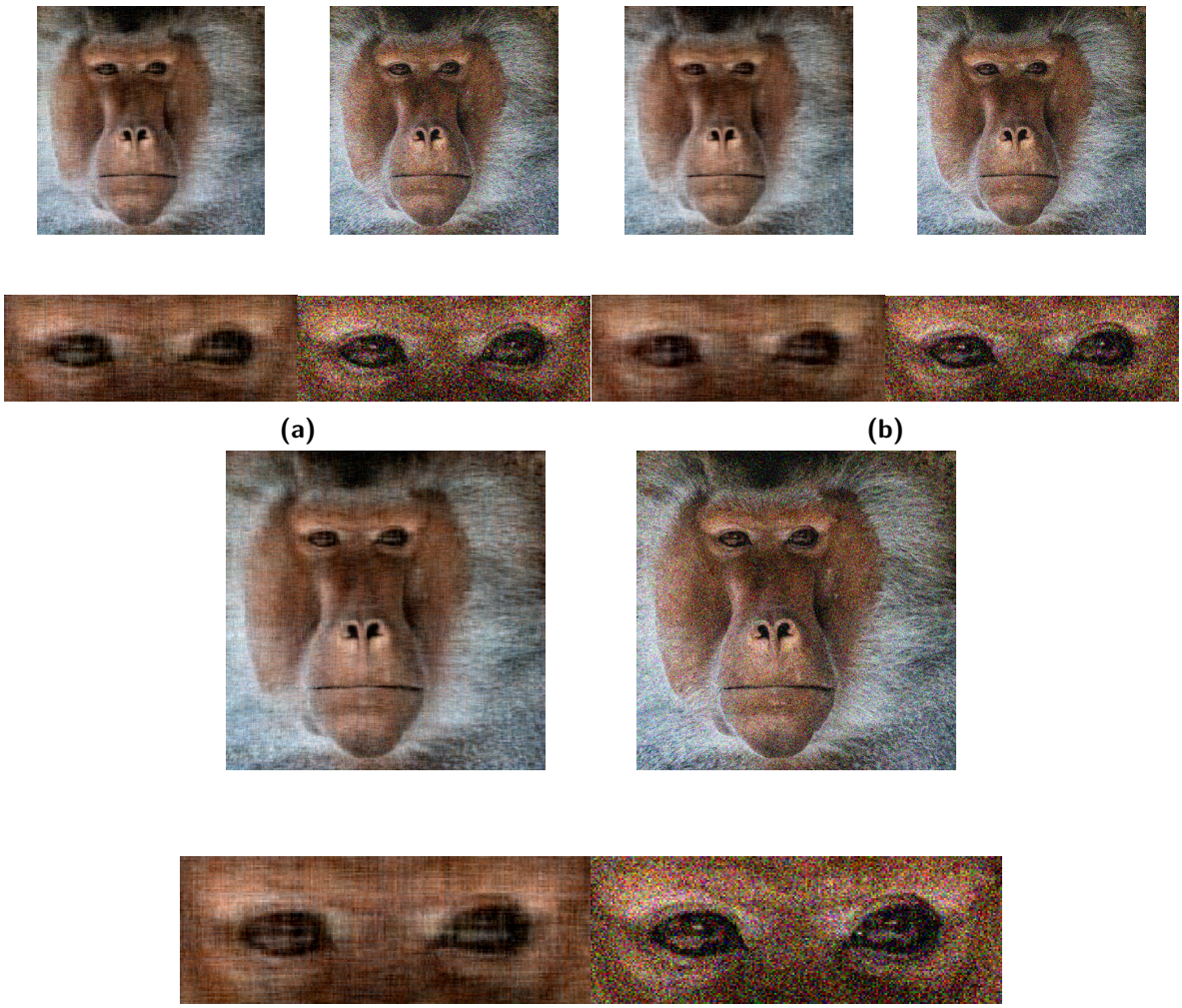

Figure 4-4: Effect of varying with CP rank = 40, kNN = [1 1 1] and $\lambda_l = 0.001$. (a) SNR = 0.1 dB, (b) SNR = 0.5 dB, (c) SNR = 1 dB

SNR (dB)	**0**	**0.1**	**0.5**	**1**	**4**	**7**
Best NMSE	0.096	0.144	0.139	0.135	0.112	0.099
Worst NMSE	8.914	0.236	0.226	0.212	8.130	8.684
Average NMSE	1.732	0.191	0.183	0.173	0.514	0.883

Table 4-4: Aggregate NMSE for varying SNR (dB)

4-3-4 Performance Evaluation Metrics

Having studied the effect of the hyperparameters on image reconstruction, we evaluate the model on the discussed Performance Evaluation Metrics.

Normalized Mean Squared Error (NMSE)

NMSE is a good indicator of the quality of image reconstruction from the decomposition. Iterating the algorithm for the same experimental setup multiple times is important to obtain an average and consistent NMSE.

Figure (4-5) indicates well the effect of regularization on NMSE with varying CP rank R and SNR (dB). Across all images, we can observe that the model does not perform well in reconstructing the image for low and high CP ranks, given by higher NMSE values.

Having no regularization is observed to result in higher NMSE values. Regularization is, therefore, necessary. Having higher regularization values allows higher CP ranks to be utilized to provide lower NMSEs. SNR of above 1 dB is seen to have increased NMSE values. As discussed, the range of noise the model gives the best image representations is from (0.1 - 1) dB.

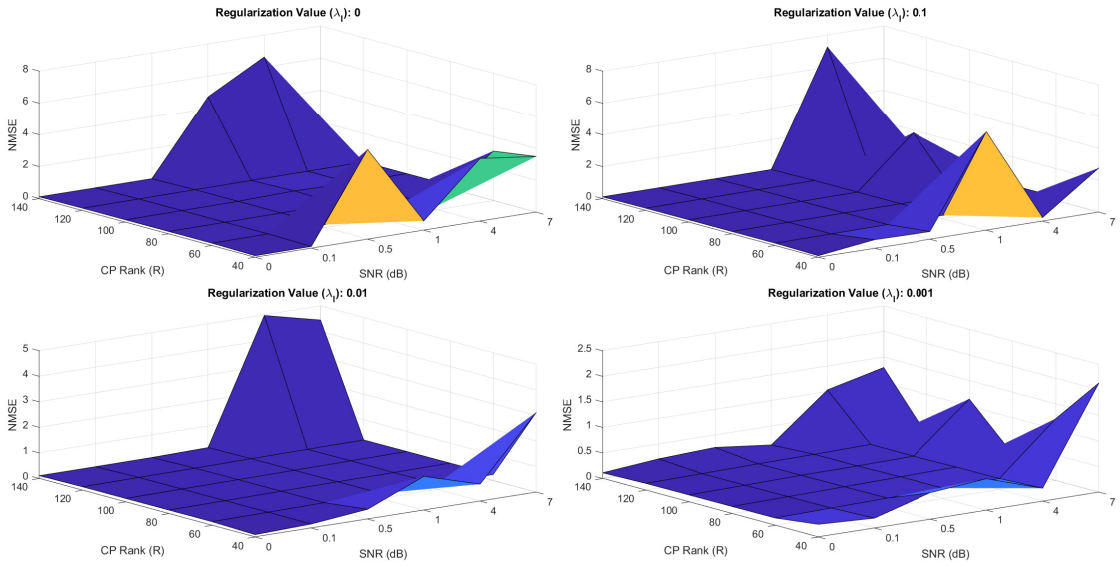

Figure 4-5: Influence of NMSE with varying CP Rank, noise SNR (dB), and regularization λ_l

Algorithm Run Times

The algorithm is timed and aggregated over ten iterations. Dependent on the choice of CP rank, we evaluate its effect on the algorithm's run time. With increasing CP rank R, it is noticed that the computational time increases linearly with R as given in Table (4-5). This is beneficial for the algorithm while solving larger-scale problems.

The average runtimes for varying regularization and noise do not vary for different values of λ_l and SNR (dB). They are observed to not influence the algorithm's run time as given in Tables (B-1) and (B-2).

CP Rank	40	60	80	100	120	140
Min Run Time (s)	1.93	6.58	10.86	16.33	20.18	31.79
Max Run Time (s)	65.82	64.99	116.25	168.66	206.29	256.07
Average Run Time (s)	32.98	44.62	58.15	69.32	88.65	99.16

Table 4-5: Aggregate runtimes for varying CP-Rank

Code Profiling

Comparing run times for varying hyperparameters is representative of their influence on the algorithm. Understanding the time taken by each code segment as a ratio to the total time taken to run the code highlights the code's computational bottlenecks. This shall be done by profiling the code and measuring the time taken by each function.

The code is profiled for two cases. The first setup is with parameters taking the most time to solve. Looking at the average run times, we choose a CP rank $R = 140$, graph kNN members = [1 1 1], regularization $\lambda_l = 0$ with SNR = 0.5 dB. Refer to Equations (3-5) and (3-6) for the matrices being evaluated.

Evaluating Function	Dimensions	Self Time Taken (%)
MTTKRP	$\mathbb{R}^{I_n \times R}$	41.5%
Hessian_Vec (3)	$\mathbb{R}^{I_n R}$	29.7%
Khatri-Rao Product	$\odot I_i = \prod_{i \neq n}^{d} I_i$	7.4%
CG Solver (2)	$\mathbb{R}^{I_n R}$	6.9%
ALS Algorithm (4)	$\mathbb{R}^{I_1 \times I_2 \ldots \times I_d}$	5.1%
Other function calls	-	9.4%

Table 4-6: Code Profile for Worst Case Algorithm Run Time with Total Average Time = 200.62s

The second setup is evaluated on the model providing the best average NMSE results. Here, the chosen model parameters are CP rank $R = 60$, graph kNN members = [1 1 1], regularization $\lambda_l = 0.001$ with SNR = 0.1 dB.

Evaluating Function	Dimensions	Self Time Taken (%)
MTTKRP	$\mathbb{R}^{I_n \times R}$	29.3%
Hessian_Vec	$\mathbb{R}^{I_n R}$	14.4%
Khatri-Rao Product	$\odot I_i = \prod_{i \neq n}^{d} I_i$	8.6%
CG Solver	$\mathbb{R}^{I_n R}$	7.3%
ALS Algorithm	$\mathbb{R}^{I_1 \times I_2 \ldots \times I_d}$	6.9%
Other function calls	-	33.5%

Table 4-7: Code Profile for BEST NMSE Algorithm Run Time with Total Average Time = 7.3s

The MTTKRP and Hessian_Vec functions contain matrix multiplications. An increase in matrix dimensions makes it computationally intensive for programs running on a CPU. Although scaling well with an increase in matrix dimensions, these functions are the identified bottlenecks in the code.

4-4 Conclusions

In this chapter, we extend the application of the proposed GRCP model framework to a synthetic dataset containing an image. The variation of CP rank R, graph Laplacian regularization λ_l, and kNN members were observed and discussed. The given `baboon.jpg` image was reconstructed with no noise to identify the GRCP model's hyperparameters. Later, noise was added to the image, and attempts to recover the original image from the noisy data were made. NMSE values of graph regularized methods showed more consistent image representation than the ones without regularization. The graph Laplacian regularization has highlighted use cases for image denoising. Given that the synthetic dataset is to evaluate the GRCP model and observe the effect of varying model hyperparameters, benchmarking studies were not necessary to be carried out.

Chapter 5

Model Validation using MovieLens Dataset

5-1 Motivation

The major contribution of this thesis is to use the proposed framework to build an RS model. This chapter will evaluate the Graph Regularized CP (GRCP) tensor decomposition model framework on the MovieLens dataset to build an RS. Numerical experiments are carried out on the given data set. The effect of varying CP rank R and graph Laplacian regularization λ_l will be observed on the dataset used. The model is compared for different values of nuclear norm regularization λ_n to evaluate its effect on CP rank. Performance metrics, including NMSE, Percentage of Fit (POF), and computational run times as defined in Section (3-3-4), are recorded for the combination of hyperparameters.

5-2 Experiment Setup

The history of the MovieLens dataset is described in detail in [71]. The dataset used in this thesis is the ML100K dataset, with 100,000 ratings given by 943 users for 1682 movies. The values of the ratings provided are on a 0-5 rating scale, with 0 for items that have not been rated by a user. The ratings are dated from September 1997 to April 1998. The number of known entries (or) sampling varies to see the model's prediction accuracy and influence on reconstruction. As defined in Equation (5-1), $|\Omega|$ is the number of known entries, and $I_1 I_2 \dots I_d$ are the dimensions of the rating tensor being constructed. Representing the data as a matrix of dimensions $\mathbb{R}^{943 \times 1682}$, the sampling or data density is 6.37%.

$$\rho = \frac{|\Omega|}{I_1 I_2 \dots I_d} \tag{5-1}$$

Setting up the experiment, the ML100K dataset is loaded onto MATLAB as a User-Item-Time tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3} \in \mathbb{R}^{943 \times 1682 \times T_m}$ of `class(T) = 'double'`. We construct tensors with

different sampling rates for our numerical simulations. The tensor's third mode $\mathbf{T}_{(3)} \in \mathbb{R}^{T_m}$, represented as time, will be the year or month of rating. When represented with the rating year, the tensor dimensions are $\mathbb{R}^{943\times1682\times2}$ with the time dimensions representing the years 1997 and 1998. Similarly, when represented with the month of rating, its dimensions are $\mathbb{R}^{943\times1682\times8}$ with the time dimensions representing months from September through to April.

The Laplacians are obtained from a kNN graph built on each tensor mode $\mathbf{T}_{(n)} \in \mathbb{R}^{I_n\times I_n}$ with the edge weight kernel defined by,

$$\mathbf{W}_{ij} = \begin{cases} \frac{\|(x_i - x_j)\|_2^2}{\sigma} & \text{if } x_j \text{ is connected to } x_i, \\ 0 & \text{otherwise.} \end{cases} \tag{5-2}$$

where x_i and x_j are the i^{th} and j^{th} entities in the graph being built and σ is the variance given by $\sigma = \frac{||x_i - x_j||_2}{N}$, with $N \in \mathbb{R}^{I_n}$. The weighting kernel used here evaluates the Pearson Correlation Coefficient matrix between the graph's x_i and x_j entities.

The data containing 100,000 ratings are split with an 80-20 ratio for training and validating the model. With tensor dimensions $\mathcal{T} \in \mathbb{R}^{943\times1682\times2}$, the training sampling rate is $\rho = 2.5\%$ while with tensor dimensions $\mathcal{T} \in \mathbb{R}^{943\times1682\times8}$, the training sampling rate is $\rho = 0.6\%$. The GRCP model framework is trained on the training dataset for given tensor dimensions. The influence of sampling, CP rank, and regularization on model prediction accuracy is studied.

The computations on the ML100K dataset are carried out on the Delft Blue Supercomputer using Intel XEON E5-6248R 24C 64-bit 24 Core Processor running at 3.0GHz with 64GB DDR4 RAM [72]. The software used is MATLAB R2021b, and the GSP Toolbox [67] for creating the graph Laplacians.

5-2-1 Algorithm Initialization

Initialization of the factor matrices and parameters for the ALS algorithm and the CG solver are studied better for an RS application. By default, MATLAB uses the `'double'` data type with a 64-bit representation for variable initializations. This is unnecessary and can be initialized as `'single'` data types with a 32-bit representation for the factor matrices and graph Laplacians. [1] This accelerates the algorithm by improving the memory throughput time and provides values NMSEs and POF similar to variables initialized as `'double'` datatype.

The factor matrices for the ALS algorithm are initialized with the leading R left singular values. If the tensor dimension is smaller than the chosen CP rank $I_n < R$, then the factor matrix is assigned values randomly between $(0, 1)$. The algorithms used are as given in (4), (2), and (3). Table (5-1) shows the parameters used.

First, the simulations are run for varying kNN members used to construct the graph. Varying the number of kNN is observed not to affect the prediction accuracy of the obtained models. Sparsity in the data could be accountable for this observation. Thus, we fix the number of kNN members with a rule of thumb `kNN(n)` $\sim \sqrt{I_n}$. CP rank R, regularizations λ_n, and λ_l

[1] Performing arithmetic operations on single floating point datatype variables provides values of precision that are acceptable to building RS models. Both range and precision required are met as given in Floating Point Numbers

are simulated for the two tensor setups. The model is trained for a combination of these values, and each hyperparameter's influence on model accuracy and computational time is investigated.

Table 5-1: Experiment Setup Parameters: MovieLens ML100K

Parameter	Values
ALS Maximum Iterations (`maxALSIter`)	200
ALS tolerance (`tolALS`)	$2.22e^{-16}$
Conjugate Gradient Maximum Iterations (`maxCGIter`)	30
Conjugate Gradient tolerance (`tolCG`)	$1e^{-12}$
Time dimension of Tensor (`cT`)	['YY', 'MM'] $\in [\mathbb{R}^2, \mathbb{R}^8]$
kNN members (`kNN`) for `cT = 'YY'`	[30 40 1]
kNN members (`kNN`) for `cT = 'MM'`	[30 40 4]
CP Ranks (`R`) for for `cT = 'YY'`	[5 7 10 12 15 20 25]
CP Ranks (`R`) for for `cT = 'MM'`	[20 30 33 36 40 50]
Laplacian Regularization (`LReg`)	$[0, 0.1, 0.01, 0.001]$

5-3 Results and Discussions

The GRCP framework is evaluated for building an RS model with the given data. The influence of CP rank R, nuclear norm, and graph Laplacian regularization λ_n and λ_l are recorded and analyzed on NMSE and computational run times.

5-3-1 Influence of CP Rank

Varying the sampling of the data, the two tensor models are solved using the GRCP model framework. Identification of the CP rank, being an NP-hard problem, is different in each case. Both training and testing NMSEs are recorded. A tradeoff between the recorded NMSEs is made before selecting a CP rank for a given model. This ensures that the model does not overfit the data while training. The CP rank is varied as a function of nuclear norm regularization λ_n, with each combination of R and λ_n simulated ten times.

Our first observation is that there is no particular trend when the CP ranks are increased for both models. A larger CP rank is supposed to have better NMSEs, but this is not the case. Discussing the trends for the model with $\rho = 0.6\%$ as given in Table (5-2), models having CP ranks $R = [36, 40, 50]$ are found to perform better than the others in terms of both training and testing data. $R = 33$ seems to be an outlier in the identification process. One can argue that $R = 50$ seems best suited for the model. Although the training NMSEs show improvement, the testing NMSEs remain comparable to $R = [36, 40]$. This is identified to be a case of overfitting. Thus, we chose $R \approx \mathbf{36}$, having made a tradeoff between the least training and testing NMSEs.

With $\rho = 2.5\%$, similar observations are made and recorded in Table (5-3). CP ranks $R = [7, 15, 25]$ are found to fit the model better while training. While testing the model, $R = [15, 25]$ performs better, with $R = 25$ being identified as another model overfitting case.

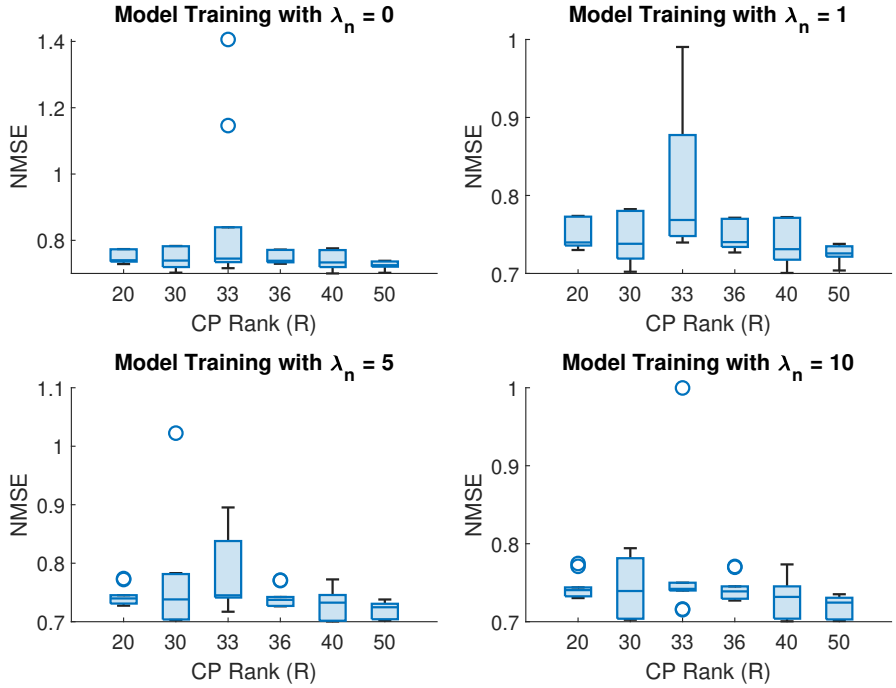

Figure 5-1: Influence of varying CP ranks on Training NMSE for $\rho = 0.6\%$

$R = 20$ is identified as an outlier in the identification process. Similar to $\rho = 0.6\%$, we chose $R \approx \mathbf{15}$. The chosen CP ranks are with the lowest NMSEs and least variance, as seen in Figures (C-1), (C-2), (C-3) and (C-4) in Appendix (C-1).

CP Rank	Training NMSE				Testing NMSE			
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
20	0.748	0.748	0.744	0.745	0.800	0.801	0.799	0.800
30	0.743	0.742	0.765	0.742	0.801	0.800	0.821	0.802
33	0.853	0.820	0.775	0.764	0.879	0.843	0.816	0.810
36	0.746	0.746	0.741	0.743	0.798	0.798	0.795	0.796
40	0.738	0.736	0.732	0.733	0.800	0.798	0.798	0.798
50	0.724	0.724	0.721	0.720	0.796	0.797	0.796	0.795

Table 5-2: Aggregate NMSE with varying CP Rank for $\rho = 0.6\%$

The influence of nuclear norm regularization λ_n on CP rank is investigated. Increasing λ_n for a given CP rank benefits model training by recording lesser variance in NMSEs as shown in Figures (5-1) and (5-2). However, its choice does not profoundly influence the testing NMSEs. Ensuring the model is not overfitting while making accurate predictions is fundamental. The choice of CP rank is the identified challenge and does not depend on the value of λ_n.

Varying ρ, lower aggregate NMSE values are recorded for higher ρ. The difference is the tensor dimensions adapted to vary the data sampling. The decomposition method is observed to benefit from dense tensors. One must note that the variance in the solutions obtained for $\rho = 2.5\%$ is more when compared to $\rho = 0.6\%$ for a given CP rank. Tables (C-1), (C-2), (C-3) and (C-4) in Appendix (C-1) highlight this.

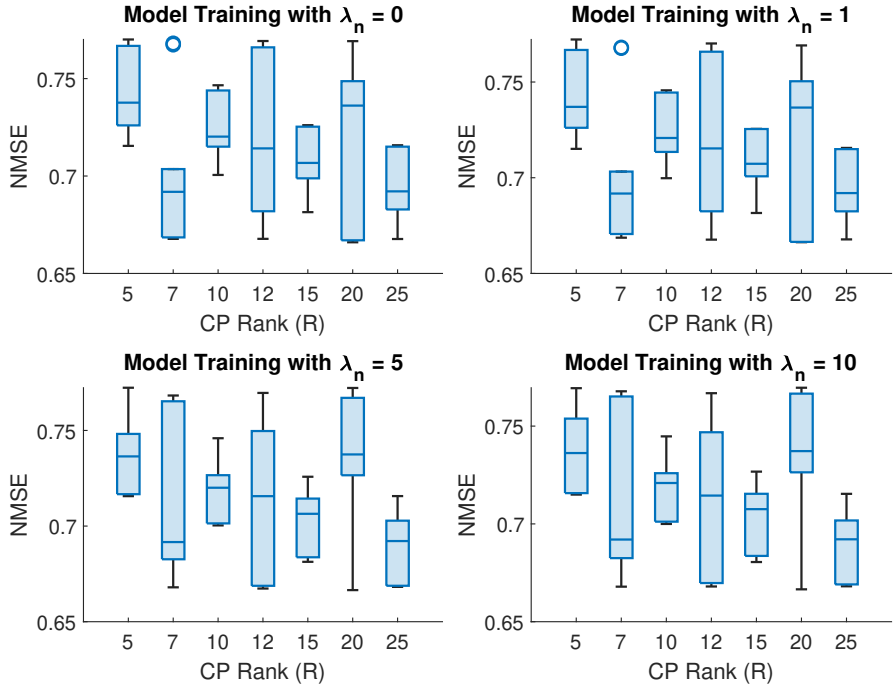

Figure 5-2: Influence of varying CP ranks on Training NMSE for $\rho = 2.5\%$

CP Rank	Training NMSE				Testing NMSE			
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
5	0.741	0.741	0.739	0.739	0.762	0.762	0.760	0.760
7	0.701	0.701	0.709	0.709	0.746	0.747	0.750	0.750
10	0.723	0.723	0.720	0.720	0.747	0.747	0.745	0.745
12	0.718	0.719	0.715	0.714	0.749	0.750	0.748	0.747
15	0.707	0.707	0.704	0.704	0.742	0.742	0.741	0.742
20	0.723	0.723	0.732	0.732	0.756	0.756	0.760	0.760
25	0.694	0.693	0.691	0.691	0.740	0.740	0.740	0.740

Table 5-3: Aggregate NMSE with varying CP Rank for $\rho = 2.5\%$

5-3-2 Influence of Regularization

The influence of regularization on model training and validation must be understood. Regularization is aimed at aiding the decomposition model and alleviating sparsity issues in RS. It helps in maintaining any structural information the data might contain. Fixing the CP rank for each model as identified in Section (5-3-1), the Laplacian regularization λ_l is varied as a function of nuclear norm regularization λ_n. Each combination of λ_l and λ_n is simulated ten times. Figures (5-3) and (5-4) depict how to graph Laplacian regularization aided with nuclear norm regularization helps in training an RS model.

Tables (5-4) and (5-5) record the variation of NMSEs with varying Laplacian regularization λ_l. Discussing the results obtained, regularization is seen to aid both $\rho = 0.6\%$ and $\rho = 2.5\%$. Values of $\lambda_l = [0.1, 0.01]$ benefit model training, showing reduced NMSEs compared to $\lambda_l =$

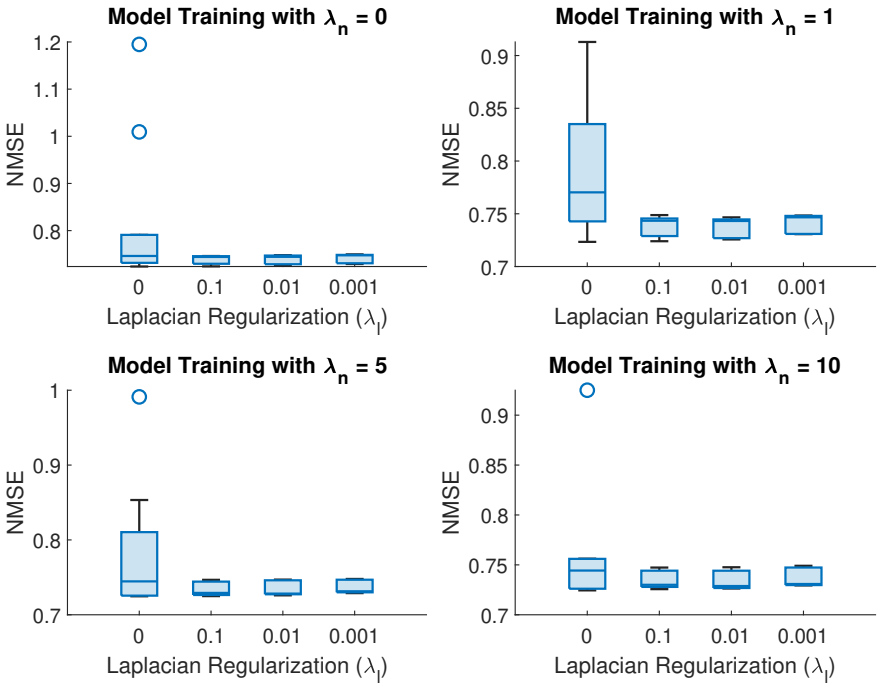

Figure 5-3: Influence of varying Laplacian Regularization λ_l on Training NMSE for $\rho = 0.6\%$

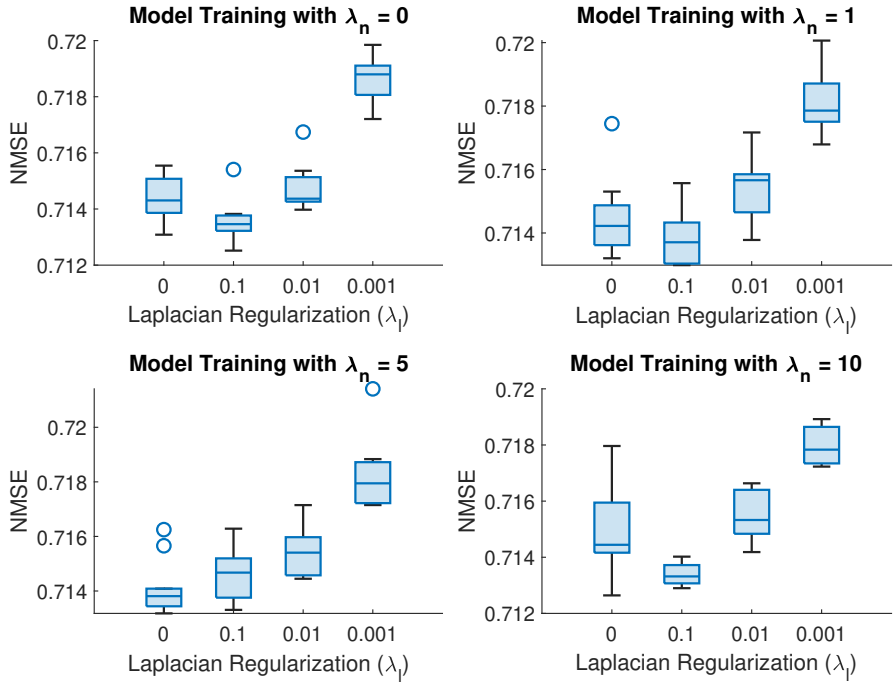

Figure 5-4: Influence of varying Laplacian Regularization λ_l on Training NMSE for $\rho = 2.5\%$

$[0, 0.001]$. However, it is to note that $\lambda_l = 0.001$ provides similar or in some cases, better testing NMSEs when compared to $\lambda_l = [0.1, 0.01]$.

The optimal regularization value is chosen to be $\lambda_l = \mathbf{0.01}$ for $\rho = 0.6\%$ and $\lambda_l = \mathbf{0.1}$ for $\rho = 2.5\%$. The need to promote sparseness in the solution explains the choice of a lower regularization value for $\rho = 0.6\%$. Laplacians constructed for $\rho = 2.5\%$ better capture the variation of ratings over the tensor, and the model benefits from higher regularization values.

Regularization λ_l	Training NMSE				Testing NMSE			
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
0	0.816	0.794	0.780	0.758	0.857	0.833	0.828	0.813
0.1	0.738	0.738	0.735	0.735	0.799	0.799	0.799	0.799
0.01	0.739	0.738	0.735	0.735	0.798	0.797	0.797	0.797
0.001	0.741	0.741	0.737	0.737	0.795	0.795	0.794	0.794

Table 5-4: Aggregate NMSE with varying Laplacian Regularization λ_l for $\rho = 0.6\%$

Introducing regularization λ_l provides better-trained models with reduced NMSEs than models without regularization. No regularization shows high NMSEs during training and testing for $\rho - 0.6\%$ compared to $\rho = 2.5\%$. Across tensor models, regularization aids reproducible results with lesser variance, as seen in Figures (C-5) and (C-6) of Appendix (C-2). This greatly influences the NMSEs when the tensors have smaller sampling rates.

Lower aggregate NMSE values are recorded for higher ρ. The role of regularization is better observed when the sampling rate is lower, i.e., $\rho = 0.6\%$. Obtaining solutions for higher ρ tensors does not need regularization but benefits if present. Tables (C-5), (C-6), (C-7) and (C-8) in Appendix (C-2) highlight the variation of regularization for varying ρ.

Regularization λ_l	Training NMSE				Testing NMSE			
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
0	0.714	0.714	0.714	0.715	0.749	0.749	0.749	0.750
0.1	0.714	0.714	0.715	0.713	0.749	0.749	0.750	0.748
0.01	0.715	0.715	0.715	0.715	0.748	0.749	0.749	0.749
0.001	0.719	0.718	0.718	0.718	0.750	0.749	0.749	0.749

Table 5-5: Aggregate NMSE with varying Laplacian Regularization λ_l for $\rho = 2.5\%$

5-3-3 Performance Evaluation Metrics

Percentage of Fit (POF)

Relative Reconstruction Error (RRE) or Percentage of Fit (POF) is the data accurately recovered by the model. The higher the POF, the better the model fits the training data. The effect of varying sampling and nuclear norm regularization λ_n on POF is analyzed.

Starting with the chosen CP ranks, we find that the POF improves with an increase in λ_n. Intended to promote sparse solutions, the nuclear norm regularization ensures a low-rank representation of the models. This is beneficial while training sparse data sets for a given CP rank. POF increases with CP rank, as seen in Tables (5-6) and (5-7), but could lead to model overfitting while training.

CP Rank				
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
20	0.157	0.157	0.160	0.159
30	0.158	0.159	0.022	0.091
33	-0.338	-0.397	-0.168	-0.064
36	0.156	0.156	0.159	0.159
40	0.164	0.165	0.169	0.169
50	0.177	0.176	0.179	0.179

Table 5-6: Aggregate POF with varying CP Rank for $\rho = 0.6\%$

The POFs associated with λ_l in Table (5-8) highlight the influence of ρ and λ_l for building RS using the GRCP framework. Given $\rho = 0.6\%$, the POF obtained is poor with $\lambda_l = 0$. Introducing λ_l improves the POF. Contrarily, with $\rho = 2.5\%$, the POFs obtained without and with λ_l are consistent with a drop at $\lambda_l = 0.001$. Sparse tensors with low ρ require λ_l to ensure a good model POF which is not the case for dense tensors with higher ρ. The POF is another indicator that highlights the benefits of Laplacian regularization in the GRCP model.

CP Rank				
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
5	0.154	0.154	0.156	0.156
7	0.185	0.185	0.179	0.179
10	0.162	0.162	0.164	0.164
12	0.169	0.168	0.171	0.172
15	0.179	0.179	0.181	0.181
20	0.168	0.168	0.161	0.161
25	0.191	0.191	0.193	0.194

Table 5-7: Aggregate POF with varying CP Rank for $\rho = 2.5\%$

Regularization λ_l	$\rho = 0.6\%$				$\rho = 2.5\%$			
	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$	$\lambda_n = 0$	$\lambda_n = 1$	$\lambda_n = 5$	$\lambda_n = 10$
0	-0.168	-0.207	-0.146	-0.030	0.174	0.174	0.174	0.173
0.1	0.162	0.162	0.165	0.165	0.174	0.174	0.174	0.174
0.01	0.162	0.163	0.165	0.165	0.173	0.172	0.172	0.172
0.001	0.160	0.160	0.163	0.163	0.170	0.170	0.170	0.170

Table 5-8: Aggregate POF with varying λ_l for $\rho = 0.6\%$ and $\rho = 2.5\%$

Increasing data sampling results in better POF of tensor models. However, the evaluation of POF may be debatable if a tensor is sparse in data. Given that no constraints are imposed on the factor matrices during the optimization, POF might be misleading. Negative POF means most of the entries in the obtained model are negative. Increasing λ_n is needed here to promote sparsity in solutions, or imposing non-negative constraints on the factor matrices must be carried out.

Normalized Mean Square Error (NMSE)

Running the algorithm multiple times and computing the average NMSE is representative of the model's accuracy. We plot the average NMSE over ten iterations with varying CP rank and regularization λ_l.

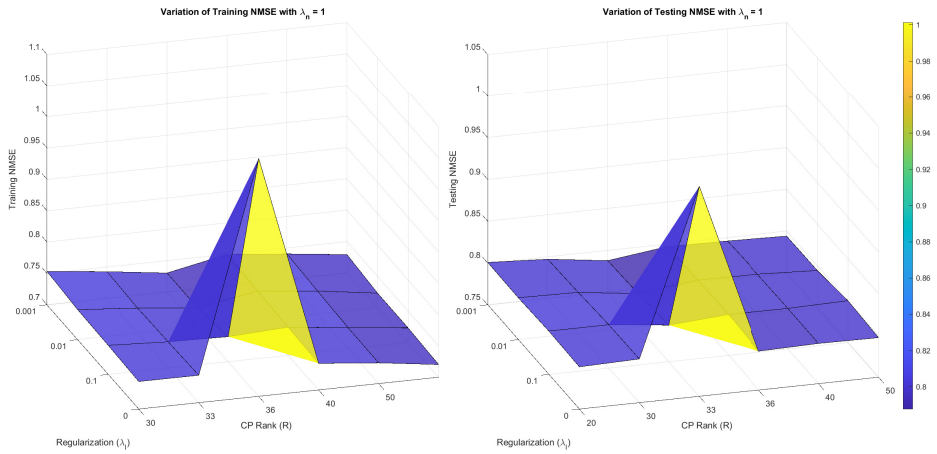

Figure 5-5: Influence of varying hyperparameters on Training NMSE for $\rho = 0.6\%$

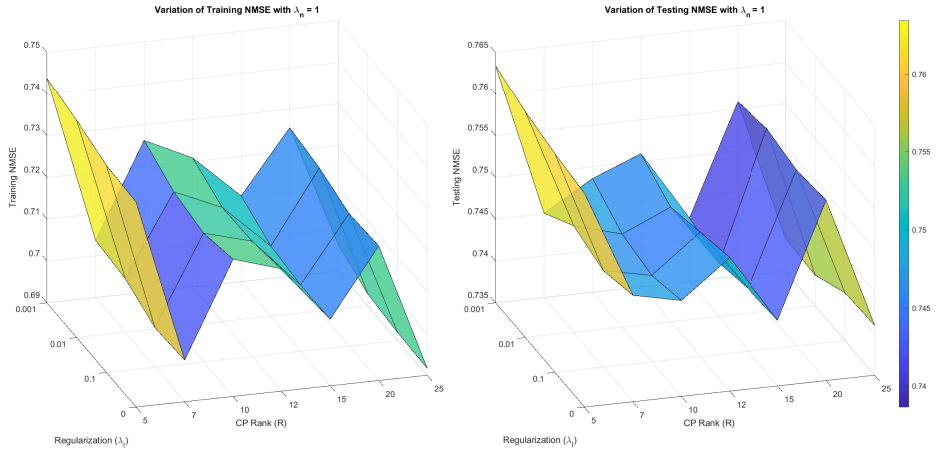

Figure 5-6: Influence of varying hyperparameters on Training NMSE for $\rho = 2.5\%$

One can see that there are distinct values of CP rank and regularization λ_l for which the models are well trained. The tensor models being solved are observed to have similar NMSEs within a neighborhood of varied hyperparameters. Identifying the right CP rank and regularization becomes difficult when the optimization problem has saddle points. This becomes the case when handling tensors having low sampling, as seen in Figure (5-5). Evaluation of the model on testing data becomes more vital in deciding the values of the hyperparameters.

k-Cross Validation and Benchmarking

The proposed model has been evaluated on various performance metrics. Reproducibility of results when the data within the tensor is shuffled or permuted must be ensured. k-Cross validation is performed on the given data set, taking `k = 5` equally shuffled training and testing data. The training and validation data sets maintain a ratio of 80-20 evaluation points.

k-Cross Validation	POF	Training NMSE	Testing NMSE
$\rho = 0.6\%$	0.171	0.724	0.781
$\rho = 2.5\%$	0.18	0.698	0.724

Table 5-9: k-Cross Validation of the chosen models

The model parameters CP rank R= 36, $\lambda_l = 0.01$ and $\lambda_n = 1$ is set for $\rho = 0.6\%$ and CP rank R= 15, $\lambda_l = 0.1$ and $\lambda_n = 1$ is set for $\rho = 2.5\%$. The models are then evaluated on shuffled data, and their performance metrics are recorded in Table (5-9). The repeatability and reproducibility of results are observed well with similar aggregated POF and NMSE values as obtained during model evaluation.

In addition to validating the chosen models, benchmarking the GRCP model against standard state-of-the-art methods is important. Table (5-10) gives us the benchmarking results performed on four models: NoReg refers to the standard CP decomposition with $\lambda_n, \lambda_l = 0$, NReg refers to a nuclear-regularized CP decomposition with $\lambda_n = 1, \lambda_l = 0$, LReg refers to a graph Laplacian regularized CP decomposition with $\lambda_n = 0, \lambda_l = [0.01, 0.001]$ depending on ρ, and (N+L)Reg is the GRCP model proposed in this thesis.

Benchmarking	$\rho = 0.6\%$				$\rho = 2.5\%$			
	NoReg	NReg	LReg	(N+L)Reg	NoReg	NReg	LReg	(N+L)Reg
Training NMSE	0.725	0.725	**0.723**	0.724	0.7	0.701	0.704	**0.698**
Testing NMSE	0.786	0.785	**0.781**	**0.781**	0.732	0.731	0.726	**0.724**
POF	**0.173**	**0.173**	0.171	0.171	**0.185**	0.184	0.177	0.18

Table 5-10: Benchmarking with other CP tensor decomposition methods

The NMSEs of the GRCP model for training and testing are observed to be better than other benchmark models. The LReg models show similar NMSE values compared to the GRCP model for lower values of ρ but requires nuclear norm regularization to promote sparse solutions as ρ is increased. Evaluating the POF, models with no regularization performs better compared to other models.

5-4 Scalability

The scalability of the GRCP framework is important in building RS models. Expected to scale in linear time, we shall evaluate the proposed framework with larger datasets. The MovieLens 1M data set containing 1,000,209 ratings from 6040 users for 3952 movies is used. The ratings are similar to the ML100K dataset, which dates from April 2000 to February 2003.

5-4-1 Experiment Setup

The ML1M dataset is loaded onto MATLAB as a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3} \in \mathbb{R}^{6040 \times 3952 \times T_m}$ of `class(T) = 'double'`. Similar to the ML100K dataset, the tensor's third mode $\mathbf{T}_{(3)} \in \mathbb{R}^{T_m}$, represented as time, will be the year or month of rating. When represented with the rating year, the tensor dimensions are $\mathbb{R}^{6040 \times 3952 \times 4}$ with the time dimensions representing the years 2000 to 2004. Similarly, when represented with the month of rating, its dimensions are $\mathbb{R}^{6040 \times 3952 \times 12}$ with the time dimensions representing months from April through March. The Laplacians are obtained from a kNN graph built on each tensor mode $\mathcal{T}_{(n)}$ with the same edge weight kernel as defined in (5-2).

The data containing 1,000,209 ratings are split with an 80-20 ratio for training and validating the model. With tensor dimensions $\mathcal{T} \in \mathbb{R}^{943 \times 1682 \times 4}$, the training sampling rate is $\rho = 0.8\%$ while with tensor dimensions $\mathcal{T} \in \mathbb{R}^{943 \times 1682 \times 12}$, the training sampling rate is $\rho = 0.3\%$. The GRCP model framework is trained on the training dataset for given tensor dimensions.

The computations on the ML1M dataset will be carried out on the Delft Blue Supercomputer using Intel XEON E5-6248R 24C 64-bit 24 Core Processor running at 3.0GHz with 64GB DDR4 RAM [72]. The software used is MATLAB R2021b, and the GSP Toolbox [67] for creating the graph Laplacians.

5-4-2 Results and Discussions

The two important aspects determining a model's scalability are the accuracy of its predictions and the time taken to train a model. Having identified the best hyperparameters for the models similar to that carried out in the ML100K dataset, we compare the NMSEs and algorithm run times of the two datasets.

Given as a function of sampling ρ, the aggregate NMSEs and POFs of the models are presented in Table (5-11). Smaller tensor dimensions benefit from lower training NMSE and a higher POF. The training times are also less, given that the time taken to train directly depends on the tensor dimensions. Noticeably, the testing NMSE reduces with increasing ρ.

Scalability	$\rho = 0.3\%$	$\rho = 0.6\%$	$\rho = 0.8\%$	$\rho = 2.5\%$
Training NMSE	0.808	0.724	0.752	0.698
Testing NMSE	0.811	0.781	0.756	0.724
POF	0.113	0.171	0.149	0.18
Run Time (s)	389.12	121.18	347.76	156.18

Table 5-11: Aggregate NMSE and Algorithm Run Times for varying ρ

The algorithm is timed on MATLAB using the `tic` and `toc` commands, and the aggregate run times are recorded. Figures (C-7), (C-8), (C-9) and (C-10) in Appendix (C-4) show the average runtimes of the algorithms. Written in terms of the big $\mathcal{O}$ notation, the number of floating point operations is calculated as follows,

$$\mathcal{O}(g) = \texttt{maxALSiter}\left[\mathcal{O}(|\Omega|R) + \mathcal{O}(R^3) + \texttt{maxCGiter}\sum_{n=1}^{d}\mathcal{O}(I_n R^2 + \text{nnz}(\mathbf{L}^{(n)})R)\right]$$

Table (5-12) gives an approximate Floating Point Operations per Second (FLOPS) that occur with the given hardware configurations. Increasing the sampling ρ benefits from fewer FLOPS required. Notably, the required FLOPS scales linearly with ρ, decided by the tensor dimensions and CP rank.

big $\mathcal{O}$	$\rho = 0.3\%$	$\rho = 0.6\%$	$\rho = 0.8\%$	$\rho = 2.5\%$
Time Taken (s)	389.12	121.18	347.76	156.18
No. Floating Point Operations	$\mathcal{O}(8.012e^{10})$	$\mathcal{O}(2.107e^{10})$	$\mathcal{O}(1.456e^{10})$	$\mathcal{O}(3.799e^{9}))$
FLOPS	$2.06e^{8}$	$1.74e^{8}$	$4.18e^{7}$	$2.43e^{7}$

Table 5-12: Computational scalability for varying ρ

5-5 Conclusions

This chapter uses the proposed GRCP tensor decomposition model framework to build an RS model. The variation of CP rank and graph Laplacian regularization λ_l are studied extensively as a function of sampling ρ and nuclear norm regularization λ_n. Structuring data in the tensor dictating data sampling (or) sparsity significantly influences NMSE and POF values. The Laplacians are constructed, and the regularization value is tuned accordingly. Nuclear norm regularization benefits the tensor decomposition process if not detrimental. One can conclude that regularization has a vital role in data recovery from sparse and dense tensors.

The chosen RS model using the ML100K dataset for $\rho = 0.6\%$ shows an average training NMSE of **0.724** and testing NMSE of **0.781**. Having $\rho = 2.5\%$ shows an average training NMSE of **0.698** and testing NMSE of **0.724**. NMSE measured in absolute terms of prediction accuracy shows that the model predicts better than traditional CF methods highlighted in the literature. Furthermore, the computational burden increases linearly with an increase in CP rank, offering promising scalability.

Chapter 6

Conclusions and Future Scope of Work

This Master thesis proposes a novel Graph Regularized CP (GRCP) tensor decomposition framework for building RS models. In our concluding remarks, we revisit the research questions and highlight the work done in this thesis for each question.

(RQ.1) *How does the GRCP model capture and utilize available data?*

Data represented as tensors is becoming common. Incorporating multi-contextual information plays a significant role in building accurate and robust data-based models. Data density can be controlled by the dimensions chosen for the tensor decomposition method. Smaller tensor dimensions increase the data density while promising improved prediction metrics and computational run times. Graphs built on available data have found their use as filters useful for regularization. The Laplacians built on each tensor mode capture the available interactions between the modeled entities.

Tensor decomposition methods have become efficient with improved computational resources. Low-rank representations of data are effective in building prediction models. With increased data sparsity, identifying a low-rank representation becomes crucial. This thesis discusses a GRCP-ALS tensor decomposition method to obtain the low-rank representations as factor matrices for a given CP rank.

Graph signal processing is observed to extract underlying data from graph structures effectively. Defining the k-nearest neighbors for a kNN graph is found to be of significant influence for dense data. The sparser the data becomes, the choice of kNN members is observed not to influence the prediction accuracy of the built model. Weighting kernels used to build the graphs is another essential factor. This thesis shows two types of weighting kernels based on the application and nature of the data present. The choice of the kernel aids in the recovery of data that is corrupted with noise. As a result of the built graphs, utilizing graph Laplacians to impose constraints on how the data should vary aids in building low-rank models.

(RQ.2) *How is the data combined in GRCP to aid recommendations?*

The addition of regularizers is relatively simple in the explanatory CP decomposition method. Low-rank solutions are ensured by imposing a nuclear norm in the objective, while the Laplacians influence the latent interactions of data between tensor modes. In conjunction with

tensor decomposition methods, this thesis uses graph Laplacians to regularize each tensor mode while solving the optimization problem. This dictates the variability that the data can contain. Furthermore, it aids tensor decomposition methods by providing accurate factor matrices for rebuilding a given model.

Nuclear norm regularization has not affected the model's CP rank. It promotes sparser and unique solutions to the problem. The choice of Laplacian matrices is also motivated for convergence analysis of the ALS algorithm. The positive definiteness of the Laplacians makes it favorable for solving convex optimization problems. The proposed GRCP model is studied for convergence and computational efficiency. Achieving global convergence, the combination of information is theoretically seen not to increase the computational burden and depends on the tensor dimensions and chosen CP rank.

(RQ.3) *How does the GRCP address the drawbacks that current RS are susceptible to?*

Applying the GRCP framework to build better RS models is the main research question of the thesis. Issues such as sparsity and robustness in prediction accuracy are studied with the given framework. The effect of nuclear and Laplacian norm regularization is recorded and discussed in detail.

Evaluating the model on performance metrics such as NMSE, POF, and computational run times, variation in data density is important for better performance with regularization. Nuclear norm regularization ensures reduced variance in NMSEs and robustness to outliers during model training. Laplacian regularization effectively produces lower NMSEs while testing a given model. Overall, regularization aids in the reconstruction of entries that are not observable, thus solving the issue of sparsity observed in current RS models. Benchmarking with other tensor decomposition models is carried out, with the GRCP performing better than other standard benchmarks.

The modified CP-ALS methodology using a CG solver benefits the optimization of the factor matrices. An increase in CP rank affects computational time linearly and, thus, is scalable for testing on larger datasets. In addition to building RS models, the thesis also discusses an extended application for image representation and presents use cases for image denoising using graph Laplacian regularization.

Future Scope of Work

This thesis limits and tests the GRCP framework with matrix dimensions computable by the given machines. Experimenting on larger datasets must be carried out to verify the algorithm's scalability in terms of prediction accuracy. Datasets such as MovieLens ML10M and ML20M are suggested to carry out this analysis.

GPUs are explicitly intended to perform matrix calculations. The thesis evaluates the GRCP framework using a CPU. CPUs process data serially, while GPUs have multi-threading and parallel processing capabilities. Given that the identified bottlenecks of the current code are matrix multiplications running the proposed framework on GPUs would improve computational efficiency multifold. This would reduce memory throughput times with fast and accurate floating point arithmetic operations being performed. The code provided in the repositories is also compatible with GPU computations.

Another promising data storage format is Tensor Trains. Using Singular Value Decomposition (SVD) methods, Tensor Train is widely used for efficient and accurate decomposition methods. Operations can be performed on Tensor Trains represented as vectors or matrices. It uses SVD computations which are computationally intensive and cannot be parallelized. However, the Tensor Train representation allows relatively smaller dimensions of the matrices being evaluated. This helps the use case for building models that current machines can handle computationally. Requirements for memory storage are also less with the use of Tensor Trains.

The variability in the data can be well defined using alternates of graph Laplacians. Using a local weighting scheme to impose smoothness measures has proven effective compared to a global weighting coefficient. This weighting matrix could also be introduced into the optimization problem to build accurate models. Laplacians of directed or bipartite graphs can be evaluated and used to define a similar optimization objective function. Variability in the data can be captured using different smoothness measures such as Sobolev Smoothness or other kernel-based methods.

Appendix A

Graph Regularized (GRCP) Tensor Decomposition

A-1 Graph Regularized CP Tensor Decomposition Model

The proposed objective function for minimization is given by,

$$f(\mathbf{X^{(n)}}) = \min_{\mathbf{X}^{(n)}} \frac{1}{2} \left\| \mathcal{W}_{\Omega^{(n)}} \left(\mathcal{T}_{(n)} - \mathbf{X}^{(n)} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \right) \right\|_F^2 + \frac{1}{2} ||\mathbf{X}^{(n)^{\mathrm{T}}} \mathbf{L}^{(n)} \mathbf{X}^{(n)}||_F^2$$

Expanding and neglecting the terms that are independent of the minimizing variable,

$$f(\mathbf{X^{(n)}}) = \min_{\mathbf{X}^{(n)}} \frac{1}{2} \operatorname{Tr} \left\{ \mathbf{X}^{(n)^{\mathrm{T}}} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right] \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \mathbf{X}^{(n)} - 2\mathbf{X}^{(n)^{\mathrm{T}}} \left(\mathcal{W}_{\Omega^{(n)}} \mathcal{T}_{(n)} \right) \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right] \right\}$$
$$+ \frac{1}{2} \operatorname{Tr} \left\{ \mathbf{X}^{(n)^{\mathrm{T}}} \mathbf{L}^{(n)} \mathbf{X}^{(n)} \right\}$$

We combine similar terms and rewrite the above objective function as a quadratic subproblem for each ALS iteration.

$$f(\mathbf{X^{(n)}}) = \underset{\mathbf{X^{(n)}} \in \mathbb{R}^{I_n \times R}}{\text{minimize}} \frac{1}{2} \operatorname{Tr} \left\{ \mathbf{X^{(n)}}^{\mathrm{T}} \left[\mathbf{C} + \mathbf{L}^{(n)} \right] \mathbf{X^{(n)}} - 2\mathbf{X^{(n)}}^{\mathrm{T}} \mathbf{Y}^{(n)} \right\}$$

with $\mathbf{Y}^{(n)} = \left(\mathcal{W}_{\Omega^{(n)}} \mathcal{T}_{(n)} \right) \left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \in \mathbb{R}^{I_n \times R}$, $\mathbf{C} = \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right] \in \mathbb{R}^{R \times R}$.

Thus, the quadratic minimization objective function is given by the form,

$$g(\mathbf{x}^{(n)}) = \frac{1}{2} \mathbf{x^{(n)}}^{\mathrm{T}} \mathbf{H}^{(n)} \mathbf{x}^{(n)} - \mathbf{x^{(n)}}^{\mathrm{T}} (\mathbf{Y}^{(n)})$$

with $\mathbf{x}^{(n)} = \operatorname{vec}(\mathbf{X}^{(n)})$ and $\mathbf{H^{(n)}} = \left[\mathbf{C} \otimes \mathbf{I}_{I_n} + \mathbf{I}_R \otimes \mathbf{L}^{(n)} \right] \in \mathbb{R}^{RI_n \times RI_n}$.

A-2 Alternating Least Squares (ALS) algorithm

Presented here is an adaptation of the ALS algorithm as given in Algorithm (1) to the objective function mentioned in Equation (3-5).

Algorithm 4 CP-ALSLap Algorithm for order d Tensor

function CP-ALSLAP($\mathcal{T}, \mathcal{W}_\Omega, \lambda_n, \lambda_l, \mathbf{L}_{(n)}$,R)
  initialize $\mathcal{X} = \langle \mathcal{W}_\Omega, \mathcal{T} \rangle$ ▷ Tensor Inner Product
  initialize $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n \in [2, d]$
  initialize $E_n = 0$
  repeat
    $E_o = E_n$
    for n = 1 ,. . . , d **do**
      $\mathbf{C} = \mathbf{X}^{(1)\mathrm{T}}\mathbf{X}^{(1)} \dots \mathbf{X}^{(n-1)\mathrm{T}}\mathbf{X}^{(n-1)}\mathbf{X}^{(n+1)\mathrm{T}}\mathbf{X}^{(n+1)} \dots \mathbf{X}^{(d)\mathrm{T}}\mathbf{X}^{(d)}$
      $\mathbf{X}_0^{(n)} = \mathcal{X}_{\times n}\left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]$
      $\mathbf{X}^{(n)} = \text{GRCP-ConjGrad}(\lambda_n, \lambda_l, \mathbf{L}_{(n)}, \mathbf{C}, \mathbf{X}_0^{(n)})$
      normalize columns of $\mathbf{X}^{(n)}$ and store norms as λ
    end for
    $E_n = 1 - \frac{||\mathcal{X} - \hat{\mathcal{X}}||_F}{||\mathcal{X}||_F}$
  until max. iterations reached or $|E_n - E_o| <$ tol
  return $\lambda, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(d)}$
end function

A-3 Convergence Analysis

A-3-1 Lipschitz Boundedness

The quadratic subproblems to be solved using the CG solver is given as follows,

$$g(\mathbf{x}^{(n)}) = \frac{1}{2}\mathbf{x}^{(n)\mathrm{T}}\mathbf{H}^{(n)}\mathbf{x}^{(n)} - \mathbf{x}^{(n)\mathrm{T}}\mathbf{y}^{(n)}$$

Let $\mathbf{x}^{(n)}$ be the initial point of the algorithm, $\mathbf{x}_*^{(n)}$ be the solution of the subproblem. The error between the optimal and initial point is given by $\mathbf{e} = \mathbf{x}^{(n)} - \mathbf{x}_*^{(n)}$. After every iteration, the initial point is updated such that $\mathbf{x}_{k+1}^{(n)} = \mathbf{x}_k^{(n)} + \mathbf{e}$. This ensures $\mathbf{x}^{(n)} \to \mathbf{x}_*^{(n)}$, $\left\|\mathbf{x}^{(n)} - \mathbf{x}_*^{(n)}\right\| < \epsilon$ for $\lim \mathbf{e} \to 0$ and $\epsilon > 0$.

$$g(\mathbf{x}_k^{(n)} + \mathbf{e}) = \frac{1}{2}(\mathbf{x}_k^{(n)} + \mathbf{e})^{\mathrm{T}}\mathbf{H}^{(n)}(\mathbf{x}_k^{(n)} + \mathbf{e}) - (\mathbf{x}_k^{(n)} + \mathbf{e})^{\mathrm{T}}\mathbf{y}^{(n)}$$

$$= g(\mathbf{x}_k^{(n)}) + \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{H}^{(n)}\mathbf{e}$$

The Hessian $\mathbf{H}^{(n)}$ being positive-definite and with $\lim \mathbf{e} \to 0$, we can conclude that $\mathbf{x}_k^{(n)}$ minimizes the objective function. The optimal solution of the subproblem is $\mathbf{x}_*^{(n)} = \mathbf{H}^{(n)^{-1}}\mathbf{y}^{(n)}$.

The method of CG finds the solution of the quadratic subproblem by performing a line search orthogonal to the gradient and equating it to zeros. Let the error $\mathbf{e} = \alpha\mathbf{p}$. Using a Netwton-Raphson method to evaluate the search direction, the Taylor series approximation of the function is given by,

$$g(\mathbf{x}^{(n)} + \alpha\mathbf{p}) \approx g(\mathbf{x}^{(n)}) + \alpha\left[\nabla g(\mathbf{x}^{(n)})\right]^{\mathrm{T}}\mathbf{p} + \frac{\alpha^2}{2}\mathbf{p}^{\mathrm{T}}\nabla^2 g(\mathbf{x}^{(n)})\mathbf{p}$$
$$\frac{d}{d\alpha}g(\mathbf{x}^{(n)} + \alpha\mathbf{p}) \approx \left[\nabla g(\mathbf{x}^{(n)})\right]^{\mathrm{T}}\mathbf{p} + \alpha\mathbf{p}^{\mathrm{T}}\nabla^2 g(\mathbf{x}^{(n)})\mathbf{p}$$

The residual $\mathbf{p}$ is initially set to the negation of the gradient function, i.e., $\mathbf{p} = -\nabla g(\mathbf{x}^{(n)}) = \mathbf{y}^{(n)} - \mathbf{H}^{(n)}\mathbf{x}^{(n)}$, $\nabla^2 g(\mathbf{x}^{(n)})$ is the Hessian $\mathbf{H}^{(n)}$. Setting the gradient obtained from the Taylor series approximation to zero, the orthogonal line search direction $\alpha = \frac{\mathbf{p}^{\mathrm{T}}\mathbf{p}}{\mathbf{p}^{\mathrm{T}}\mathbf{H}^{(n)}\mathbf{p}}$. Substituting the obtained values in the gradient obtained from the Taylor series approximation, we obtain the Lipschitz bounds for the objective function as follows,

$$\left\|\left[\frac{d}{d\alpha}g(\mathbf{x}^{(n)} + \alpha\mathbf{p})\right]\right\| \approx 2\left\|\mathbf{p}\right\|^2$$

The bounds obtained is the Lipschitz bound of the gradient function as $\mathbf{e} = \alpha\mathbf{p} \to 0$ with each CG iteration. With no improvement in the objective function, the CG solver is reset after max iterations.

A-3-2 KL Inequality for Real Analytic Functions

$$\min_{\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R}} \frac{1}{2}\left\|\mathcal{W}_{\Omega^{(n)}}\left(\mathcal{T}_{(n)} - \mathbf{X}^{(n)}\left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}}\right)\right\|_F^2 + \frac{1}{2}||\mathbf{X}^{(n)^{\mathrm{T}}}\mathbf{L}^{(n)}\mathbf{X}^{(n)}||_F^2$$

where $\mathbf{L}^{(n)} = \lambda_n\mathbf{I} + \lambda_l\mathbf{L}_{(n)}$. The objective function given above is can be written as a sum of two functions.

$$g(\mathbf{X}^{(\mathbf{n})}) = g_e(\mathbf{X}^{(\mathbf{n})}) + g_r(\mathbf{X}^{(\mathbf{n})})$$

where $g_e(\mathbf{X}^{(\mathbf{n})})$ is the error minimization term and $g_r(\mathbf{X}^{(\mathbf{n})})$ is the regularizer term. Dropping the $\mathcal{W}_{\Omega^{(n)}}$ term and writing out $g_e(\mathbf{X}^{(\mathbf{n})})$,

$$g_e(\mathbf{X}^{(\mathbf{n})}) = \frac{1}{2}\left\|\mathcal{T}_{(n)} - \mathbf{X}^{(n)}\left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}}\right\|_F^2 = \frac{\left\|\mathbf{X}^{(n)} - \hat{\mathbf{X}}^{(\mathbf{n})}\right\|_F^2}{2\left\|\left[\left(\mathbf{X}^{(i)}\right)^{\odot i \neq n}\right]^{\mathrm{T}}\right\|_F^2}$$

The above can be written as a power series that is represented as,

$$g_e(\mathbf{X}^{(\mathbf{n})}) = \sum_{j=0}^{\infty} a_j(\mathbf{X}^{(n)} - \alpha)^j$$

with $a_j = \frac{1}{2\left\| \left[\left(\mathbf{X}^{(i)} \right)^{\odot i \neq n} \right]^{\mathrm{T}} \right\|_F^2}$ and $\alpha = \hat{\mathbf{X}}^{(\mathbf{n})}$. Similarly, the regularizer term $g_r(\mathbf{X}^{(\mathbf{n})})$ can be written out as,

$$g_r(\mathbf{X}^{(\mathbf{n})}) = \frac{1}{2}||{\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{L}^{(n)} \mathbf{X}^{(n)}||_F^2 = \frac{\left\| {\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{X}^{(n)} \right\|_F^2}{2||\mathbf{L}^{(n)}||_F^2}$$

The power series when applied to $g_r(\mathbf{X}^{(n)})$ becomes $g_r(\mathbf{X}^{(n)}) = \sum_{j=0}^{\infty} b_j ({\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{X}^{(n)})^j$ with $b_j = \frac{1}{2||\mathbf{L}^{(n)}||_F^2}$ and $\alpha = 0$. The objective function can, thus, be written as a power series of the form as given in Equation (3-8) with,

$$g(\mathbf{X}^{(\mathbf{n})}) = \sum_{j=0}^{\infty} a_j (\mathbf{X}^{(n)} - \alpha)^j + \sum_{j=0}^{\infty} b_j ({\mathbf{X}^{(n)}}^{\mathrm{T}} \mathbf{X}^{(n)})^j$$

The series converges as $j \to \infty$ on some interval of positive radius centered at the combination of the two functions. This concludes the proof that the formulated objective function is a sum of real analytic functions and, thus, satisfies the KL inequality.

Appendix B

Model Validation Using Synthetic Dataset

B-1 Performance Evaluation Metrics

B-1-1 Algorithm Run Times

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**
Min Run Time (s)	2.93	1.93	17.85	8.26
Max Run Time (s)	256.07	124.95	205.50	206.29
Average Run Time (s)	66.81	63.72	67.05	64.34

Table B-1: Aggregate runtimes for varying regularization λ_l

SNR (dB)	**0**	**0.1**	**0.5**	**1**	**4**	**7**
Min Run Time (s)	27.21	8.26	1.93	20.49	19.42	2.93
Max Run Time (s)	134.18	106.95	256.07	117.55	90.69	112.42
Average Run Time (s)	63.46	60.65	71.43	70.59	60.16	61.95

Table B-2: Aggregate runtimes for varying SNR (dB)

Appendix C

Model Validation Using MovieLens Dataset

C-1 Influence of CP Rank

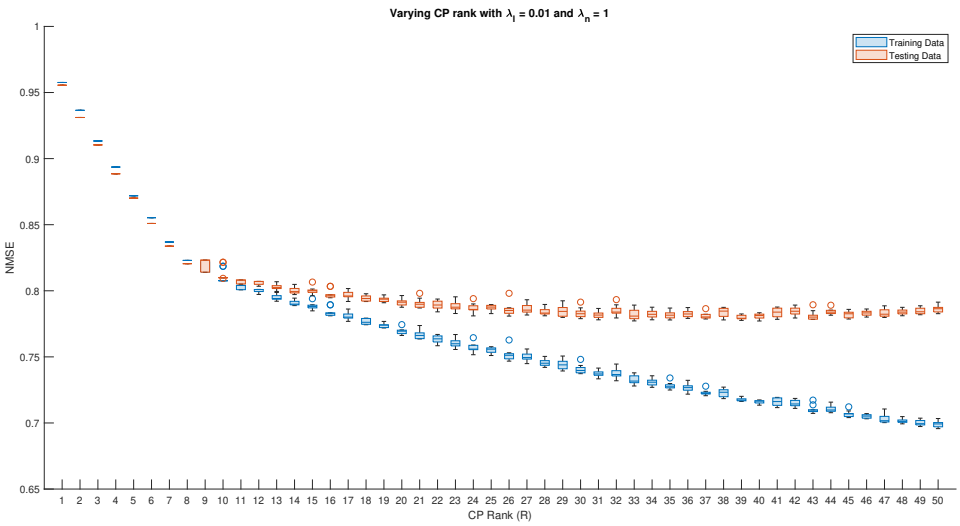

Figure C-1: Influence of varying CP ranks for $\rho = 0.6\%$

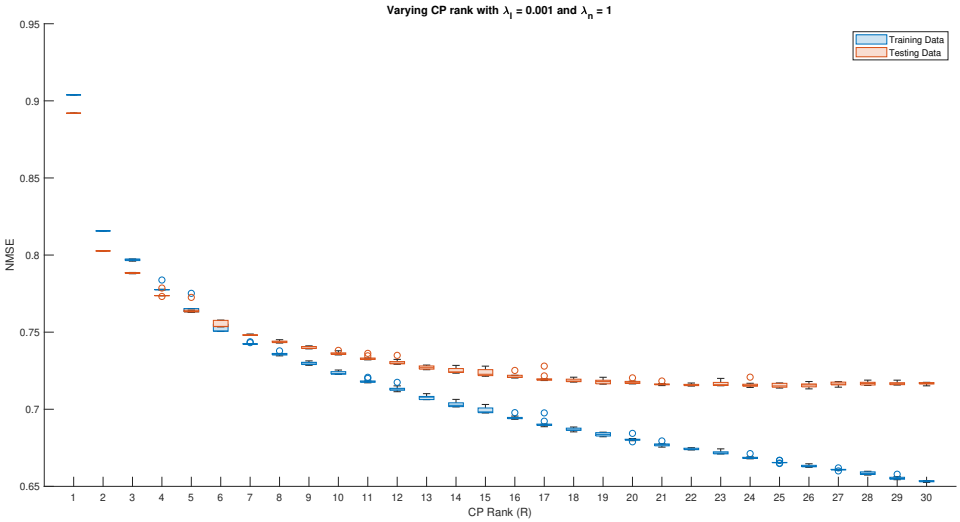

Figure C-2: Influence of varying CP ranks for $\rho = 2.5\%$

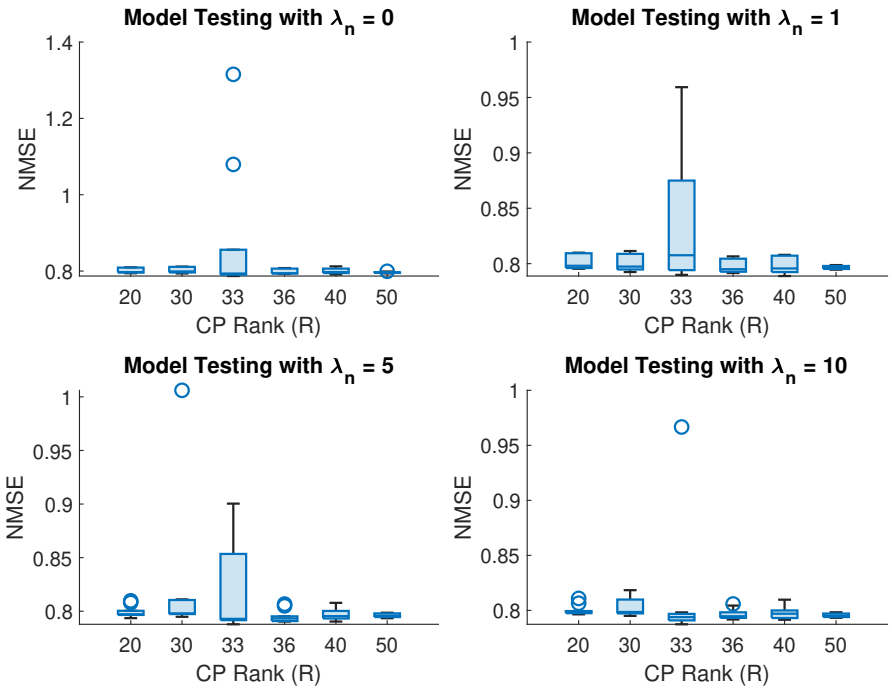

Figure C-3: Influence of varying CP ranks on Testing NMSE for $\rho = 0.6\%$

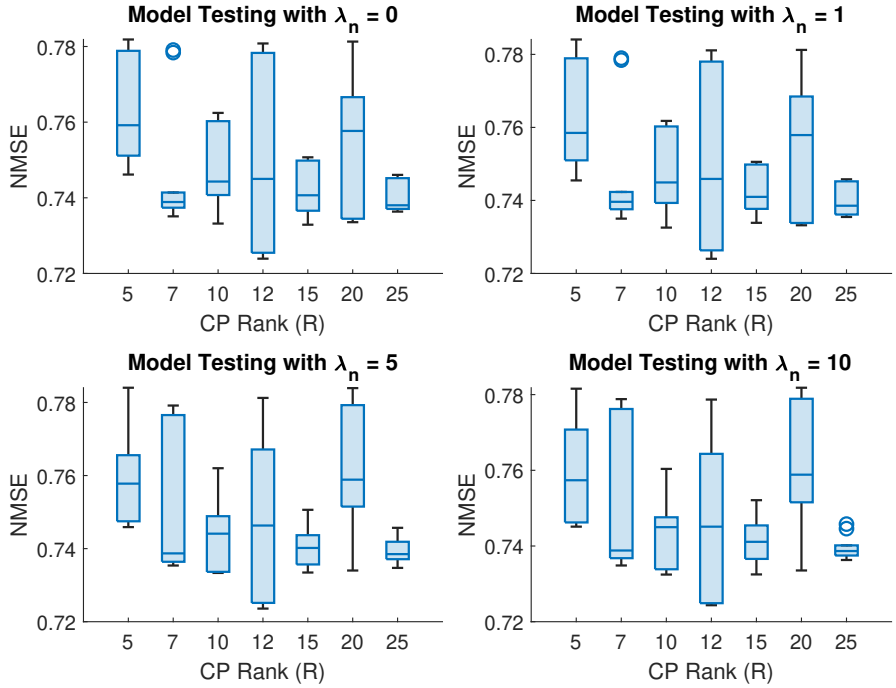

Figure C-4: Influence of varying CP ranks on Testing NMSE for $\rho = 2.5\%$

CP Rank	20	30	33	36	40	50	20	30	33	36	40	50
	$\lambda_n = 0$						$\lambda_n = 1$					
Min NMSE	0.744	0.742	0.730	0.743	0.736	0.720	0.744	0.740	0.730	0.743	0.733	0.720
Max NMSE	0.752	0.744	1.212	0.750	0.741	0.729	0.753	0.744	1.080	0.750	0.741	0.729
Average NMSE	0.748	0.743	0.853	0.746	0.738	0.724	0.748	0.742	0.820	0.746	0.736	0.724
	$\lambda_n = 5$						$\lambda_n = 10$					
Min NMSE	0.741	0.736	0.728	0.738	0.730	0.717	0.742	0.737	0.728	0.740	0.730	0.717
Max NMSE	0.748	0.847	0.905	0.746	0.736	0.726	0.749	0.757	0.864	0.746	0.736	0.726
Average NMSE	0.744	0.765	0.775	0.741	0.732	0.721	0.745	0.742	0.764	0.743	0.733	0.720

Table C-1: Aggregate Training NMSE with varying CP Ranks for $\rho = 0.6\%$

CP Rank	20	30	33	36	40	50	20	30	33	36	40	50
	$\lambda_n = 0$						$\lambda_n = 1$					
Min NMSE	0.799	0.798	0.787	0.797	0.798	0.794	0.800	0.797	0.788	0.796	0.796	0.794
Max NMSE	0.801	0.804	1.144	0.798	0.801	0.798	0.802	0.802	1.001	0.799	0.798	0.800
Average NMSE	0.800	0.801	0.879	0.798	0.800	0.796	0.801	0.800	0.843	0.798	0.798	0.797
	$\lambda_n = 5$						$\lambda_n = 10$					
Min NMSE	0.798	0.796	0.787	0.794	0.795	0.792	0.798	0.795	0.787	0.795	0.795	0.792
Max NMSE	0.800	0.886	0.890	0.796	0.799	0.799	0.802	0.812	0.870	0.798	0.800	0.798
Average NMSE	0.799	0.821	0.816	0.795	0.798	0.796	0.800	0.802	0.810	0.796	0.798	0.795

Table C-2: Aggregate Testing NMSE with varying CP Ranks for $\rho = 0.6\%$

CP Rank	5	7	10	12	15	20	25	5	7	10	12	15	20	25
	$\lambda_n = 0$							$\lambda_n = 1$						
Min NMSE	0.739	0.699	0.721	0.716	0.705	0.721	0.691	0.739	0.699	0.721	0.717	0.706	0.722	0.692
Max NMSE	0.744	0.707	0.726	0.721	0.710	0.726	0.697	0.744	0.704	0.726	0.721	0.710	0.726	0.696
Average NMSE	0.741	0.701	0.723	0.718	0.707	0.723	0.694	0.741	0.701	0.723	0.719	0.707	0.723	0.693
	$\lambda_n = 5$							$\lambda_n = 10$						
Min NMSE	0.737	0.707	0.718	0.713	0.702	0.730	0.690	0.736	0.707	0.717	0.712	0.703	0.730	0.689
Max NMSE	0.740	0.712	0.722	0.717	0.708	0.736	0.693	0.741	0.712	0.723	0.716	0.707	0.734	0.694
Average NMSE	0.739	0.709	0.720	0.715	0.704	0.732	0.691	0.739	0.709	0.720	0.714	0.704	0.732	0.691

Table C-3: Aggregate Training NMSE with varying CP Ranks for $\rho = 2.5\%$

CP Rank	5	7	10	12	15	18	20	5	7	10	12	15	18	20
	$\lambda_n = 0$							$\lambda_n = 1$						
Min NMSE	0.761	0.745	0.746	0.749	0.740	0.755	0.739	0.761	0.745	0.745	0.749	0.741	0.756	0.739
Max NMSE	0.763	0.747	0.748	0.750	0.743	0.756	0.742	0.763	0.748	0.749	0.751	0.743	0.757	0.741
Average NMSE	0.762	0.746	0.747	0.749	0.742	0.756	0.740	0.762	0.747	0.747	0.750	0.742	0.756	0.740
	$\lambda_n = 5$							$\lambda_n = 10$						
Min NMSE	0.759	0.749	0.744	0.747	0.740	0.759	0.738	0.758	0.749	0.743	0.747	0.741	0.759	0.739
Max NMSE	0.762	0.750	0.746	0.750	0.741	0.762	0.742	0.762	0.750	0.746	0.748	0.743	0.761	0.741
Average NMSE	0.760	0.750	0.745	0.748	0.741	0.760	0.740	0.760	0.750	0.745	0.747	0.742	0.760	0.740

Table C-4: Aggregate Testing NMSE with varying CP Ranks for $\rho = 2.5\%$

C-2 Influence of Regularization

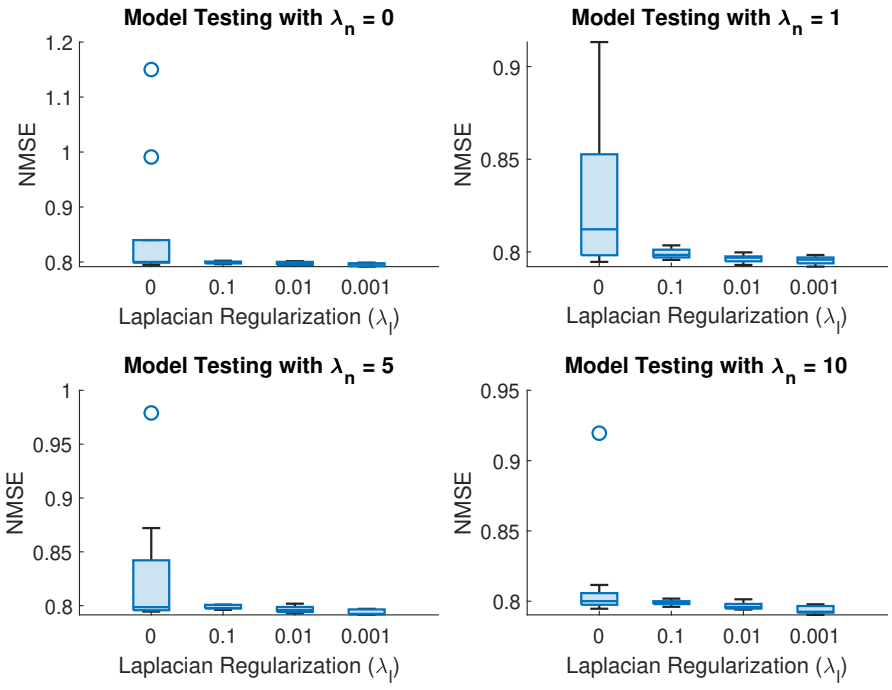

Figure C-5: Influence of varying Laplacian Regularization λ_l on Testing NMSE for $\rho = 0.6\%$

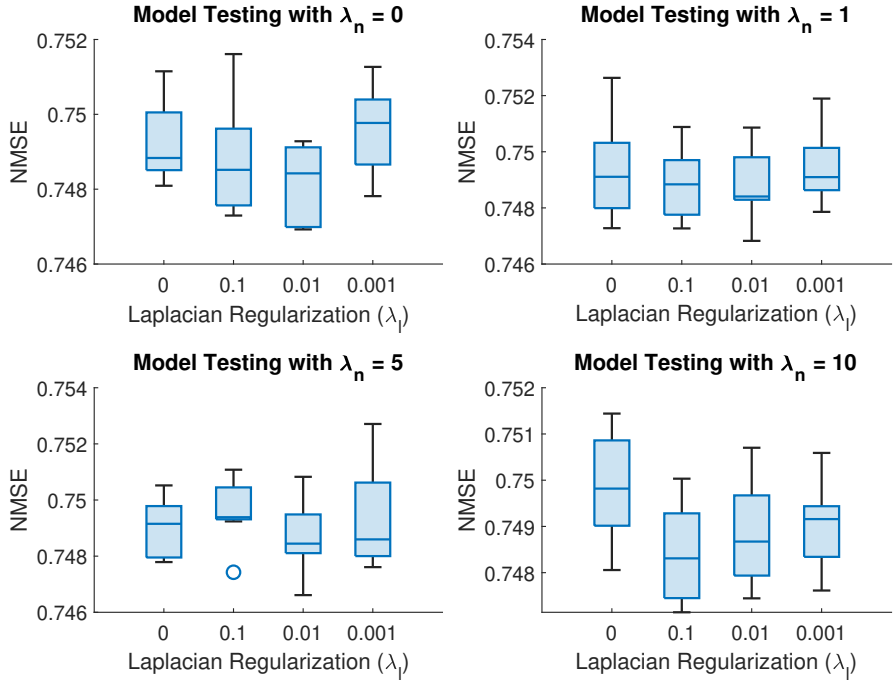

Figure C-6: Influence of varying Laplacian Regularization λ_l on Testing NMSE for $\rho = 2.5\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min NMSE	0.720	0.721	0.725	0.729	0.720	0.723	0.723	0.729
Max NMSE	1.212	0.745	0.749	0.752	1.080	0.746	0.748	0.753
Average NMSE	0.816	0.738	0.739	0.741	0.794	0.738	0.738	0.741
	$\lambda_n = 5$				$\lambda_n = 10$			
Min NMSE	0.717	0.720	0.723	0.726	0.717	0.719	0.721	0.726
Max NMSE	0.905	0.742	0.745	0.748	0.864	0.743	0.745	0.749
Average NMSE	0.780	0.735	0.735	0.737	0.758	0.735	0.735	0.737

Table C-5: Aggregate Training NMSE with varying regularization λ_l for $\rho = 0.6\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min NMSE	0.797	0.794	0.792	0.787	0.796	0.794	0.790	0.788
Max NMSE	1.144	0.803	0.801	0.799	1.001	0.802	0.800	0.801
Average NMSE	0.857	0.799	0.798	0.795	0.833	0.799	0.797	0.795
	$\lambda_n = 5$				$\lambda_n = 10$			
Min NMSE	0.795	0.795	0.792	0.787	0.796	0.794	0.790	0.787
Max NMSE	0.890	0.802	0.800	0.798	0.870	0.802	0.799	0.798
Average NMSE	0.828	0.799	0.797	0.794	0.813	0.799	0.797	0.794

Table C-6: Aggregate Testing NMSE with varying regularization λ_l for $\rho = 0.6\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min NMSE	0.691	0.693	0.693	0.697	0.692	0.692	0.693	0.696
Max NMSE	0.740	0.739	0.741	0.744	0.739	0.740	0.742	0.744
Average NMSE	0.714	0.714	0.715	0.719	0.714	0.714	0.715	0.718
	$\lambda_n = 5$				$\lambda_n = 10$			
Min NMSE	0.690	0.691	0.690	0.693	0.689	0.689	0.690	0.694
Max NMSE	0.737	0.739	0.738	0.740	0.739	0.736	0.738	0.741
Average NMSE	0.714	0.715	0.715	0.718	0.715	0.713	0.715	0.718

Table C-7: Aggregate Training NMSE with varying regularization λ_l for $\rho = 2.5\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min NMSE	0.741	0.742	0.739	0.739	0.741	0.741	0.739	0.739
Max NMSE	0.762	0.761	0.762	0.763	0.761	0.761	0.762	0.763
Average NMSE	0.749	0.749	0.748	0.750	0.749	0.749	0.749	0.749
	$\lambda_n = 5$				$\lambda_n = 10$			
Min NMSE	0.741	0.740	0.738	0.738	0.741	0.740	0.739	0.739
Max NMSE	0.760	0.762	0.759	0.762	0.761	0.759	0.761	0.762
Average NMSE	0.749	0.750	0.749	0.749	0.750	0.748	0.749	0.749

Table C-8: Aggregate Testing NMSE with varying regularization λ_l for $\rho = 2.5\%$

C-3 Performance Evaluation Metrics

C-3-1 Relative Reconstruction Error (RRE) and POF

CP Rank	**20**	**30**	**33**	**36**	**40**	**50**	**20**	**30**	**33**	**36**	**40**	**50**
	$\lambda_n = 0$						$\lambda_n = 1$					
Min POF	0.153	0.158	-1.829	0.153	0.161	0.172	0.153	0.158	-2.062	0.153	0.162	0.172
Max POF	0.160	0.159	0.163	0.158	0.166	0.180	0.159	0.160	0.163	0.158	0.167	0.179
Average POF	0.157	0.158	-0.338	0.156	0.164	0.177	0.157	0.159	-0.397	0.156	0.165	0.176
	$\lambda_n = 5$						$\lambda_n = 10$					
Min POF	0.157	-0.400	-1.154	0.156	0.166	0.174	0.156	-0.122	-0.736	0.156	0.165	0.174
Max POF	0.162	0.164	0.165	0.161	0.171	0.182	0.162	0.163	0.164	0.161	0.171	0.182
Average POF	0.160	0.022	-0.168	0.159	0.169	0.179	0.159	0.091	-0.064	0.159	0.169	0.179

Table C-9: Aggregate POF with varying CP Rank for $\rho = 0.6\%$

CP Rank	**5**	**7**	**10**	**12**	**15**	**20**	**25**	**5**	**7**	**10**	**12**	**15**	**20**	**25**
	$\lambda_n = 0$							$\lambda_n = 1$						
Min POF	0.152	0.181	0.160	0.167	0.175	0.165	0.187	0.152	0.182	0.160	0.166	0.175	0.165	0.188
Max POF	0.156	0.187	0.163	0.170	0.181	0.170	0.194	0.156	0.187	0.164	0.169	0.181	0.170	0.194
Average POF	0.154	0.185	0.162	0.169	0.179	0.168	0.191	0.154	0.185	0.162	0.168	0.179	0.168	0.191
	$\lambda_n = 5$							$\lambda_n = 10$						
Min POF	0.155	0.176	0.163	0.169	0.177	0.158	0.190	0.154	0.175	0.162	0.170	0.178	0.159	0.190
Max POF	0.158	0.181	0.165	0.172	0.184	0.163	0.196	0.158	0.181	0.166	0.173	0.183	0.163	0.196
Average POF	0.156	0.179	0.164	0.171	0.181	0.161	0.193	0.156	0.179	0.164	0.172	0.181	0.161	0.194

Table C-10: Aggregate POF with varying CP-Rank for $\rho = 2.5\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min POF	-1.829	0.153	0.156	0.153	-2.062	0.152	0.156	0.153
Max POF	0.180	0.179	0.176	0.172	0.179	0.178	0.176	0.172
Average POF	-0.168	0.162	0.162	0.160	-0.207	0.162	0.163	0.160
	$\lambda_n = 5$				$\lambda_n = 10$			
Min POF	-1.154	0.155	0.159	0.156	-0.736	0.155	0.158	0.156
Max POF	0.182	0.180	0.178	0.174	0.182	0.181	0.179	0.174
Average POF	-0.146	0.165	0.165	0.163	-0.030	0.165	0.165	0.163

Table C-11: Aggregate POF with varying regularization λ_l for $\rho = 0.6\%$

Regularization λ_l	**0**	**0.1**	**0.01**	**0.001**	**0**	**0.1**	**0.01**	**0.001**
	$\lambda_n = 0$				$\lambda_n = 1$			
Min POF	0.155	0.156	0.154	0.152	0.156	0.155	0.153	0.152
Max POF	0.194	0.193	0.191	0.187	0.194	0.193	0.191	0.188
Average POF	0.174	0.174	0.173	0.170	0.174	0.174	0.172	0.170
	$\lambda_n = 5$				$\lambda_n = 10$			
Min POF	0.158	0.156	0.156	0.155	0.156	0.158	0.156	0.154
Max POF	0.196	0.195	0.193	0.190	0.196	0.196	0.193	0.190
Average POF	0.174	0.174	0.172	0.170	0.173	0.174	0.172	0.170

Table C-12: Aggregate POF with varying regularization λ_l for $\rho = 2.5\%$

C-4 Scalability

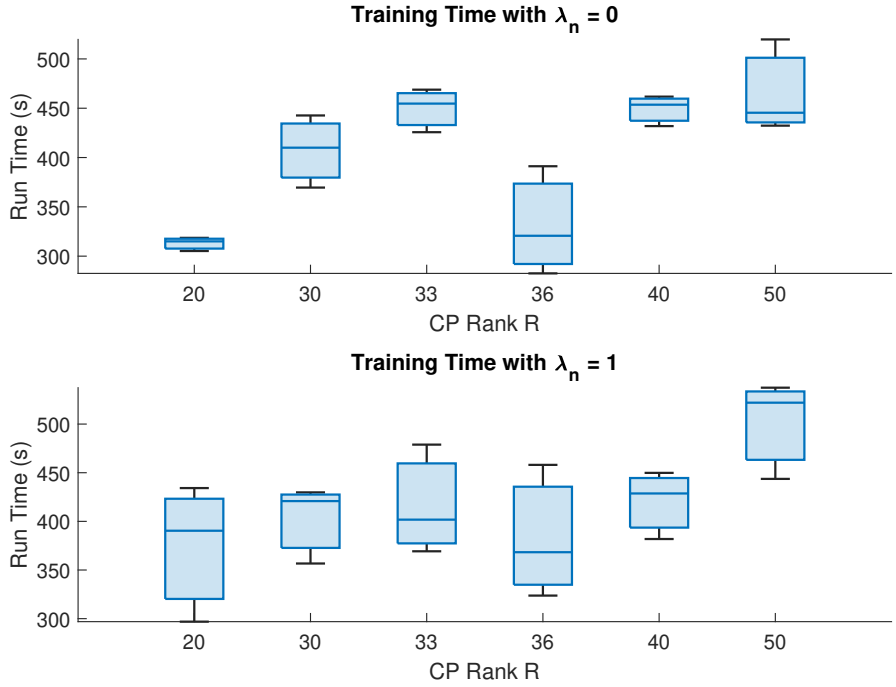

Figure C-7: Algorithm Run Times with varying CP rank for $\rho = 0.3\%$

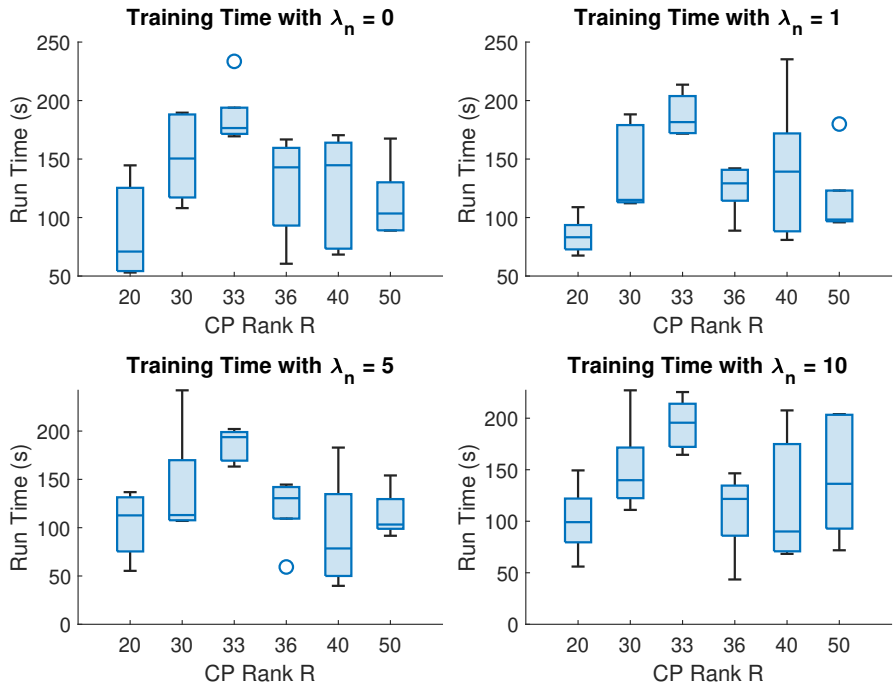

Figure C-8: Algorithm Run Times with varying CP rank for $\rho = 0.6\%$

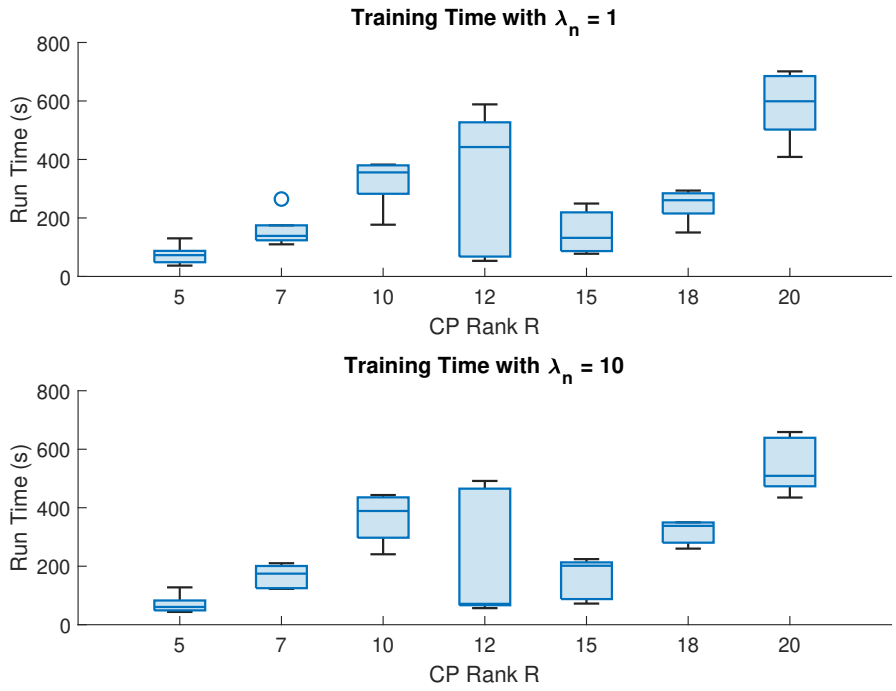

Figure C-9: Algorithm Run Times with varying CP rank for $\rho = 0.8\%$

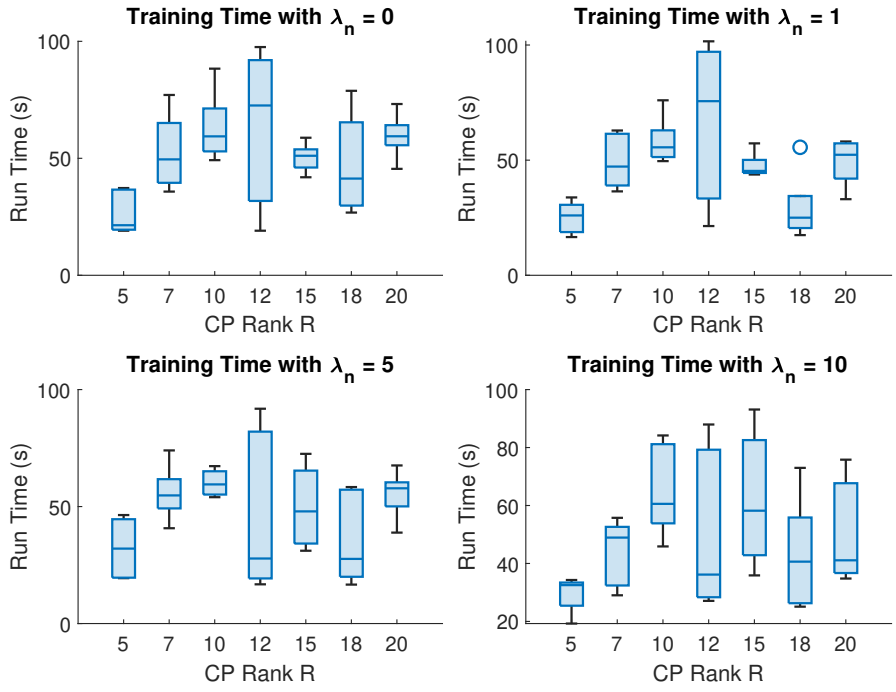

Figure C-10: Algorithm Run Times with varying CP rank for $\rho = 2.5\%$

Bibliography

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, pp. 455–500, 2009.

[2] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 115–153, 2001.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, 2005.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[5] D. C. Anastasiu, E. Christakopoulou, S. Smith, M. Sharma, and G. Karypis, "Big data and recommender systems," *Novática: Journal of the Spanish Computer Scientist Association*, October 2016.

[6] J. Bennett, S. Lanning, *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007, p. 35, New York, 2007.

[7] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 116–142, 2004.

[8] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*, pp. 291–324, Springer, 2007.

[9] P. Symeonidis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos, "Collaborative recommender systems: Combining effectiveness and efficiency," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2995–3013, 2008.

[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on computer supported cooperative work*, pp. 175–186, 1994.

[11] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Transactions on Signal Processing*, vol. 66, no. 19, pp. 5066–5081, 2018.

[12] F. R. Chung, *Spectral graph theory*, vol. 92. American Mathematical Soc., 1997.

[13] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[14] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[15] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Investigation of various matrix factorization methods for large recommender systems," *2008 IEEE International Conference on Data Mining Workshops*, pp. 553–562, 2008.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[17] Y.-X. Wang and Y. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[18] P. Symeonidis and A. Zioupos, *Matrix and tensor factorization techniques for recommender systems*, vol. 1. Springer, 2016.

[19] A. Bhaskara, M. Charikar, and A. Vijayaraghavan, "Uniqueness of tensor decompositions with applications to polynomial identifiability," in *Conference on Learning Theory*, pp. 742–778, PMLR, 2014.

[20] T. Anwar and V. Uma, "A review of recommender system and related dimensions," *Data, Engineering and Applications: Volume 1*, pp. 3–10, 2019.

[21] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proceedings of the 2010 SIAM international conference on data mining*, pp. 199–210, SIAM, 2010.

[22] E. Frolov and I. Oseledets, "Tensor methods and recommender systems," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 3, p. e1201, 2017.

[23] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 287–296, 2011.

[24] S. Friedland and L.-H. Lim, "Nuclear norm of higher-order tensors," *Mathematics of Computation*, vol. 87, no. 311, pp. 1255–1281, 2018.

[25] K. Batselier, "Low-rank tensor decompositions for nonlinear system identification: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 42, no. 1, pp. 54–74, 2022.

[26] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[27] F. L. Hitchcock, "Multiple invariants and generalized rank of a p-way matrix or tensor," *Journal of Mathematics and Physics*, vol. 7, no. 1-4, pp. 39–79, 1928.

[28] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l_r,l_r,1) terms, and a new generalization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, 2013.

[29] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 23, no. 7-8, pp. 393–405, 2009.

[30] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, pp. 83–98, 2013.

[31] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, 2017.

[32] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis, "Graph kernels: A survey," *Journal of Artificial Intelligence Research*, vol. 72, pp. 943–1027, 2021.

[33] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saerens, "An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification," *Neural Networks*, vol. 31, pp. 53–72, 7 2012.

[34] M. Yang, M. Coutino, G. Leus, and E. Isufi, "Node-adaptive regularization for graph signal reconstruction," *IEEE Open Journal of Signal Processing*, vol. 2, pp. 85–98, 2 2021.

[35] A. Uschmajew, "Local convergence of the alternating least squares algorithm for canonical tensor approximation," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 2, pp. 639–652, 2012.

[36] E. Hale and A. Prater-Bennette, "Comparison of cp and tucker tensor decomposition algorithms," in *Big Data III: Learning, Analytics, and Applications*, vol. 11730, pp. 68–85, SPIE, 2021.

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

[38] N. Singh, L. Ma, H. Yang, and E. Solomonik, "Comparison of accuracy and scalability of gauss–newton and alternating least squares for candecomc/parafac decomposition," *SIAM Journal on Scientific Computing*, vol. 43, no. 4, pp. C290–C311, 2021.

[39] L. Hu, S. Jian, L. Cao, Z. Gu, Q. Chen, and A. Amirbekyan, "Hers: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3830–3837, 2019.

[40] G. F. Lu, Y. Wang, and J. Zou, "Low-rank matrix factorization with adaptive graph regularizer," *IEEE Transactions on Image Processing*, vol. 25, pp. 2196–2205, 5 2016.

[41] Y. Su, X. Bai, W. Li, P. Jing, J. Zhang, and J. Liu, "Graph regularized low-rank tensor representation for feature selection," *Journal of Visual Communication and Image Representation*, vol. 56, pp. 234–244, 10 2018.

[42] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach," in *Twenty-fourth AAAI conference on Artificial Intelligence*, 2010.

[43] L. Yao, Q. Z. Sheng, Y. Qin, X. Wang, A. Shemshadi, and Q. He, "Context-aware point-of-interest recommendation using tensor factorization with social regularization," *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[44] F. M. Almutairi, N. Sidiropoulos, and G. Karypis, "Context-aware recommendation-based learning analytics using tensor and coupled matrix factorization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 729–741, 2017.

[45] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[46] J. Zhao, W. Wang, Z. Zhang, Q. Sun, H. Huo, L. Qu, and S. Zheng, "Trusttf: A tensor factorization model using user trust and implicit feedback for context-aware recommender systems," *Knowledge-Based Systems*, vol. 209, p. 106434, 2020.

[47] J. Zhao, S. Yang, H. Huo, Q. Sun, and X. Geng, "Tbtf: an effective time-varying bias tensor factorization algorithm for recommender system," *Applied Intelligence*, vol. 51, no. 7, pp. 4933–4944, 2021.

[48] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[49] C. Groetsch, "The theory of tikhonov regularization for fredholm equations," *104p, Boston Pitman Publication*, 1984.

[50] M. Fuhry and L. Reichel, "A new tikhonov regularization method," *Numerical Algorithms*, vol. 59, no. 3, pp. 433–445, 2012.

[51] P.-Y. Chen and S. Liu, "Bias-variance tradeoff of graph laplacian regularizer," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1118–1122, 2017.

[52] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2010.

[53] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[54] B. Jiang, C. Ding, J. Tang, and B. Luo, "Image representation and learning with graph-laplacian tucker tensor decomposition," *IEEE Transactions on Cybernetics*, vol. 49, pp. 1417–1426, 2019.

[55] V. N. Ioannidis, A. S. Zamzam, G. B. Giannakis, and N. Sidiropoulos, "Coupled graphs and tensor factorization for recommender systems and community detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, pp. 909–920, 2021.

[56] N. Shahid, F. Grassi, and P. Vandergheynst, "Tensor robust pca on graphs," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5406–5410, 2019.

[57] S. Hu, "Relations of the nuclear norm of a tensor and its matrix flattenings," *Linear Algebra and its Applications*, vol. 478, pp. 188–199, 2015.

[58] C. Lu, X. Peng, and Y. Wei, "Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5996–6004, 2019.

[59] G. Polak, E. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 3, no. R1, pp. 35–43, 1969.

[60] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, pp. 149–154, 1 1964.

[61] E. Polak, *Computational Methods in Optimization : A Unified Approach*, vol. 77. New York: Academic Press, 1971.

[62] L. Scales, *Introduction to Non-linear Optimization*. Springer-Verlag, 1985.

[63] A. P. da Silva, P. Comon, and A. L. de Almeida, "An iterative deflation algorithm for exact cp tensor decomposition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3961–3965, IEEE, 2015.

[64] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[65] H. K. Khalil, *Nonlinear control*, vol. 406. Pearson New York, 2015.

[66] J. Bolte, A. Daniilidis, and A. Lewis, "The lojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 1205–1223, 2007.

[67] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "Gspbox: A toolbox for signal processing on graphs," 2014.

[68] P. G. Age Smilde, Rasmus Bro, *Multi-Way Analysis with Applications in the Chemical Sciences*, ch. 6, pp. 111–144. John Wiley & Sons, Ltd, 2004.

[69] S. E. Umbaugh, *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools, Second Edition*. USA: CRC Press, Inc., 2nd ed., 2010.

[70] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[71] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TIIS)*, vol. 5, no. 4, pp. 1–19, 2015.

[72] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 1)." https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.

Glossary

List of Acronyms

RS Recommender Systems

CF Collaborative Filtering

kNN k Nearest Neighbours

MF Matrix Factorization

PCA Principal Component Analysis

SVD Singular Value Decomposition

CP CANDECOMP/PARAFAC

GRCP Graph Regularized CP Tensor Decomposition

GLR Graph Laplacian Regularizer

ALS Alternating Least Squares

CG Conjugate Gradient

ADMM Alternating Direction Method of Multipliers

MTTKRP Matricized Khatri-Rao Product

NMSE Normalized Mean Square Error

RRE Relative Reconstruction Error

POF Percentage of Fit

KL Kurdyka-Lojasiewicz

SNR Signal to Noise Ratio

RGB Red, Green, Blue

GSP Graph Signal Processing

CPU Central Processing Unit

GPU Graphics Processing Unit