POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Symmetric Probabilistic Matrix Factorization Techniques for Recommender Systems

ABDOLRASOUL BAHARIFARD
Département de Mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques appliquées

Août 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Symmetric Probabilistic Matrix Factorization Techniques for Recommender Systems

présentée par **Abdolrasoul BAHARIFARD**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Soumaya YACOUT, présidente
Luc ADJENGUE, membre et directeur de recherche
Richard LABIB, membre
Mohammad Sajjad GHAEMI, membre externe

DEDICATION

To my beloved wife, Farnaz Ebrahimi, my constant support and endless love ...

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my thesis supervisor, Dr. Luc Adjengue, for his unwavering support and guidance throughout my academic journey. Dr. Adjengue provided me with invaluable insights and expertise, which has been instrumental in shaping the direction of my research. His encouragement and support have been a source of inspiration and motivation, and I am deeply grateful for his contributions to my thesis.

Without Dr. Adjengue's support, I would not have been able to navigate the challenges of my research and academic journey. His expertise has helped me to approach my thesis with clarity and focus, and his encouragement has been a source of motivation when I needed it most. I am truly grateful for the impact that Dr. Adjengue has had on my academic journey, and I will always remember the lessons and insights he has shared with me.

In addition to my supervisor, I would also like to extend my heartfelt thanks to my wife, Farnaz Ebrahimi, for her love, support, and encouragement. She has been my rock throughout the ups and downs of my academic journey, and I am truly grateful for her unwavering support. Her presence in my life has been a source of inspiration, and I would not have been able to achieve this milestone without her.

In this statement, I express my gratitude to all the individuals who have supported me throughout my academic journey. I note that their support, encouragement, and guidance have been essential in helping me to achieve this milestone, and I am deeply grateful for their contributions to my thesis.

Throughout my academic journey, I have been fortunate to have the support of a community of individuals who have encouraged and inspired me. This community includes my family, friends, professors, and colleagues, all of whom have played a role in helping me to stay focused, motivated, and committed to my goals.

In particular, my family has been a constant source of support and encouragement. Their belief in my abilities has been instrumental in helping me to pursue my academic dreams, and their unwavering support has given me the strength to persevere through difficult times.

Overall, I am deeply grateful to all those who have supported me throughout my academic journey. Their contributions have been essential in helping me to achieve this milestone, and I will always remember their support and encouragement with deep gratitude.

RÉSUMÉ

La multitude d'options de quoi disponibles pour les consommateurs modernes peut être écrasante. Les détaillants électroniques et les sites Web de commerce électronique offrent une vaste gamme de produits pour répondre à un large éventail de besoins et de préférences. L'alignement des exigences des consommateurs avec les produits les plus appropriés est un aspect essentiel pour l'amélioration de la satisfaction des utilisateurs et de la fidélisation de la clientèle. En conséquence, un nombre croissant d'entreprises investissent dans des systèmes de recommandation afin d'identifier les modèles d'intérêt des consommateurs pour leurs produits et de proposer des recommandations qui correspondent aux besoins et aux préférences de l'utilisateur.

Les systèmes de recommandation sont utilisés pour mettre en relation les consommateurs avec les produits qu'ils sont susceptibles d'aimer. Il existe deux principaux types de systèmes de recommandation : le filtrage basé sur le contenu et le filtrage collaboratif. Le filtrage basé sur le contenu utilise les préférences d'une personne pour suggérer de nouveaux produits. Le filtrage collaboratif, quant à lui, utilise des informations provenant d'autres personnes ayant des préférences similaires pour faire des recommandations. Il s'appuie sur les choix effectués par un groupe de personnes pour suggérer de nouveaux produits.

Une classe d'algorithmes de filtrage collaboratif utilisée dans les systèmes de recommandation est la factorisation matricielle. Ces algorithmes décomposent la matrice d'interaction entre utilisateurs et articles en produit de deux matrices de dimension inférieure, qui sont plus petites en termes de dimensionnalité. Cette approche se approxime de la matrice d'origine en trouvant une combinaison de ces deux matrices aussi proche que possible de la matrice d'origine, basée sur la minimisation de l'erreur entre l'original et le produit des deux matrices plus petites, généralement en utilisant des techniques d'optimisation telles que les moindres carrés. Il s'agit d'un problème très mal posé car la factorisation peut être réalisée de nombreuses manières, conduisant à une infinité de solutions possibles sans une réponse correcte unique. Par conséquent, les chercheurs doivent formuler des hypothèses raisonnables (telles que des contraintes de régularisation ou des connaissances supplémentaires dans le domaine) concernant la factorisation afin de trouver la solution la mieux adaptée à leurs besoins.

Dans cette recherche, nous présentons la factorisation matricielle probabiliste symétrique

(SPMF) à travers la création d'une matrice de blocs symétrique à l'aide de la matrice d'interaction entre utilisateurs et articles. SPMF utilise une méthodologie probabiliste pour traiter les valeurs manquantes dans la matrice d'interaction et génère des recommandations en estimant la distribution de probabilité des préférences de l'utilisateur et des caractéristiques des articles en fonction des interactions observées.

SPMF simplifie le problème de factorisation en transformant la fonction objectif en un problème convexe, garantissant un optimal global. Cette transformation conduit à une solution optimale avec un taux de convergence plus rapide, rendant le processus d'optimisation plus efficace. Dans cette nouvelle approche, les deux matrices originales de factorisation sont fusionnées en une seule matrice, ce qui rend la solution plus facile à trouver puisque le problème complexe d'optimisation à deux facteurs est réduit à une seule fonction de coût. Cette nouvelle méthode d'optimisation possède non seulement une solution optimale globale unique et est convexe, mais elle nous permet également de comprendre simultanément les relations entre les utilisateurs, les éléments et les interactions entre eux.

Par ailleurs, nous étudions une variante de SPMF conçue pour factoriser des matrices binaires, qui a des applications pertinentes dans le domaine des systèmes de recommandation. Dans cette approche, les relations utilisateur-utilisateur et article-article sont conceptualisées sous forme de graphe, où la présence d'une arête signifie une relation étroite entre la paire. La matrice d'interaction entre utilisateurs et articles est transformée de telle sorte qu'un article est recommandé si l'entrée correspondante est 1, et non recommandé si l'entrée est 0. Nous appelons cette modification Factorisation matricielle binaire probabiliste symétrique (SPBMF). SPBMF est une méthode de factorisation robuste applicable à divers domaines, notamment le regroupement de graphes, l'analyse de l'expression génique, le traitement d'images, l'analyse des réseaux sociaux, les réseaux de capteurs et les systèmes de recommandation. Cette variante de SPMF démontre l'avantage significatif d'une approche probabiliste, qui est un thème central de cette thèse. Il offre la flexibilité nécessaire pour adapter le cadre à n'importe quel type de données en ajustant de manière appropriée une distribution de probabilité pertinente pour l'adapter à la source de données sous-jacente.

De plus, nous avons étudié SPMF en le testant sur des sous-ensembles de l'ensemble de données MovieLens 100K et FilmTrust. Les résultats de cette étude contribuent au développement de systèmes de recommandation et seront précieux pour les chercheurs, les praticiens et les acteurs de l'industrie dans le domaine des systèmes de recommandation.

ABSTRACT

The multitude of options of what available to modern consumers can be overwhelming. Electronic retailers and e-commerce websites offer a vast array of products to cater to a diverse range of needs and preferences. Aligning consumer requirements with the most suitable products is a critical aspect in enhancing user satisfaction and building customer loyalty. As a result, an increasing number of businesses are investing in recommendation systems to identify patterns of consumer interest in their products and offer recommendations that align with a user's needs and preferences.

Recommendation systems are used to match consumers with products they might like. There are two main types of recommendation systems: content-based filtering and collaborative filtering. Content-based filtering uses a person's preferences to suggest new products. Collaborative filtering, on the other hand, uses information from other people with similar preferences to make recommendations. It looks at the choices made by a group of people to suggest new products.

A class of collaborative filtering algorithms used in recommendation systems is matrix factorization. These algorithms decompose the user-item interaction (rating) matrix into the product of two lower-dimensional matrices, which are smaller in terms of their dimensionality. This approach approximates the original matrix by finding a combination of these two matrices that is as close as possible to the original matrix, based on minimizing the error between the original and the product of the two smaller matrices, typically using optimization techniques like least squares. This is a highly ill-posed problem because the factorization can be achieved in numerous ways, leading to infinitely many possible solutions without a unique correct answer. Therefore, researchers must make reasonable assumptions (such as regularization constraints or additional domain knowledge) about the factorization to find a solution that works best for their needs.

In this research, we present Symmetric Probabilistic Matrix Factorization (SPMF) through the creation of a symmetric block matrix using the user-item rating matrix. SPMF utilizes a probabilistic methodology to address missing values in the interaction matrix and generates recommendations by estimating the probability distribution of user preferences and item features based on observed interactions.

SPMF simplifies the factorization problem by transforming the objective function into a convex problem, ensuring a global optimum. This transformation leads to an optimal solution with a faster convergence rate, making the optimization process more efficient. In this new approach, the original two matrices of the factorization are merged into a single matrix, making the solution easier to find as the complex two-factor optimization problem is reduced to a single cost function. This new optimization method not only has a unique global optimal solution and is convex, but it also allows us to simultaneously understand the relationships between users, items, and the interactions between them.

Furthermore, we investigate a variant of SPMF designed to factorize binary matrices, which has relevant applications in the recommendation system domain. In this approach, user-user and item-item relationships are conceptualized as a graph, where the presence of an edge signifies a close relationship between the pair. The user-item rating matrix is transformed such that an item is recommended if the corresponding entry is 1, and not recommended if the entry is 0. We refer to this modification as Symmetric Probabilistic Binary Matrix Factorization (SPBMF). SPBMF is a robust factorization method applicable to various fields, including graph clustering, gene expression analysis, image processing, social network analysis, sensor networks, and recommendation systems. This variant of SPMF demonstrates the significant advantage of a probabilistic approach, which is a central theme of this thesis. It provides the flexibility to adapt the framework to any data type by appropriately tuning a relevant probability distribution to fit the underlying data source.

Additionally, we studied SPMF by testing it against subsets of the MovieLens 100K and FilmTrust dataset. The results of this study contribute to the development of recommendation systems and will be valuable for researchers, practitioners, and industry stakeholders in the field of recommendation systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

RS	Recommendation Systems
CF	Collaborative Filtering
CB	Content-Based
RMSE	Root Mean Squared Error
SVD	Singular Value Decomposition
MF	Matrix Factorization
SMF	Symmetric Matrix Factorization
TF	Tensor Factorization
NMF	Non-negative Matrix Factorization
SNMF	Symmetric Non-Negative Matrix Factorization
PMF	Probabilistic Matrix Factorization
SPMF	Symmetric Probabilistic Matrix Factorization
SPBMF	Symmetric Probabilistic Binary Matrix Factorization
BPMF	Bayesian Probabilistic Matrix Factorization
GRPMF	Graph Regularized Probabilistic Matrix Factorization
PMFDL	Probabilistic Matrix Factorization Recommendation Method based on Deep Learning
DBPMF	Deep Bias Probabilistic Matrix Factorization
ALS	Alternating Least Squares
ANLS	Alternating Nonnegative Least Squares
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
MSRA	Multi-step Resource Allocation
TPR	True Positive Rate
FPR	False Positive Rate
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
MF-MSI	Matrix Factorization with Multimodal Side Information
PLSA	Probabilistic Latent Semantic Analysis

LIST OF NOTATIONS

m	Number of users
n	Number of items
R	$m \times n$ Rating Matrix
r_{ui}	Rating of user u on item i
U	User-features Matrix
V	Item-features Matrix
k	Number of latent factors
r	Rank of the user-item rating matrix R
K	Number of folds in K-fold cross-validation
λ_i	eigenvalues of matrix UU^T
$\|.\|_F$	Frobenius norm
$\|.\|_2$	L_2 norm
R_{RN}	Row-normalized user-item rating matrix R
R_{CN}	Column-normalized user-item rating matrix R
S	Symmetric block matrix
$\mathcal{S}$	Normalized symmetric block matrix
$\mathfrak{s}_{ij}$	Any entry of $\mathcal{S}$

CHAPTER 1 INTRODUCTION

Nowadays, online shoppers not only expect to obtain all relevant information about shopping items but also anticipate receiving some guidance to make the right shopping decision with that huge amount of information. With current technology advancements, Internet searches are the first choice for shoppers to collect information about products and services, but most of the time they end up with a vast amount of irrelevant information. This situation is known as "information overload" [1]. ***Recommendation Systems (RS)*** are designed to overcome the problem of information overload and filter out unrelated information.

In this thesis, we aim to explore the various types of RS algorithms and how they can be tailored to fit the needs of different types of users and products. Our goal is to develop a RS that can provide accurate recommendations to online shoppers, thereby enhancing their shopping experience and increasing their likelihood of making a purchase.

To address the problem of information overload for online shoppers, the aim of this thesis is to develop a recommendation system that provides personalized recommendations to users. To achieve this goal, a comprehensive review of the literature on RS and information overload will be conducted, with the aim of identifying the key challenges and opportunities for RS in addressing this problem. This will form the background of the research.

After the background section, the problem definition will follow, which will highlight the importance of mitigating information overload for online shoppers and the potential benefits of using RS to address this problem. The research objectives will be defined, which are to identify the approaches for developing RS capable of providing accurate and personalized recommendations to online shoppers. To achieve these objectives, a set of experiments will be conducted to evaluate the effectiveness of different RS algorithms in diverse scenarios. These experiments will allow a comprehensive assessment of the algorithms performance.

1.1 Background

RS are a set of tools and techniques that provide suggestions for items that might be useful or bought by users. The recommendations are directed at individuals and consumed by users who do not have enough information or personal experiences to process and select from the enormous number of available alternatives offered by, say, a website. In the content streaming industry, RS assist a user in selecting and watching a movie based on his or her previous movie lists or the movie lists of similar users. With the fast-growing volume of online contents, RS have revealed an important impact on managing the information overload problem by providing more proactive and personalized information services to users. Indeed, they are applications to help users in a decision-making process where they want to choose one item amongst hundreds or thousands of products or services [1].

RS are among the most commercialised applications of artificial intelligence, having a considerable impact on the performance of the e-commerce industry [2]. Many e-commerce websites use them to provide personalized information and experiences to their customers [2]. According to a McKinsey report from 2013, RS accounted for 35% of all Amazon transactions [3] and Amazon has a roughly 60% conversion rate for the products it recommends! We define the conversion rate as the number of users that accept a recommendation divided by the number of all visitors who browse through the website. Similarly, with an annual budget of $150 millions just for recommendations, Netflix has managed to save $1 billion each year by increasing their retention rate [4]. The retention rate is the percentage of customers who remain loyal and subscribe on a regular basis. In addition, 75% of all content watched by Netflix viewers now comes from a recommendation [4]. Although RS suggest items based on an individual's preferences, they can also be used in a more general way to make websites and platforms more customer-centric. When people need to make a decision without having enough information on available alternatives, the natural action is to rely on the experience and opinion of others, which can be provided by RS.

Given the advantages of RS, it is no surprise that they have become an essential element in many web applications. One of the key benefits of RS is their ability to enhance the overall user experience by providing personalized recommendations that align with the user's interests and preferences. This interaction can lead to increased customer satisfaction and loyalty, ultimately resulting in higher revenue for businesses.

Another important advantage of RS is their ability to streamline the decision-making process by reducing the time and effort users need to spend searching for products or services that meet their needs. This is particularly useful in industries such as e-commerce, where users may be overwhelmed by the sheer volume of available options. With RS, users can easily find products or services that match their preferences and needs, without the need for extensive research or comparison.

In summary, RS are a critical tool for managing the issue of information overload, as well as providing a personalized user experience and improving the efficiency of decision-making. These benefits have made RS a valuable component of various industries, and their continued development and refinement will likely contribute to further growth in the future.

Now that we have reviewed the background and benefits of recommendation systems, it is important to consider the challenges that come with developing and implementing RS. One major challenge is ensuring that the recommendations provided to users are accurate and relevant to their interests and preferences. This requires the RS to have access to high-quality data and to use advanced algorithms and techniques to analyze this data and make personalized recommendations. Additionally, RS must be designed to handle large volumes of data and to scale effectively as the user base grows.

In the following sections, we will delve into these challenges in more detail and explore potential solutions and techniques for addressing them.

1.2 Problem Definition

The rapid growth of data due to advancements in sensors and storage technologies has led to an unprecedented influx of information in various domains, including e-commerce, digital media, and online services. This explosion of data has resulted in information overload, making it challenging for users to find relevant content. Recommendation Systems have emerged as essential tools to tackle this issue by providing personalized suggestions based on users' preferences and behaviors, thereby enhancing user experience and engagement.

Matrix factorization techniques have become fundamental in the development of recommendation systems due to their ability to uncover latent factors in user-item interactions. These latent factors help in predicting user preferences and generating recommendations. Among the most widely utilized matrix factorization techniques are Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), and Probabilistic Matrix Factorization (PMF). Each of these techniques offers advantages but also comes with limitations, which this research aims to address through the introduction of Symmetric Probabilistic Matrix

Factorization (SPMF). Below we explain the drawbacks of each model briefly to justify the introduction of SPMF.

Singular Value Decomposition (SVD)

SVD decomposes the user-item interaction matrix into three matrices, revealing latent factors that represent underlying patterns in the data. Despite its widespread use, SVD has some limitations:

- **Scalability**: SVD requires significant computational resources, making it less efficient for large-scale datasets [5].
- **Interpretability**: The latent factors produced by SVD are not inherently interpretable, which can complicate the understanding and explanation of the recommendations [6].

Non-negative Matrix Factorization (NMF)

NMF factorizes the user-item matrix into two non-negative matrices, ensuring that the resulting factors are more interpretable. However, NMF also faces several challenges:

- **Uniqueness**: The factorization produced by NMF is not unique. Different runs with different initializations can produce different factor matrices, which can complicate the reproducibility and interpretation of results [7].
- **Complexity**: The optimization problem in NMF is non-convex, which complicates the convergence to a global optimum [7].

NMF, an ill-posed problem is one that lacks a unique and stable solution. NMF seeks to decompose a non-negative user-item rating matrix R into two non-negative matrices U and V such that $R \approx UV$. This factorization is inherently non-unique due to scaling and rotational ambiguities, where multiple pairs of matrices can equally well approximate R. Additionally, NMF is sensitive to perturbations in the data, leading to instability in the solutions. Small changes in the input matrix can result in significant variations in the resulting factorization, further complicating the interpretation of the results.

To address the ill-posed nature of NMF, several strategies can be employed. Regularization techniques, such as L_1 and L_2 regularization, help control the complexity and promote stability of the factorization. Careful initialization of the factor matrices and the incorporation of constraints or prior knowledge can reduce ambiguity and improve interpretability. Running the factorization multiple times with different initializations and aggregating the results can also lead to robust and consistent patterns. These approaches help mitigate the challenges raise by the ill-posed nature of NMF, enhancing the reliability of the decomposed matrices for practical applications.

Probabilistic Matrix Factorization (PMF)

PMF incorporates a probabilistic framework to matrix factorization, providing a probabilistic interpretation of the latent factors. Despite its advantages, PMF is not without its drawbacks:

- **Non-Convex Optimization**: The optimization problem in PMF is non-convex, meaning it can have multiple local minima. This makes it difficult to find the global minimum, and the solution can be sensitive to the initial values of the parameters [8].
- **Overfitting**: PMF is prone to overfitting without proper regularization [5].

PMF also suffers from ill-posedness, similar to NMF. PMF can lead to non-unique solutions due to multiple sets of latent variables that provide equally good explanations for the observed data. Additionally, PMF is sensitive to initial conditions and noise in the data, which can result in significant variations in the learned latent factors, making the solutions unstable.

To mitigate these issues, various strategies can be employed. Regularization terms in the PMF objective function help control model complexity and promote stability. Bayesian extensions of PMF, such as Bayesian Probabilistic Matrix Factorization (BPMF), incorporate prior distributions over latent factors, reducing ambiguity and incorporating prior knowledge. Robust optimization techniques and multiple initializations can further enhance the stability and reliability of the solutions. These approaches help address the ill-posed nature of PMF, leading to more consistent and interpretable factorization outcomes.

To address these limitations, we propose Symmetric Probabilistic Matrix Factorization (SPMF), which aims to provide a robust and interpretable factorization approach for recommendation systems. SPMF constructs a symmetric block matrix using the user-item rating matrix and estimates this matrix with a single low-rank matrix.

It has been noted that the term SPMF exists in the literature, typically applied to symmetric matrices, which restricts its applications. Similarly, other symmetric approaches such as

Symmetric Matrix Factorization (SMF) and Symmetric Non-negative Matrix Factorization (SNMF) are also confined to symmetric matrices. However, the SPMF model proposed in this thesis introduces several novel contributions that distinguish it from these existing methods and make it original. Some advantages are listed below:

- **Symmetrizing any arbitrary matrix and factorizing it**: Unlike traditional symmetric approaches, our SPMF model is designed to symmetrize any arbitrary rectangular matrix. This transformation broadens the applicability of the model beyond symmetric matrices, making it suitable for a wider range of real-world datasets that are typically not symmetric. By extending the application of SPMF to both symmetric and non-symmetric matrices, our model can be utilized in various domains, such as recommendation systems, social network analysis, and bioinformatics, where the data structure may not inherently be symmetric.

- **Probabilistic Flexibility**: We employ a probabilistic approach to factorize the symmetric matrix, introducing a new method for symmetric matrix factorization. The parametric nature of this probabilistic approach allows for the prediction of unseen data using the learned model, distinguishing it from nonparametric techniques that merely approximate a given matrix. Additionally, the choice of probabilistic distributions (e.g., Gaussian and Laplace) in SPMF can be tailored to improve the robustness of recommendations.

- **Transforming Biconvex Optimization into Convex Optimization**: SPMF reduces the two-factor biconvex optimization problem into a single-factor convex optimization problem with a unique global optimal solution. This transformation ensures a unique global optimal solution, to address the limitations of NMF and PMF.

- **Handling Sparsity**: SPMF deals with sparsity by utilizing a probabilistic methodology to estimate missing values and generate robust recommendations.

In this research, we will demonstrate how SPMF leverages the preferences of users with similar choices to create personalized recommendations. We will evaluate SPMF by assessing its performance on subsets of the MovieLens 100K and FilmTrust datasets, showcasing its capabilities with real-world data.

1.3 Research Questions

To address the challenges and opportunities presented by SPMF, this thesis aims to answer the following research questions:

1. What are the comparative advantages of SPMF over existing matrix factorization techniques such as SVD, NMF, and PMF?
2. How can we leverage user-user or item-item interactions into the user-item matrix? And what are the benefits of integrating these knowledge into one single matrix?
3. What are the benefits and trade-offs of transforming the biconvex optimization problem into a convex optimization problem in terms of computational complexity and interpretability?
4. How can the optimal number of latent factors be determined for the SPMF model in recommendation systems, and what impact does this number have on the model's performance?
5. What impact does the choice of probabilistic distribution (e.g., Gaussian and Laplace) in SPMF have on the robustness of recommendations?
6. How to binarize a user-item, item-item, and user-user rating matrix?
7. How to use the developed SPMF for other data structures such as an arbitrary binary matrix?
8. What role does hyperparameter tuning play in the effectiveness of SPMF, and what are the most critical parameters to optimize?
9. How does the proposed SPMF model improve recommendation robustness and interpretability compared to existing methods when evaluated on subsets of the MovieLens 100K and FilmTrust datasets?
10. What are the theoretical and mathematical advantages of SPMF compare to those of state-of-the-art matrix factorization techniques such as NMF and PMF?

By addressing these questions, this research aims to develop a robust SPMF-based recommendation systems that provides interpretable and personalized recommendations, thereby enhancing user experience and engagement on online platforms. This work will contribute to the ongoing development and refinement of recommendation systems by providing new insights into the application and optimization of SPMF.

1.4 Originality and Contribution

We introduce SPMF, a novel approach to matrix factorization in recommendation systems that addresses the limitations of existing techniques such as SVD, NMF, and PMF. The originality and contributions of this work are summarized as follows:

1. **Comparative Advantages of SPMF**: We systematically investigate the comparative advantages of SPMF over SVD, NMF, and PMF. In contrast to traditional factorization techniques, SPMF converts the original matrix into a symmetric blockwise matrix to circumvent the rank deficiency issues caused by overdetermination or underdetermination. Consequently, to address the non-uniqueness and ill-posed nature of NMF and PMF, SPMF approximates this blockwise matrix by incorporating user-user and item-item information into a single low-rank factor in order to solve a convex objective function. This approach not only provides an optimal solution with a faster convergence rate but also offers interpretability advantage inspired by the modeled user-user and item-item interactions from a probabilistic perspective. Moreover, SPMF is flexible enough to address any arbitrary data type by selecting a relevant distribution better suited for the given problem. For instance, factorizing binary matrices can be achieved by choosing a Bernoulli distribution to model the underlying data structure. This adaptability makes SPMF an interpretable and versatile tool for a wide range of matrix factorization applications.

2. **Unified Matrix Representation**: By using a single unified matrix in SPMF, we explore the benefits and trade-offs in terms of computational complexity and interpretability. The unified approach simplifies the factorization process and improves the interpretability of the latent factors. User-item rating matrices often face issues of overdetermination or underdetermination when there is a significant imbalance between the number of items and users. Such imbalances can complicate the interpretation of relationships within the data and lead to a matrix that lacks sufficient rank, resulting in computational inefficiencies and inaccurate recommendations. To enhance interpretability, we construct a blockwise symmetric matrix that incorporates both user-user and item-item relationships. This transformation converts the original rectangular user-item matrix into a square symmetric matrix, allowing us to capture and analyze the underlying interactions which leads to a convex single factor prediction. By mitigating the limitations posed by an overdetermined or underdetermined input matrix, this approach provides an insightful representation of the data, with the potential to enhance user-item prediction, alleviate the rank deficiency problem, and improve the

interpretability of the factorization process.

3. **Transforming biconvex to convex optimization**: By transforming a biconvex optimization problem into a convex optimization problem, we achieve several benefits, including ensuring a unique optimal solution and improved convergence properties. Additionally, by consolidating two distinct factors into a single unified low-rank factor, this technique enhances the interpretability of matrix factorization methods, providing a better understanding of the underlying data structure.

4. **Optimal Number of Latent Factors**: This research proposes methods to determine the optimal number of latent factors (k) for the SPMF model in recommendation systems. We investigate the impact of k on model performance, providing insights into optimal model tuning. Determining the optimal k is critical in matrix factorization-based recommendation systems. The number of latent factors directly affects the model's complexity and performance. A small k may fail to capture the underlying patterns in the data, resulting in poor recommendations. Conversely, a large k can lead to overfitting, where the model performs well on training data but poorly on unseen data. To identify the optimal k, we propose the following methodology: We employ K-fold cross-validation to evaluate the SPMF model's performance across different values of k. Model performance is assessed using metrics such as Root Mean Squared Error (RMSE), which provides a measure of the model's accuracy and recommendation relevance. We systematically vary k and record the corresponding performance metrics. By analyzing these results, we identify the k value that minimizes the performance metrics. Additionally, we assess model complexity and potential overfitting by comparing training and validation performance. A significant gap between these performances indicates overfitting, suggesting the need for a lower k. By following this methodology, we can systematically determine the optimal number of latent factors for the SPMF model, ensuring a balance between model complexity and performance.

5. **Impact of Probabilistic Distribution**: In this study, we explore the impact of different probabilistic distributions, such as Gaussian and Laplace, on the robustness of SPMF recommendations against noise and outliers. The normal distribution is suitable for capturing the central tendency of user ratings and is computationally efficient, but it is sensitive to outliers. The Laplace distribution, with its heavier tails, offers robustness to outliers and promotes sparsity in factorized matrices, making it a better choice for data with extreme values. The choice between these distributions depends on the dataset characteristics and the desired properties of the recommendation system. We propose methodologies involving Maximum Likelihood Estimation (MLE) and Bayesian

estimation with these distributions, ensuring a balance between model robustness and interpretability.

6. **Binarization of the Matrix**: In this research, we investigate a new approach to binarize an arbitrary user-rating matrix and its advantages for simplifying the factorization process. In the rapidly evolving landscape of AI-enabled digital entertainment, platforms like YouTube, Amazon, and Netflix have revolutionized video content consumption and advertising. The effectiveness of their recommendation systems is crucial for maximizing revenue through targeted advertisements by capturing and retaining user attention. Accurate predictions of user preferences enhance user engagement and advertisement delivery. A significant challenge lies in recommending suitable content from vast repositories, where user preferences are often represented as binary ratings (likes or dislikes). Developing a robust recommendation system for binary user-item rating matrices is essential for improving user satisfaction by delivering relevant content, thereby optimizing advertisement exposure and engagement.

7. **Symmetric Probabilistic Binary Matrix Factorization (SPBMF)**: Moreover, we modified our developed SPMF to factorize a binary matrix based on the Bernoulli distribution. This novel mitigation is called Symmetric Probabilistic Binary Matrix Factorization (SPBMF) that is inspired by the notion of logistic function where dot products values are transformed into $[0, 1]$ range by sigmoid function.

8. **Hyperparameter Tuning**: The study highlights the critical role of hyperparameter tuning in the effectiveness of SPMF. It identifies the most important parameters to optimize, providing a framework for achieving the best possible model performance.

9. **Comprehensive Performance Evaluation**: Through extensive experimentation and evaluation using subsets of the MovieLens 100K and FilmTrust datasets, this research demonstrates improvements in recommendation robustness and interpretability over existing methods.

10. **Theoretical and Mathematical Properties of SPMF**: Unlike traditional matrix factorization techniques, which employ two factors with biconvex objective functions, SPMF utilizes a single factor with a convex objective function. This leads to a faster convergence rate compared to Alternative Least Squares (ALS) and Stochastic Gradient Descent (SGD) methods traditionally used for optimizing matrix factorization techniques. We prove that SPMF has a logarithmic convergence rate of $\mathcal{O}(\log(\frac{1}{\epsilon}))$ when the objective function is strongly convex with a Lipschitz continuous gradient. The latter

condition always holds true, but to demonstrate the conditions under which the objective functions is always stronlgy convex, we establish an upper bound: $\sum_{i=1}^{n+m} \lambda_i > \frac{n+m}{3}$, where m and n represent the number of users and items, respectively, and each λ_i denotes an eigenvalue of the matrix UU^T, that approximates the transformed blockwise matrix. Additionally, we demonstrate that incorporating a Gaussian prior $\mathcal{N}(0, \sigma^2)$ on the low-rank matrix U is equivalent to imposing a strong convexity constraint on the objective function. This constraint denoted by $\frac{\|U\|_F^2}{2\sigma^2}$, can be achieved within a Bayesian inference framework by leveraging the smoothness of the elements of U as a regularization term to satisfy the required conditions to achieve a convex objective function for efficient optimization. This theoretical analysis sheds light on the conditions that ensure an efficient convergence rate for our proposed SPMF model.

By introducing these innovations, in this PhD thesis, we made novel contributions to the field of recommendation systems. The introduction of SPMF not only overcomes the limitations of traditional matrix factorization techniques but also offers a robust framework for developing more interpretable recommendation systems. This work paves the way for future advancements in personalized recommendation technologies through the insights acquired from user-user, and item-item.

1.5 Methodology

This thesis employs the following methodology to address the research questions:

Literature Review (Chapter 2): A review of existing literature on matrix factorization techniques and recommendation systems will be conducted to identify the current state of research, existing methodologies, and gaps that this research aims to fill.

Model Development (Chapter 3): The SPMF model will be developed. This includes:

- Innovating a new paradigm for constructing a symmetric block matrix that integrates the user-item rating matrix with user-user and item-item matrices. This innovative approach blends the characterization of diverse user-user and item-item interactions within the recommendation system to facilitate personalized recommendations with interpretable low rank factorization.
- Designing a new algorithm for consolidating two distinct factors into a single unified low-rank factor, representing the holistic data matrix. This technique enhances the

interpretability of matrix factorization methods, providing a more comprehensive understanding of the underlying data structure.

- Establishing a new framework for binarizing a data matrix in the context of item recommendation systems. This method intuitively translates user preferences into binary ratings, representing likes and dislikes. By simplifying the complexity of preference data while maintaining the interpretability and usefulness of recommendations, this approach enhances the practicality and applicability of recommendation algorithms. It is particularly beneficial in scenarios where binary ratings are more relevant, such as video recommendations on platforms like YouTube or social media sites like Facebook and 𝕏 (formerly Twitter).

- Developing of a novel method, called SPBMF, which extends our suggested SPMF method by incorporating the Bernoulli distribution. This innovation is particularly suited for handling arbitrary binary matrices, enhancing the flexibility and applicability of probabilistic matrix factorization techniques in various binary data contexts.

- Developing and applying probabilistic methodologies to tackle data sparsity and estimate missing values. To achieve this, we explore the following approaches for estimating the low rank matrix: Maximum Likelihood Estimation (MLE) with a normal distribution, MLE with a Laplace distribution, and Bayesian estimation, which uses a normal distribution for the error term and a normal prior distribution for the entries of the low rank matrix.

- Conducting a rigorous analysis of the theoretical convergence rate of SPMF, providing insights into the efficiency and computational cost of this method in approximating symmetric matrices. This analysis will be compared against state-of-the-art techniques to highlight the significant improvements in convergence rate achieved by SPMF, demonstrating its superior performance and potential benefits in practical applications.

Experimental Design and Data Analysis (Chapter 4): A series of experiments will be designed to test and validate the SPMF model. Also, the data collected from the experiments will be analyzed to assess the performance of SPMF. These include:

- Selecting appropriate datasets (e.g., subsets of the MovieLens 100K and FilmTrust datasets).

- The critical hyperparameters of the SPMF model will be identified and optimized to achieve the best possible performance.

- Comparing the performance of SPMF with other matrix factorization techniques like SVD, NMF, and PMF based on RMSE.
- Evaluating the accuracy, efficiency, and robustness of SPMF.
- Analyzing the impact of different probabilistic distributions.
- Making predictions for the missing values (items that are not rated by users) using SPMF.
- Comparing user-user and item-item similarity obtained from SPMF with the original user-user and item-item similarity.

By following this structured methodology, in this PhD thesis we aim to develop a robust SPMF-based recommendation system that provides interpretable and personalized recommendations to addressing the identified research questions.

CHAPTER 2 OVERVIEW AND SYSTEMATIC LITERATURE REVIEW

RS are artificial intelligence algorithms that are associated with machine learning and use big data to recommend or suggest additional products to consumers. These suggestions can be based on several criteria, including search history, demographic information, past purchases, and other factors. RS are quite helpful since they direct consumers to things they might not have discovered on their own. They are trained to discover the preferences, previous decisions, and characteristics of users and products (items) using the data collected about their interactions. These include impressions, likes, clicks, ratings, and purchases. Moreover, they are highly favored by content creators, marketers, and product providers such as Netflix, Amazon, and Spotify due to their ability to predict user interests and preferences on a personalized level. For instance, Netflix uses RS to recommend movies and TV shows, Amazon suggests products based on past purchases, and Spotify curates playlists tailored to individual listening habits. These systems guide users to items that align with their interests, such as movies, books, clothing, music, and more, thereby enhancing user engagement and increasing sales.

In this chapter, we present definitions and fundamental objects of recommendation systems (RS), categorize RS approaches, and discuss machine learning techniques currently employed to address RS problems. Subsequently, we provide an overview of existing matrix factorization methods in recommendation systems, exploring various techniques such as SVD, NMF, and PMF, along with their respective mathematical models and benefits. Finally, we conduct a systematic literature review, encompassing research questions, hypotheses, keywords, and databases searched. This review synthesizes findings from multiple studies, highlighting key trends and gaps in matrix factorization for recommendation systems.

2.1 Preliminaries

RS are data processing systems that actively gather various types of information to generate personalized recommendations. Data is about the items to recommend and the users who will get these recommendations. Generally, there are recommendation techniques that are knowledge-poor, namely, those that use very simple and basic data, such as user ratings or evaluations of items. Other techniques are much more knowledge-dependent, in that they use ontological descriptions of the users or the items, constraints, or social relations and activities of the users [1]. As a general classification, the data used by RS refers to three types of objects: items, users, and transactions, that is, relations between the users and the items. In this section, we provide a general overview of RS objects, usages, and categories.

2.1.1 Recommendation Systems Objects

The data used by RS is divided into three types of objects: items, users, and transactions [1].

Items: In recommendation systems, "items" are the objects that are recommended to users. These items can be products, services, or any other type of content that a recommendation system is designed to recommend. The nature of the items can vary, and they can be distinguished by their value or utility.

For example, in the case of movie recommendations on Netflix, the items are movies that users might be interested in watching. The recommendation system collects data on the user's viewing history, preferences, and other factors to suggest movies that align with their interests. Similarly, in the case of music recommendations on YouTube, the items are music videos or songs that align with the user's interests. Amazon's recommendation system suggests computers and other electronics based on a user's past purchases, search history, and other relevant data.

RS items are the objects that are recommended to users by the recommendation system based on the user's preferences, past behavior, and other relevant data.

Users: In the context of RS, a user can refer to any person who is seeking recommendations for specific items. These users have different preferences, needs, goals, and characteristics that must be taken into account when creating personalised recommendations. The RS algorithm will collect information about the user's past behaviour, such as the items they have previously purchased or viewed, and will use this information to build a user profile. The profile can also include demographic information such as age, gender, and location, and psychographic data such as user interests, lifestyle, and personality.

The goal of creating a user profile is to improve the accuracy of recommendations and provide users with items that are relevant to their interests and preferences. For instance, if a user watches several romantic movies on Netflix, the RS algorithm will likely recommend other romantic movies to that user in the future. Similarly, if a user frequently purchases computer accessories on Amazon, the RS algorithm will likely recommend other related products such as computers or peripherals.

Transactions: A transaction refers to the recorded data between a user and the RS, which serves as an important element of information exchanged during the human-computer interaction and can be used to improve the recommendation algorithm. Transactions can take many forms, such as user ratings, textual reviews, or purchases made by the user.

One of the most common types of transaction data collected by RS is user ratings. Ratings can take on a variety of forms, including numerical ratings such as the 1–5 stars provided in the book recommendation associated with Amazon.com [1], ordinal ratings such as "strongly agree", "agree", "neutral", "disagree", "strongly disagree" where the user is asked to select the term that best indicates their opinion regarding an item, and binary ratings that model choices in which the user is simply asked to decide if a certain item is good or bad [1]. For the purpose of this thesis, we will focus on using numerical ratings to construct our proposed model.

Explicit feedback refers to the information that users provide in a deliberate and direct way, such as ratings or textual reviews, as opposed to implicit feedback which is gathered passively from user behavior. In contrast, implicit feedback gives a wide range of information about user behavior and preferences. This is implicit because we only know that the customer bought the items, but we cannot tell if they liked them or which one they preferred. Some other examples of implicit feedback are the number of clicks, the number of page visits, the number of times a song was played, etc.

In addition to ratings, textual reviews provided by users can also serve as a source of transaction data. These reviews can provide more detailed information about the user's preferences and opinions on specific items, which can be used to improve the accuracy of the recommendation algorithm.

In general, transaction data is an important source of information for RS, as it can provide valuable insights into user behavior and preferences. By analyzing transaction data, RS can personalize recommendations and provide more relevant suggestions to users, which can ultimately lead to a better user experience.

2.1.2 Why Recommendation Systems?

RS are used in various applications, including:

- Playlist generators for video and music services such as Netflix, YouTube, and Spotify. These systems help users discover new content that they may enjoy based on their viewing or listening history, ratings, and preferences. By using RS, these services provide a personalized experience for each user, increasing engagement and satisfaction.

- Product recommenders for services such as Amazon. RS analyze a user's purchase history, search queries, and ratings to provide personalized product recommendations. By doing so, these services improve customer experience and increase sales.

- Content recommenders for social media platforms such as Facebook and Twitter [9]. These systems analyze a user's activity, including likes, comments, and shares, to provide personalized content recommendations. By doing so, social media platforms increase engagement and retention.

RS are also used in other areas, such as job recommendations, news article recommendations, and dating services. The wide range of applications and the potential benefits of using RS have led to extensive research in this field. In the following sections, we will explore the key concepts and techniques used in RS.

There are multiple reasons to explain why service, product, and content providers exploit RS. The main reasons are to increase the number of items sold, sell more diverse items, and enhance user satisfaction. Below, we explain each reason.

Increase item sales

In the context of e-commerce, one of the main objectives of RS is to increase the number of items sold, which can be achieved by increasing the conversion rate. The conversion rate is the ratio of the number of users who make a purchase to the number of users who visit a website or use an application. A higher conversion rate indicates that more users are buying products, which results in increased revenue for the business.

One of the reasons why RS can improve the conversion rate is that they help users find items that match their interests or needs. This is especially important in situations where the user may not know exactly what they want or where there are many options to choose from. For example, imagine a user who wants to buy a pair of shoes but is not sure what style or color they prefer. By providing personalized recommendations based on the user's browsing or purchase history, RS can help the user discover products that they may not have found otherwise.

Sell more diverse items

Another important functionality of RS is to increase the sales of a business by recommending items that users may not have discovered otherwise. The goal is not only to recommend popular items but also to sell more diverse items that may not have been visible to the user without the recommendation. In other words, RS helps users discover and purchase a wider variety of items than they would have without the system's assistance.

From a commercial perspective, RS can help businesses increase sales diversity. RS can be particularly beneficial for businesses that have a large inventory of products that may be difficult to navigate or for businesses that are looking to promote new or less popular products. By offering personalized recommendations of such products, RS can ensure that most or all of the products in a business's catalogue are purchased.

Moreover, by recommending more diverse items, RS can help businesses to achieve a competitive advantage over other similar businesses by providing a more comprehensive range of products to customers. This can lead to higher sales and increased customer loyalty.

Enhance user satisfaction

Enhancing user satisfaction is another important goal of RS. With a well-designed RS, users are more likely to find the recommendations relevant and interesting, leading to a better experience with e-commerce platforms or applications. In today's competitive market, providing a personalized and relevant user experience is critical to the success of any business.

In addition, users are more likely to return to an e-commerce platform or application that provides accurate recommendations, leading to increased user loyalty and retention. A well-designed RS can also help to build trust between the user and the service provider, as users are more likely to trust a platform that provides accurate and relevant recommendations. Therefore, enhancing user satisfaction is a key factor in the success of an RS and is essential for building a loyal customer base and increasing sales.

2.1.3 User-Item Rating Matrix

The user-item rating matrix is the concept in recommendation systems that represents the recorded ratings data for a set of users and items. In general an RS deals with a fixed number of users and items which can be denoted as m and n, respectively. The users are denoted by $u_1, u_2, \ldots, u_m$, and the items are denoted by $i_1, i_2, \ldots, i_n$.

Moreover, the user-item rating matrix R is an $m \times n$ matrix where each row represents a user and each column represents an item. The element r_{jk} in the j-th row and k-th column represents the rating that user u_j gave to item i_k. In other words, r_{jk} is the interaction or transaction between user u_j and item i_k.

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & r_{jk} & \vdots \\ r_{m1} & \cdots & r_{mn} \end{pmatrix}. \tag{2.1}$$

For example, if we consider a movie recommendation system, then each row of the user-item rating matrix represents a user, and each column represents a movie. The element in the j-th row and k-th column of the matrix represents the rating that user u_j gave to movie i_k. These ratings can take various forms and can be integers, real values, positive or negative. The nature of the values of R is essential in determining the appropriate modeling techniques and algorithms to use in RS.

Rating values in R can be integers, with typical values ranging from 1 to 5 or 1 to 10, where higher values represent a better rating. The choice of integer ratings is common in rating systems where users are asked to rate items on a scale of predefined values. For example, in movie rating systems, users are often asked to rate movies on a scale of 1 to 5 or 1 to 10.

However, ratings in R can also be real values, where the ratings are not restricted to integer values. In some cases, users may provide more detailed feedback on an item, and the system may allow for fractional ratings. For example, a user may rate a movie 3.5 stars out of 5 [10]. Ratings can also be in the form of binary values such as like or dislike, interested or not interested, or simply 0 and 1.

In this study, the user-item rating matrix is assumed to consist of real positive ratings ranging from 0 to 5. This range is chosen to align with common rating systems and to allow for detailed feedback from users. In the event that the predicted ratings fall outside this range, a transformation will be employed to map the predicted ratings back into the desired range.

Normalization of the user-item rating matrix refers to the process of adjusting the ratings in the matrix to make them comparable for use in recommendation algorithms. The goal of normalization is to remove biases in the ratings due to factors such as different scales used by different users, varying popularity of different items, or different rating patterns among users [1]. Normalization was performed to ensure that the recommendation algorithms accurately compare and process the ratings.

One common normalization technique is mean-centering, where each user's average rating is subtracted from their ratings. This standardizes the data, mitigating the influence of individual user rating habits and item popularity. Research has shown that normalization techniques like mean-centering reduce rating bias and variance. For example, Koren et al. (2009) demonstrated that normalization enhances the performance of collaborative filtering algorithms by addressing biases in the input data [11].

There are several techniques for normalizing the user-item rating matrix, including:

- Subtracting the row mean: For each row, we subtract the mean rating of that row from each rating. This centers the ratings around zero and removes the row-wise biases in the ratings.
- Subtracting the global mean: We calculate the mean rating across all users and items and subtract this mean from each rating. This centers the ratings around zero and removes the global biases in the ratings.
- Scaling by the maximum value: We divide each rating by the maximum rating in the matrix. This scales the ratings to a common range of 0 to 1 and makes them directly comparable.

- Normalizing with L_2-norm: We normalize each row/column of the matrix by dividing each rating in the row/column by the Euclidean norm of the row/column. The Euclidean norm, also known as the L_2-norm, is the square root of the sum of the squares of the ratings in the row/column. For example, the Euclidean norm of a rating vector $\mathbf{r_i}$, where $\mathbf{r_i} = (r_{i1}, \ldots, r_{in})$ represents the ratings that user i made on the items 1 through n, is given by:
$$||\mathbf{r_i}||_2 = \sqrt{\sum_{j=1}^{n} r_{ij}^2} \tag{2.2}$$
This scales each row/column to have unit length and makes them directly comparable.

Let R denote the user-item rating matrix, as defined in (2.1). Prior to normalization, any missing values in R are replaced by 0. Subsequently, the row-normalized matrix R_{RN} for R can be calculated as follows:

$$R_{RN} = \begin{pmatrix} \frac{r_{11}}{\sqrt{r_{11}^2+\cdots+r_{1n}^2}} & \cdots & \frac{r_{1n}}{\sqrt{r_{11}^2+\cdots+r_{1n}^2}} \\ \vdots & \ddots & \vdots \\ \frac{r_{m1}}{\sqrt{r_{m1}^2+\cdots+r_{mn}^2}} & \cdots & \frac{r_{mn}}{\sqrt{r_{m1}^2+\cdots+r_{mn}^2}} \end{pmatrix} \tag{2.3}$$

Similarly, the column-normalized matrix R_{CN} for R can be calculated as follows:

$$R_{CN} = \begin{pmatrix} \frac{r_{11}}{\sqrt{r_{11}^2+\cdots+r_{m1}^2}} & \cdots & \frac{r_{1n}}{\sqrt{r_{1n}^2+\cdots+r_{mn}^2}} \\ \vdots & \ddots & \vdots \\ \frac{r_{m1}}{\sqrt{r_{11}^2+\cdots+r_{m1}^2}} & \cdots & \frac{r_{mn}}{\sqrt{r_{1n}^2+\cdots+r_{mn}^2}} \end{pmatrix} \tag{2.4}$$

In this thesis, we will normalize the user-item rating matrix R using the L_2 norm (either by row or column) before applying any RS technique. Normalizing the user-item rating matrix with the L_2 norm is a common technique in RS that has several benefits. By scaling each row/column of the matrix to have unit length, this normalization method makes the rows/columns directly comparable and reduces the impact of outliers or extreme values in the ratings [1]. Additionally, it helps to mitigate the effects of varying scales and biases in the ratings, while enhancing the interpretability and stability of the RS results.

Normalization with the L_2 norm provides two key justifications:

1. **Standardization of Scale**: After normalization, each row/column has a Euclidean norm of 1. This ensures that the ratings are on the same scale, regardless of the orig-

inal magnitude of the ratings. This standardization is crucial for algorithms that rely on distance or similarity measures, as it ensures that no single user's or item's ratings disproportionately influence the calculations due to differing scales. For example, consider two users who rate items differently within the same system: one user tends to rate movies between 1 and 5, using the full range of the scale, while another user rates mostly between 3 and 5, focusing on higher ratings. Without normalization, the second user's ratings could disproportionately influence the recommendation algorithm due to their generally higher values.

2. **Highlighting Relative Differences**: By scaling each row/column to unit length, normalization emphasizes the relative differences in the ratings rather than their absolute values. This helps the recommendation algorithm to focus on the pattern of ratings (e.g., preferences or dispreferences) rather than the magnitude of ratings, which can vary significantly between users.

One practical advantage of normalizing with the L_2 norm is that it allows us to obtain the similarity matrix for the normalized matrix by computing the dot product of the rows/columns, which is equivalent to multiplying the normalized matrix by its transpose. This property holds for similarity metrics based on cosine similarity, which are commonly used in RS [1]. We will elaborate more on the similarity matrix in the next section.

2.1.4 Similarity Matrix

In practice, we would compute the similarity matrix for all pairs of items in the user-item rating matrix and use it as input for recommendation algorithms such as collaborative filtering or matrix factorization. The similarity matrix captures the degree of similarity between each pair of items based on their ratings by users, and can help identify items that are likely to be recommended to users who have expressed interest in similar items. In the case of a user-item rating matrix, similarity between users or items is typically calculated using a similarity metric such as cosine similarity or Pearson correlation coefficient. These metrics measure the similarity between two vectors by computing the cosine of the angle between them or the correlation between their values. Cosine similarity between two items i and j

can be calculated as follows:

$$\text{Cosine Similarity}(i,j) = \frac{\sum_{u=1}^{m} r_{ui} r_{uj}}{\sqrt{\sum_{u=1}^{m} r_{ui}^2}\sqrt{\sum_{u=1}^{m} r_{uj}^2}}, \quad i,j = 1,...,n \tag{2.5}$$

where r_{ui} and r_{uj} are the ratings of user u for items i and j respectively.

Similarly, Pearson similarity between two items i and j can be calculated as follows:

$$\text{Pearson Similarity}(i,j) = \frac{\sum_{u=1}^{m} (r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u=1}^{m} (r_{ui} - \bar{r_i})^2}\sqrt{\sum_{u=1}^{m} (r_{uj} - \bar{r_j})^2}}, \quad i,j = 1,...,n \tag{2.6}$$

where $\bar{r_i}$ and $\bar{r_j}$ are the average ratings for items i and j, respectively. The average rating for item i, $\bar{r_i}$, is calculated as follows:

$$\bar{r_i} = \frac{1}{m} \sum_{u=1}^{m} r_{ui} \tag{2.7}$$

Similarly, the average rating for item j, $\bar{r_j}$, is calculated as follows:

$$\bar{r_j} = \frac{1}{m} \sum_{u=1}^{m} r_{uj} \tag{2.8}$$

The similarity between two users can be defined in a similar manner.

As mentioned in the previous section, one significant characteristic of the normalized user-item rating matrix R_{RN} calculated in (2.3) is that multiplying the normalized matrix by its transpose results in a matrix that represents the similarities between all pairs of users based on their item ratings, known as the user-similarity matrix S_{RN} of R_{RN}. To show this, we first compute $(R_{RN}R_{RN}^T)_{uv}$ for the users u and v as follows:

$$(R_{RN}R_{RN}^T)_{uv} = \frac{\sum_{j=1}^{n} r_{uj} r_{vj}}{\sqrt{\sum_{j=1}^{n} r_{uj}^2}\sqrt{\sum_{j=1}^{n} r_{vj}^2}}. \tag{2.9}$$

On the other hand, the user-similarity matrix $(S_{RN})_{uv}$ of R_{RN} for the users u and v, computed using the cosine similarity, is as follows:

$$(S_{RN})_{uv} = \frac{\sum_{j=1}^{n} r_{uj} r_{vj}}{\sqrt{\sum_{j=1}^{n} r_{uj}^2}\sqrt{\sum_{j=1}^{n} r_{vj}^2}}. \tag{2.10}$$

As we can see $(R_{RN}R_{RN}^T)_{uv} = (S_{RN})_{uv}$, meaning that multiplying the row-normalized user-item rating matrix by its transpose results in a matrix that represents the similarities between all pairs of users based on their item ratings. Similarly, if we normalize R columnwise, we obtain an item-item similarity matrix by multiplying the transpose of the column-normalized user-item rating matrix with the column-normalized user-item rating matrix.

2.1.5 Evaluating Recommendation Systems

The most commonly used evaluation measure for RS is Root Mean Squared Error (RMSE), which measures the difference between predicted and actual ratings for a set of users and items. To calculate RMSE, the data is split into a training set used for training the model and a test set used for evaluating the model's performance. The test set consists of a subset of user-item pairs that are held out from the training set. For each user-item pair (u, i) in the test set τ, where the actual rating r_{ui} is known, we generate the prediction $\hat{r}_{ui}$ and compare it to the actual rating r_{ui} using the following formula:

$$RMSE = \sqrt{\frac{1}{|\tau|} \sum_{(u,i)\in\tau} (\hat{r}_{ui} - r_{ui})^2} \ .$$

In this formula, RMSE measures the square root of the average squared differences between predicted ratings and actual ratings across all user-item pairs in the test set. A lower RMSE indicates that the model's predictions are closer to the actual ratings, signifying better recommendation accuracy. Therefore, RMSE is a critical tool in assessing and comparing the performance of various recommendation models.

The other evaluation metric is "classification accuracy metrics," which attempt to assess recommendation algorithms' ability to make successful decisions [12]. They count the number of correct and incorrect classifications as relevant (liked by user) or irrelevant (disliked by user) items made by the recommendation system and are thus useful for user tasks such as finding good items. Furthermore, because only the correct or incorrect classification is measured, these metrics disregard the exact rating or ranking of items [13]. This type of metric is especially appropriate for e-commerce applications that attempt to persuade users to make certain decisions, such as purchasing products or services [12].

2.2 Recommendation Systems Categories

Generally, recommendation lists are produced based on user preference, item features, past user-item interactions, and some other additional information, such as temporal data [14]. RS techniques are usually classified into three categories [15]: Collaborative Filtering, Content Based and Hybrid RS. The following paragraphs provide the definition of each category.

2.2.1 Collaborative Filtering Recommendation Systems

The most popular method for RS is the Collaborative Filtering (CF) approach [1]. This approach makes recommendations based on user-item historical interactions [14]. The system produces recommendations by using only the gathered information about rating profiles for different users or items. This approach is based on collecting and analyzing a large amount of information on a user's behavior, activities, or preferences and predicting what the user will like based on their similarity to other users. The assumption of CF is that people will like similar kinds of items as they have in the past. And also, it assumes that users with similar tastes for some items may also have similar preferences for other items. The main idea of CF is to use the behaviour history of other like-minded users to provide the current user with good recommendations.

The following are types of collaborative filtering [1]:

- Memory-Based: This method performs recommendations by directly accessing the database and using user rating information to calculate the similarity between users or items. This calculated likeness is then used to make recommendations. It is adaptable to data changes because it accesses the database directly, but it takes a long time to compute due to the size of the data.

- Model-Based: This method uses the transaction data to create a model that can generate recommendations. The models are created using data mining and machine learning techniques, such as matrix factorization, clustering, and classification algorithms. These algorithms identify patterns and relationships within the training data, which are then used to make predictions for new data. This method has a constant computing time regardless of the size of the data but is not adaptive to data changes.

- Hybrid: Various programs combine the model-based and memory-based CF algorithms to leverage the advantages of both approaches. This means using the memory-based approach's ability to adapt to new data and the model-based approach's ability to generate recommendations once the model is trained. By integrating both methods, hybrid systems can provide more accurate recommendations and address the limitations of each individual approach.

Below, we provide a detailed description of each approach.

Memory-Based CF

Memory-Based Collaborative filtering is a type of collaborative filtering that directly accesses the database of user ratings to make recommendations. It works by finding users who have rated items similarly to the target user and uses these ratings to calculate a similarity score between the users. Once the similarity score is calculated, it is used to identify the items that the target user has not yet rated but that the similar users have, and these items are recommended to the target user [1].

There are two main types of Memory-Based Collaborative Filtering: User-based and Item-based. In User-based Collaborative Filtering, the similarity is calculated between users based on the similarity of their rating patterns. For example, if two users have rated a particular set of items similarly, they are considered more similar than two users who have rated those same items very differently. Once the similarity is calculated, recommendations are made based on the items rated highly by similar users. While this approach involves calculating similarity scores, it can also be viewed through the lens of clustering, where users with similar rating patterns can be considered part of the same implicit cluster. However, it is important to note that traditional User-based Collaborative Filtering does not explicitly form clusters but rather relies on these similarity scores to make recommendations.

In Item-based Collaborative Filtering, the similarity is calculated between items instead of users. This method is particularly useful when dealing with large datasets because calculating similarities between items is generally more computationally efficient than calculating similarities between users. Once the similarity is calculated, recommendations are made based on items that are similar to the ones rated highly by the target user. While this approach involves calculating similarity scores between items, it can also be conceptually related to clustering, where items with similar rating patterns can be considered part of the same implicit cluster. However, it is important to note that traditional Item-based Collaborative Filtering does not explicitly form clusters but rather relies on these similarity scores to make recommendations.

Memory-Based Collaborative Filtering has several advantages. It adapts to data changes because it accesses the database directly and can handle new users or items. It is also straightforward to implement and interpret, making it a common choice for many recommendation systems. However, it also has some limitations. The size of the database can impact the time required to compute similarities, and it can be prone to overfitting if the dataset is small [1].

Model-Based CF

Model-based collaborative filtering (CF) is another type of recommendation system approach that uses a model to generate recommendations. In this method, the system creates a model using transaction data, which includes data on the items that users have rated or interacted with, such as clickstream data. The model then analyzes this data using data mining techniques to identify patterns or relationships between the users and items [1].

The goal of this method is to create models that can be used to make recommendations for new users or items that have not been seen before. These models can be based on a variety of techniques, including matrix factorization, Bayesian networks, decision trees, and clustering algorithms [1]. In some cases, hybrid models that combine multiple techniques are used to improve the accuracy and robustness of the recommendations.

One of the benefits of model-based CF is that it often has predictable and efficient computation times, which makes it a good choice for businesses with large datasets to analyze. Additionally, model-based techniques can handle sparsity in the user-item matrix by using a low-dimensional space representation of users and items to make predictions. This representation captures the underlying patterns and relationships in the user-item interaction data, allowing for accurate predictions for unseen user-item pairs. However, one limitation of this method is that it is not adaptive to data changes, meaning that the model needs to be retrained if new data is added or if user behavior changes significantly [1]. Overall, model-based CF is a tool for generating recommendations, especially for large datasets. However, it requires expertise in data mining and machine learning techniques to build an accurate and robust model, and it may not be as adaptable to changing data as other approaches, such as memory-based CF [1]. In this thesis, our focus will be on the model-based approach, and these models will be discussed in more detail in 2.3.

Hybrid CF

Hybrid collaborative filtering combines the strengths of both memory-based and model-based approaches to improve recommendation accuracy. The idea is to create a hybrid approach by combining the model-based and memory-based techniques, taking advantage of their strengths while minimizing their weaknesses. This technique has gained popularity in recent years due to its ability to tackle some of the issues faced by memory-based and model-based approaches [16].

In a hybrid CF approach, the system uses a combination of memory-based and model-based algorithms to create a more accurate recommendation system. For example, the system can use the memory-based approach to make recommendations based on the user's historical data and combine this with the model-based approach that uses a mathematical model to predict a user's future preferences based on similar users' historical data [16].

Hybrid CF is advantageous because it can handle sparse data, as it can use the strengths of memory-based and model-based CF algorithms to predict recommendations. This approach provides a more accurate recommendation as the model-based approach can handle the cold start problem, while the memory-based approach can handle changes in user preferences over time, which makes it difficult for the model-based approach to adapt.

Moreover, hybrid CF can handle scalability issues faced by memory-based CF, which requires large amounts of memory to store the user-item rating matrix, and model-based CF, which requires a lot of computation power to build models. By combining these two approaches, hybrid CF can provide better scalability and accuracy, making it a popular choice for recommendation systems in practice [16].

2.2.2 Content-Based Recommendation Systems

Content-Based Recommendation Systems use a different approach compared to collaborative filtering. Instead of relying on the input of other users to make recommendations, this method creates a user profile based on their preferences and interests. The profile is built by analyzing the items that the user likes and the keywords that describe these items. For example, if a user frequently watches action movies, the system identifies the keywords associated with these movies (such as "action," "thriller," and "adventure") and uses this information to recommend similar content [1]. In comparison to item-based collaborative filtering, which calculates the similarity between items based on user ratings, content-based filtering focuses on the characteristics of the items themselves. While item-based collaborative filtering might recommend items that similar users have liked, content-based filtering recommends items that are similar in content to what the user has liked in the past. This approach allows for personalized recommendations based on the user's individual preferences rather than the preferences of other users.

When there is not much user information available, but item data is easily accessible, content-based filtering performs well. The system uses the item's attributes, such as its name, description, and location, to create a user profile that indicates the type of items that the user is likely to prefer [1].

This approach focuses on two types of information to create a user profile. Firstly, it uses a model to determine the user's preferences. Secondly, it uses a history of the user's interaction with the RS, including the items they have liked or disliked, to refine the user profile. The system can then use this profile to make recommendations for items that are similar to those the user has already shown an interest in.

2.2.3 Hybrid Recommendation Systems

Hybrid Recommendation Systems are a combination of two or more recommendation techniques, typically content-based and collaborative filtering, to overcome their limitations and improve their performance. Hybrid Recommendation Systems have become increasingly popular due to their ability to provide more accurate and diverse recommendations by combining the strengths of different recommendation techniques.

The main idea behind a hybrid RS is to create a more powerful recommendation system by combining the content-based and collaborative filtering methods, which can complement each other's limitations. Content-based RS typically recommend items similar to those the user has previously interacted with, whereas collaborative filtering-based RS recommend items based on the user's preferences and similarities with other users.

There are several schemes to implement a hybrid RS [17]:

1. **Separate Predictions and Combination**: In this scheme, the content-based and collaborative filtering methods make their predictions separately, and the results are combined to produce the final recommendations. One way to combine the results is to give them equal weights, or to assign weights based on their predicted accuracy.

2. **Content-based and Collaborative Filtering Integration**: In this scheme, the content-based and collaborative filtering techniques are combined into one model, which provides recommendations based on both approaches. This is typically done by adding content-based capabilities to the collaborative filtering approach, or by incorporating collaborative filtering into the content-based approach.

   For example, one approach is to use the item attributes as additional input to the collaborative filtering algorithm, so that the model considers both the item attributes and the user-item interactions. Another approach is to use the collaborative filtering algorithm to generate initial recommendations and then filter them based on their similarity to the user's profile.

3. **Unified Approach**: In this scheme, the content-based and collaborative filtering methods are integrated into a single model that learns the user-item preferences from both the content and the user-item interactions. This approach can be challenging because it requires finding a way to combine the two types of data and requires significant data preprocessing and feature engineering.

   For example, one approach is to use a neural network that takes both the item features and the user-item interactions as input and generates recommendations based on both types of data. Another approach is to use matrix factorization to learn a joint representation of the users and items, which can capture both the content-based and collaborative filtering signals.

Hybrid RS have been shown to outperform individual recommendation methods in terms of accuracy. However, creating a hybrid RS can be challenging, and the performance gains may not always be significant [1]. The choice of the hybrid scheme depends on the availability of data and the specific application requirements. In practice, a hybrid RS often requires significant experimentation to find the optimal combination of techniques and parameters [17].

Cold-start is one of the more well-known issues in recommendation systems. Cold-start refers to the issue that the system cannot extract any inferences for new users or items. In situations where data is sparse or when dealing with the challenges of cold start (e.g., limited data for new users or items), hybrid recommendation systems can be a suitable approach [17]. A good example of this method is Netflix. When the website makes recommendations based on comparing the watching and searching habits of similar users, it uses the CF approach, and when recommending movies that share similar characteristics that a user has rated highly before, it uses the content-based filtering method [1].

2.3 Matrix Factorization Overview in Recommendation Systems

As mentioned in the previous section, memory-based models are methods for developing recommendation systems. However, the accuracy of rating predictions can be negatively impacted by the sparsity of the user-item rating matrix. Additionally, measuring the similarity between users can be computationally challenging, with complexity increasing as the data volume grows [1]. Specifically, the number of pairwise comparisons required to calculate similarity grows exponentially with the number of users and items in the system, leading to significant computational overhead. This exponential growth makes it difficult to scale the system to manage large datasets.

In this Section, we review the popular matrix factorization algorithms: Singular Value Decomposition (SVD), Basic Matrix Factorization, Non-negative Matrix Factorization (NMF), and Probabilistic Matrix Factorization (PMF) used in RS.

2.3.1 Singular Value Decomposition

The SVD is a linear algebra method that can be used as a dimensionality reduction technique in machine learning. In the context of RS, the SVD method is applied as a collaborative filtering technique to find latent factors of the user-item rating matrix R and predict user-item interactions. In this thesis, the two terms "factors" and "latent factors" are treated as synonyms.

Given an $m \times n$ matrix R with rank r, the Singular Value Decomposition (SVD) of R, denoted as SVD(R), is defined as [18] :

$$\text{SVD}(R) = U\Sigma V^T \tag{2.11}$$

where:

- U is an $m \times m$ orthogonal matrix; the first r columns of U are the eigenvectors of RR^T (left singular vectors of R);
- V is an $n \times n$ orthogonal matrix; the first r columns of V are the eigenvectors of $R^T R$ (right singular vectors of R);
- Σ is an $m \times n$ diagonal matrix with r nonzero entries, such that $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, where $\sigma_i \geq 0$ for $1 \leq i \leq r$, $\sigma_i \geq \sigma_{i+1}$, and $\sigma_j = 0$ for $j > r$;
- $\{\sigma_1, \sigma_2, \dots, \sigma_r\}$ are the nonnegative square roots of the eigenvalues of $R^T R$ and are referred to as the singular values of R.

For a given number $k \leq r$, known as the number of latent factors, the matrix $R_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ minimizes the Frobenius norm $\|R - R_k\|_F$ across all rank-k matrices:

$$\min_{k \leq r} \|R - R_k\|_F \tag{2.12}$$

Thus, by retaining only the first k singular values of Σ and the corresponding columns and rows of U and V, the matrix R can be approximated as R_k:

$$R \approx R_k = U_k \Sigma_k V_k^T \tag{2.13}$$

where U_k consists of the first k columns of U and V_k^T consists of the first k rows of V. The matrix R_k obtained through this method is an approximation rather than an exact factorization of R. By selecting only the k largest singular values, this technique captures the structure of R while simultaneously eliminating noise [18]. Menon and Elkan [19] provide a comparative analysis of different methods for approximating the decomposition in the context of large matrices.

SVD is a utilized matrix decomposition technique, but it possesses certain limitations that have spurred the investigation of alternative approaches. Computational complexity stands as a prominent constraint of SVD, particularly when dealing with large matrices. The algorithm's time and memory demands escalate with matrix size, rendering it impractical for extensive datasets. SVD's efficacy in handling missing values is also limited, as even a small number of missing entries can substantially compromise the accuracy of the decomposition. To mitigate this, imputation techniques must be employed prior to applying SVD [1].

To overcome these challenges, researchers have developed various matrix factorization methods that can break the original matrix down into smaller, more manageable components. This approach allows for incremental updates and makes matrix factorization more scalable than SVD. Matrix factorization techniques can be tailored to specific applications and incorporate domain-specific information, such as user or item characteristics, to improve the factorization's quality. This adaptability makes matrix factorization a more versatile and responsive method than SVD [1].

In the following section, we will demonstrate how a user-item rating matrix in a movie recommendation scenario can be factorized into the product of three matrices: U, Σ, and V^T. We will also illustrate how the resulting factors can be interpreted in terms of user preferences and movie characteristics. Additionally, we will show how to predict ratings for users who have not rated certain movies.

Singular Value Decomposition : A Toy Example

Below, we demonstrate how to factorize a user-movie rating matrix using SVD. Consider four users: Alice, Bob, Carol, and Dave, who have rated three movies: "Matrix" (Action), "Titanic" (Romance), and "Inception" (Action). The user-movie rating matrix R is shown below, where each entry represents the rating given by a user to a movie on a scale of 1 to 5:

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \left(\begin{array}{ccc} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{array}\right) \end{array}. \tag{2.14}$$

In this matrix, Alice, Bob, Carol, and Dave are the users, and "Matrix," "Titanic," and "Inception" are the items (movies). For example, Alice rated "Matrix" a 4, "Titanic" a 1, and did not rate "Inception." Bob rated "Matrix" a 2, "Titanic" a 5, and "Inception" a 2, and so on.

"Action" and "Romance" are identified as two latent factors because they represent the underlying patterns or dimensions in the data that capture the most significant sources of variation in user preferences and item characteristics. In the context of movie ratings, these factors can be interpreted as the primary themes or genres that influence user ratings. The "Action" factor is associated with movies like "Matrix" and "Inception," which are action-oriented, while the "Romance" factor is associated with "Titanic," a romance movie. These latent factors help in understanding how different users' preferences align with different movie genres. In real-world problems, determining the exact number of latent factors is challenging and requires additional contextual knowledge and insight.

To factorize this matrix using SVD with $k = 2$, we use the 'TruncatedSVD' function from the 'sklearn.decomposition' module in Python. Also, SVD assumes all missing entries are 0. The SVD decomposition yields three matrices: U, Σ, and V^T. Below, we interpret each matrix separately.

Matrix U (User-Feature Matrix):

Matrix U represents the relationship between users and latent factors. Each row corresponds to a user, and each column in U corresponds to a latent factor. Higher values in each row indicate a stronger association of the user with the corresponding latent factor.

$$U = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} -0.22 & 0.59 \\ -0.49 & -0.17 \\ -0.62 & 0.50 \\ -0.58 & -0.61 \end{pmatrix} \end{array}.$$

However, the presence of negative values within the matrix complicates its interpretation. For instance, consider the case of Dave, who rated both "Romance" and "Action" movies as 5. Despite this, the matrix reveals associations of -0.58 and -0.61 for these genres, respectively. This discrepancy introduces challenges in accurately discerning his true preferences.

Matrix Σ (Diagonal Matrix of Singular Values):

The singular values (10.89 and 5.16) represent the importance or strength of the corresponding latent features. The first latent feature (10.89), related to "Action", is more significant than the second (5.16), related to "Romance".

$$\Sigma = \begin{array}{c} \\ \\ \end{array}\begin{array}{cc} \text{Action} & \text{Romance} \\ \end{array}$$
$$\Sigma = \left(\begin{array}{cc} 10.89 & 0 \\ 0 & 5.16 \end{array}\right).$$

This suggests that the action feature captures more of the underlying structure in the user-movie interactions. The first and second latent factors represent underlying patterns or dimensions in the data that capture the most significant sources of variation in user preferences and movie characteristics.

Matrix V (Item-Feature Matrix):

Matrix V characterizes the relationship between movies and latent factors. Each row corresponds to a movie, and each column in V corresponds to a latent factor. Higher values in each column indicate a stronger association of the movie with the corresponding latent factor. The presence of both negative and positive values in V makes interpretation more challenging.

$$V = \begin{array}{c|cc} & \text{Action} & \text{Romance} \\ \text{Matrix} & -0.45 & 0.87 \\ \text{Titanic} & -0.62 & -0.45 \\ \text{Inception} & -0.64 & -0.18 \end{array}.$$

For example, for the movie "Inception," which is an action movie, the value of the latent factor "Action" (-0.64) is not interpretable.

Predicting Missing Values:

As we can see from the rating matrix R, Alice did not rate "Inception" and Dave did not rate "Matrix." By using these three obtained matrices, we can predict the missing ratings in the matrix R. This is done by multiplying U, Σ, and V^T to reconstruct the matrix $\hat{R}$:

$$\hat{R} = U\Sigma V^T.$$

This reconstructed matrix $\hat{R}$ will provide estimates for the missing values, based on the relationships captured by the latent factors.

$$\hat{R} = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 3.76 & 0.14 & \mathbf{1.01} \\ 1.64 & 3.69 & 3.53 \\ 5.28 & 3.01 & 3.81 \\ \mathbf{0.10} & 5.36 & 4.58 \end{pmatrix} \end{array}.$$

Because Alice is interested in "Matrix," which is an action movie, we would expect SVD to predict a high rating for "Inception" for Alice. However, the predicted rating is only 1.01. Similarly, since Dave is interested in "Inception," which is also an action movie, we would expect SVD to predict a high rating for "Matrix" for Dave. Instead, the predicted rating is just 0.10. This discrepancy highlights the challenges and limitations of interpreting SVD predictions.

In this example, we factorized a user-movie rating matrix using SVD. The matrix includes ratings from four users (Alice, Bob, Carol, and Dave) for three movies ("Matrix," "Titanic," and "Inception"). The decomposition produced three matrices: U, Σ, and V^T, which revealed underlying patterns in user preferences related to two latent features, Action and Romance. Despite these insights, the presence of both positive and negative values complicates interpretation. Predictions for missing ratings, such as Alice's rating for "Inception" and Dave's rating for "Matrix," demonstrated the limitations of SVD. The reconstructed matrix showed lower-than-expected values, highlighting challenges in accurately capturing user preferences solely based on latent features.

In this thesis, we aim to introduce a method called Symmetric Probabilistic Matrix Factorization (SPMF) that addresses this gap.

2.3.2 Basic Matrix Factorization

Latent factor models are a framework for modeling user-item affinity in collaborative filtering. The fundamental assumption is that this affinity can be captured by introducing latent factors that represent low-dimensional representations of users and items. These latent factors are unknown variables that characterize the underlying characteristics or attributes of users and items. By incorporating these latent factors, the model aims to capture the inherent structure and patterns in the user-item interactions [20].

In the context of recommendation systems, matrix factorization is a common form of latent factor model. It decomposes the user-item interaction matrix into two lower-dimensional matrices: one for users and one for items. The goal of these models is to learn these latent factors from observed data and use them to make predictions, such as recommending products or items to users.

In general, let R be a user-item rating matrix with dimensions $m \times n$, as defined in equation (2.1), and let k be the number of latent factors, which is initially unknown. The optimal number of latent factors, k, is typically determined through experimentation and cross-validation. This process involves training the model with various values of k and evaluating its performance on a validation set. The value of k that provides the best trade-off between model complexity and predictive accuracy is selected as the optimal number of latent factors. Our goal is to find two matrices U (an $m \times k$ matrix) and V (a $k \times n$ matrix) such that their product approximates R:

$$R \approx \hat{R} = UV.$$

The concept of matrix factorization can be expressed in detail as follows:

$$\hat{r}_{pq} = u_p v_q = \sum_{j=1}^{k} u_{pj} v_{jq}, \tag{2.15}$$

where u_p is a row of matrix U and v_q is a column of matrix V. To derive the factor vectors u_p and v_q, the RS minimizes the sum of squared errors on the set of given ratings using the following optimization objective:

$$\min_{u_p, v_q} \sum_{(p,q) \in \kappa} (r_{pq} - u_p v_q)^2 + \lambda(||u_p||^2 + ||v_q||^2), \tag{2.16}$$

where κ is the set of the known ratings r_{pq} for which user p made a rating for item q and $\lambda \geq 0$ is the regularization term which can be estimated by cross-validation. There are two approaches to minimizing (2.16): stochastic gradient descent and alternating least squares (ALS) [11]. Below, we describe each approach briefly.

Stochastic Gradient Descent

In stochastic gradient descent, we need to find a way to obtain matrices U and V. One approach to solve this problem is to first initialize the two matrices with some values and then calculate the sum of squared errors between the estimated rating and the real rating, as follows [21]:

$$\sum_{(p,q)\in\kappa} e_{pq}^2 = \sum_{(p,q)\in\kappa} (r_{pq} - \hat{r}_{pq})^2 = \sum_{(p,q)\in\kappa} (r_{pq} - \sum_{j=1}^{k} u_{pj}v_{jq})^2. \tag{2.17}$$

Then, in an iterative process, the goal is to minimize the sum of squared errors using gradient descent to find a local minimum. To minimize the sum of the squared error, we need to know the gradient at the current values and, therefore, differentiate the above equation with respect to these two variables, which are v_{jq} and u_{pj}, separately [21]:

$$\begin{aligned} \frac{\partial}{\partial v_{jq}} e_{pq}^2 &= -2(r_{pq} - \hat{r}_{pq})u_{pj} = -2e_{pq}u_{pj}, \\ \frac{\partial}{\partial u_{pj}} e_{pq}^2 &= -2(r_{pq} - \hat{r}_{pq})v_{jq} = -2e_{pq}v_{jq}. \end{aligned} \tag{2.18}$$

Having obtained the gradient, we can now formulate the update rules for both u_{pj} and v_{jq}:

$$\begin{aligned} v'_{jq} &= v_{jq} - \alpha\frac{\partial}{\partial v_{jq}} e_{pq}^2 = v_{jq} + 2\alpha e_{pq}u_{pj}, \\ u'_{pj} &= u_{pj} - \alpha\frac{\partial}{\partial u_{pj}} e_{pq}^2 = u_{pj} + 2\alpha e_{pq}v_{jq}. \end{aligned} \tag{2.19}$$

where α is a constant whose value determines the rate of approaching the minimum [21]. The update rules are derived from the gradient descent optimization algorithm. Specifically, the rule for updating v_{jq} is derived as follows:

First, we start with the gradient with respect to v_{jq},

$$\frac{\partial}{\partial v_{jq}} e_{pq}^2 = -2e_{pq}u_{pj},$$

the gradient descent update rule is:

$$v'_{jq} = v_{jq} - \alpha\frac{\partial}{\partial v_{jq}} e_{pq}^2 = v_{jq} + \alpha 2e_{pq}u_{pj},$$

where α is the learning rate that controls the size of the update step. Similarly, the rule for updating u_{pj} is:

$$u'_{pj} = u_{pj} - \alpha \frac{\partial}{\partial u_{pj}} e^2_{pq} = u_{pj} + \alpha 2 e_{pq} v_{jq}.$$

In Algorithm 1 we describe in more detail the iteration process.

Algorithm 1 Stochastic Gradient Descent for the Basic MF in RS

```
Require: $U$ and $V$ (initial matrices), $\kappa$ (set of user-item pairs)
Ensure: Optimized matrices $U$ and $V$
 1: Initialize matrices $U$ and $V$ with random values
 2: Set the learning rate $\alpha$
 3: Set the maximum number of iterations
 4: for ITER $\leftarrow 1$ to maximum iterations do
 5:     Calculate the sum of squared differences: $e^2_{pq} = \sum_{(p,q)\in\kappa}(r_{pq} - \sum_{j=1}^{k} u_{pj}v_{jq})^2$
 6:     for $(p,q) \in \kappa$ do
 7:         Compute the gradients:
 8:             $\frac{\partial}{\partial v_{jq}} e^2_{pq} = -2e_{pq}u_{pj}$
 9:             $\frac{\partial}{\partial u_{pj}} e^2_{pq} = -2e_{pq}v_{jq}$
10:         Update the values of $v_{jq}$ and $u_{pj}$:
11:             $v'_{jq} = v_{jq} + 2\alpha e_{pq}u_{pj}$
12:             $u'_{pj} = u_{pj} + 2\alpha e_{pq}v_{jq}$
13:     end for
14:     Update matrices $U$ and $V$ with the new values: $U \leftarrow U'$, $V \leftarrow V'$
15: end for
16: return Optimized matrices $U$ and $V$
```

Alternating Least Squares (ALS)

In the equation (2.16), both u_p and v_q are unknowns, and hence this equation is not convex. Nevertheless, if one fixes one of the unknowns, the optimization problem turns out to be a quadratic problem and can be optimally solved. Therefore, ALS techniques revolve between fixing the u_p's and the v_q's. By fixing all u_p's, the system calculates the v_q's by solving a least-squares problem and vice versa. This causes each step to decrease (2.16) until convergence [21].

One advantage of ALS versus stochastic gradient descent is its parallelization power. Meaning that, in ALS the system calculates each v_q independently of the other item factors and calculates each u_p independently of the other user factor which generates possibly massive parallelization of the algorithm [11]. The term "massively parallel" refers to the use of a huge number of computer processors (or independent computers) to execute a series of coordinated computations in parallel at the same time [22]. The second advantage is ALS can efficiently handle the sparsity of user-item rating matrix [11].

Basic matrix factorization is a technique used to predict user-item ratings based on past user-item interactions [21]. However, this technique has some limitations that can affect its effectiveness in certain applications. One of the main limitations of basic matrix factorization is that it does not enforce non-negativity constraints on the latent factors. This can make the factors difficult to interpret, especially in applications such as RS where the factors represent user preferences or item features. For example, a negative factor value for a user preference or item feature does not have a clear interpretation.

Non-negative matrix factorization (NMF) is a variant of matrix factorization that imposes non-negativity constraints on the factor matrices. This constraint ensures that the latent factors are non-negative, which makes them more interpretable in some RS applications. In particular, NMF can be used to factorize the user-item rating matrix into non-negative user and item factors, which can be interpreted as user preferences and item features, respectively. In the following section we will present this approach in more detail.

In the following section, we will demonstrate how a user-movie rating matrix can be factorized into the product of two matrices: U and V using basic matrix factorization technique. We will also illustrate how the resulting factors can be interpreted in terms of user preferences and movie characteristics. Additionally, we will show how to predict ratings for users who have not rated certain movies, thereby enhancing the recommendation system's accuracy.

Basic Matrix Factorization: A Toy Example

We will factorize the same matrix we used for SVD, but this time using Basic Matrix Factorization. Below, we demonstrate how to factorize a user-movie rating matrix using this technique. To remind the reader, consider four users: Alice, Bob, Carol, and Dave, who have rated three movies: "Matrix" (Action), "Titanic" (Romance), and "Inception" (Action). The user-movie rating matrix R is shown below, where each entry represents the rating given by a user to a movie on a scale of 1 to 5:

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{pmatrix} \end{array}$$

To factorize this matrix using Basic Matrix Factorization, we implement the algorithm 1 in Python. Also, because there are two latent factors, "Action" and "Romance," in this toy example, we set $k = 2$. The Basic Matrix Factorization yields two matrices: U and V. Below, we interpret each matrix separately.

Matrix U (User-Feature Matrix):

As mentioned before, matrix U represents the relationship between users and latent factors. Each row corresponds to a user, and each column in U corresponds to a latent factors. Higher values in each row indicate a stronger association of the user with the corresponding latent factors.

$$U = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} -0.16 & 1.56 \\ -1.84 & 0.62 \\ -0.49 & 1.92 \\ -1.67 & 1.81 \end{pmatrix} \end{array}.$$

However, the presence of negative values in the matrix makes interpretation challenging. For example, the association between Carol and the latent factor "Action" is -0.49 which is not interpretable.

Matrix V (Item-Feature Matrix):

Matrix V characterizes the relationship between movies and latent factors. Each row corresponds to a latent factor, and each column corresponds to a movie. Higher values in each row indicate a stronger association of the movie with the corresponding latent factor. The presence of both negative and positive values in V adds complexity to the interpretation.

$$V = \begin{array}{c} \\ \text{Action} \\ \text{Romance} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \left(\begin{array}{ccc} -0.22 & -2.57 & -0.22 \\ 2.54 & 0.37 & 2.54 \end{array} \right) \end{array}.$$

For example, the value -0.22 for the latent factor "Action" in the case of "Inception" is not interpretable. Additionally, although "Matrix" is primarily an action movie, the value for the latent factor "Romance" (2.54) is higher than that for the latent factor "Action" (-0.22).

Predicting Missing Values:

From the rating matrix R, we can see that Alice did not rate "Inception," and Dave did not rate "Matrix." By using the two obtained matrices U and V, we can predict the missing ratings in matrix R. This is done by multiplying U and V to reconstruct the matrix $\hat{R}$:

$$\hat{R} = UV.$$

This reconstructed matrix $\hat{R}$ provides estimates for the missing values based on the relationships captured by the latent factors.

$$\hat{R} = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \left(\begin{array}{ccc} 3.99 & 0.99 & \mathbf{3.98} \\ 1.97 & 4.95 & 1.98 \\ 4.98 & 1.96 & 4.98 \\ \mathbf{4.96} & 4.96 & 4.96 \end{array} \right) \end{array}.$$

For example, since Alice is interested in "Matrix," which is an action movie, we expect the Basic Matrix Factorization to predict a high rating for "Inception" for Alice, resulting in a prediction of 3.98, which is acceptable. Similarly, since Dave is interested in "Inception," which is also an action movie, we expect the Basic Matrix Factorization to predict a high rating for "Matrix" for Dave, resulting in a prediction of 4.96, which meets our expectations.

The presence of negative values in the factor matrices makes it difficult to interpret the latent factors in a meaningful way. This highlights the limitations of basic matrix factorization and suggests the potential advantage of using Non-Negative Matrix Factorization (NMF), which enforces non-negativity constraints, leading to more interpretable and meaningful latent factors.

2.3.3 Nonnegative Matrix Factorization

In Section 2.3.2, we presented the concept of Latent Factor Models as a means to reduce the dimensionality of the user-item rating matrix. These models enable us to represent ratings using a limited number of latent factors, depicting users and items in a lower-dimensional space. One prevalent technique for uncovering these latent factors and representing users and items involves low-rank matrix factorization of the rating matrix.

Low-rank matrix factorization refers to the process of approximating the original high-dimensional user-item rating matrix with the product of two lower-dimensional matrices. This approximation relies on the assumption that the user-item interactions can be captured by a smaller number of underlying factors, thus reducing the rank of the original matrix. By doing so, we represent the interactions in the data with fewer parameters, which improve generalization and computational efficiency. The resulting low-rank matrices contain the latent factors that summarize the preferences of users and the characteristics of items, which then be used for tasks such as recommendation.

Nonnegative Matrix Factorization (NMF) is a specific technique used for low-rank matrix factorization. Its objective is to decompose the rating matrix into non-negative matrices with a reduced rank k. In this context, the rank k corresponds to the number of latent factors we aim to discover during the factorization process. NMF proves to be particularly valuable when working with data featuring non-negative entries. This technique enables the representation of users and items within a non-negative factor space.

Below, we describe how NMF is formulated from a mathematical standpoint. Let $R \in \mathbb{R}^{m\times n}$ be a non-negative matrix (i.e., $m_{ij} \geq 0$), and let $k < \min\{m, n\}$ be an integer. In the NMF technique, the objective is to find two matrices, $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{k\times n}$, with non-negative entries such that UV is the closest matrix to R with respect to an appropriate norm. The commonly used choice is the Frobenius norm, and the task can be formulated as an optimization problem, seeking to find U and V by solving the non-convex optimization problem [23]:

$$\min_{U\geq 0, V\geq 0} G(U, V), \tag{2.20}$$

where $G(U, V) = \frac{1}{2}||R - UV||_F^2$, and $U \geq 0$ and $V \geq 0$ mean that all elements of U and V are nonnegative. The optimization problem (2.20) is a non-convex problem, which means that it is expected to find only a local minimum [23]. The minimization problem can be solved using the Alternating Nonnegative Least Square (ANLS) framework. In the given equation, F represents the Frobenius norm, which is a way to measure the magnitude of a matrix. In the context of matrix factorization, the Frobenius norm is often used as a measure of how well the original matrix can be approximated by its factorization. The goal of the optimization problem is to find the values of U and V that minimize the Frobenius norm of the difference between the original matrix R and its reconstruction using the factors U and V.

The NMF algorithm starts by initializing U and V with non-negative elements. The choice of k is crucial in NMF because it determines the rank of the low-dimensional approximation of the matrix R. The optimal value of k is typically determined through experimentation and cross-validation. This involves training the NMF model with various values of k and evaluating its performance on a validation set. The value of k that provides the best trade-off between model complexity and predictive accuracy is selected as the optimal number of latent factors. After the initialization of U with non-negative elements, the ANLS algorithm iteratively updates U and V to minimize the objective function $G(U, V)$ as defined in (2.20). In each iteration, the algorithm first fixes V and solves for U, and then fixes U and solves for V. The ANLS algorithm continues to alternate between updating U and V until certain convergence criteria are achieved, such as a maximum number of iterations or a small change in the objective function value.

To better understand how NMF works, we will explore a toy example in the following section. We will demonstrate the factorization of a user-movie rating matrix into two matrices, U and V, using the NMF technique. This example will also illustrate how the resulting factors can be interpreted in terms of user preferences and movie characteristics. Furthermore, we will show how to predict ratings for users who have not rated certain movies, thereby improving the accuracy of the recommendation system.

Non-negative Matrix Factorization: A Toy Example

We will factorize the same matrix used for SVD and Basic Matrix Factorization, but this time employing NMF. We will demonstrate how to decompose a user-movie rating matrix using this technique. The user-movie rating matrix R is shown below, with each entry representing the rating given by a user to a movie on a scale from 1 to 5:

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{pmatrix} \end{array}.$$

To factorize this matrix using NMF, we implement an algorithm in Python utilizing the loss function (2.20). Given that there are two latent factors, "Action" and "Romance," in this toy example, we set $k = 2$. The NMF decomposition results in two matrices: U and V. Below we will interpret each matrix separately.

Matrix U (User-Feature Matrix):

As mentioned before, the matrix U illustrates the connection between users and latent factors. Each row in U corresponds to a specific user, while each column represents a latent factor. Higher values within a row signify a stronger association between the user and the respective latent factor.

$$U = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} 0.96 & 0 \\ 0.42 & 1.29 \\ 1.35 & 1 \\ 0.03 & 1.9 \end{pmatrix} \end{array}.$$

Below, we interpret the matrix U in detail:

- **Alice:** (0.96, 0) - Predominantly associated with "Action".
- **Bob:** (0.42, 1.29) - Associated with both "Action" and "Romance", with a higher association with "Romance".
- **Carol:** (1.35, 1) - Highly associated with both "Action" and "Romance". This association is not accurate because Carol rated the action movies (Matrix and Inception) highly and the Romance movie (Titanic) poorly.
- **Dave:** (0.03, 1.9) - Barely associated with "Action" and highly associated with "Romance". This association is not accurate because Dave rated the Romance movie and action movie equally.

Matrix V (Item-Feature Matrix)

Matrix V represents the relationship between movies and latent factors. Each row corresponds to a specific latent factor, and each column corresponds to a movie. Higher values within each row signify a greater association between the movie and the respective latent factor.

$$V = \begin{array}{c|ccc} & \text{Matrix} & \text{Titanic} & \text{Inception} \\ \hline \text{Action} & 3.9 & 0.15 & 1.05 \\ \text{Romance} & 0 & 2.82 & 2.4 \end{array}.$$

Below, we interpret the matrix V in detail:

- **Matrix (Action movie):** (3.9, 0) - Predominantly associated with "Action".
- **Titanic (Romance movie):** (0.15, 2.82) - Predominantly associated with "Romance".
- **Inception (Action movie):** (1.05, 2.4) - Associated with both "Action" and "Romance", with a higher association with "Romance". This association is not accurate because Inception is primarily known as an action movie and less so as a Romance movie.

Predicting Missing Values:

From the rating matrix R, we can see that Alice did not rate "Inception," and Dave did not rate "Matrix." By using the two obtained matrices U and V and obtain $\hat{R} = UV$, we can predict the missing ratings in matrix R.

$$\hat{R} = \begin{array}{c|ccc} & \text{Matrix} & \text{Titanic} & \text{Inception} \\ \hline \text{Alice} & 3.76 & 0.14 & \mathbf{1.01} \\ \text{Bob} & 1.64 & 3.69 & 3.53 \\ \text{Carol} & 5.28 & 3.01 & 3.81 \\ \text{Dave} & \mathbf{0.1} & 5.36 & 4.58 \end{array}.$$

The predicted rating for "Inception" (Action) is 1.01, which is unexpectedly low. Given that Alice rated "Matrix" highly, we would anticipate a similarly high rating for "Inception" as both are action movies. The predicted rating for "Matrix" (Action) is 0.10, which is extremely low. Considering that Dave rated "Inception" highly as an action movie, we would expect a high rating for "Matrix" as well.

2.3.4 Probabilistic Matrix Factorization

As mentioned in Section 2.2.1, the collaborative filtering (CF) approach can be used when there is insufficient data provided by a user to make item recommendations. In this situation, data provided by other users with similar tastes can be leveraged. Probabilistic Matrix Factorization (PMF) is a method used in CF to exploit these recommendations [8]. Below, we describe this methodology in more detail.

One significant issue with the user-item rating matrix R is its sparsity. This means that only some of its entries have ratings, while many remain empty. For a given user u, the system should be able to recommend items based on his or her preferences as well as the choices made by similar users and similar items. It is not necessary for user u to explicitly rate a particular item for it to be recommended. Instead, other users who have similar preferences can be used to infer the missing information about user u. Therefore, PMF falls into the CF category, as it addresses this problem by utilizing ratings provided by similar users.

In PMF, the user-item rating matrix $R \in \mathbb{R}^{m \times n}$ with m users and n items can be approximated by two low-rank matrices U and V as follows:

$$R \approx UV, \tag{2.21}$$

where U is an $m \times k$ matrix and V is a $k \times n$ matrix, with k denoting the number of latent factors. Salakhutdinov et al. [8] trained such a model using Bayesian inference to find the best rank-k approximation to the observed $m \times n$ user-item rating matrix R under a given loss function. Because U and V are low-rank matrices, PMF is also known as a low-rank matrix factorization problem.

The probabilistic model for PMF can be described by the following equation:

$$P(U, V|R, \sigma^2) \propto P(R|U, V, \sigma^2)P(U, V|\sigma_U^2, \sigma_V^2), \tag{2.22}$$

where σ^2 is the variance (diagonal covariance matrix) of a zero-mean spherical Gaussian distribution [8].

Since matrix U is related to users and matrix V is related to items, and given that users and items are assumed to be independent of each other, these two matrices are also independent of each other. Therefore, equation (2.22) can be rewritten as:

$$P(U, V|R, \sigma^2) \propto P(R|U, V, \sigma^2)P(U|\sigma_U^2)P(V|\sigma_V^2). \tag{2.23}$$

Salakhutdinov et al. [8] determined each component of this equation ($P(R|U, V, \sigma^2)$, $P(U|\sigma_U^2)$, and $P(V|\sigma_V^2)$) by obtaining their likelihood functions. By utilizing these likelihood functions, they derived the loss function to train the model. To train the model, they aimed to maximize the loss function by setting the derivatives with respect to the parameters U and V to zero. Once the training process is complete, the learned feature matrices U and V can be used to predict the missing ratings in the user-item rating matrix R.

To gain a deeper understanding of how PMF operates, we will examine a toy example in the following section. This example will illustrate the factorization of a user-movie rating matrix into two matrices, U and V, using the PMF technique. We will also interpret the resulting factors in terms of user preferences and movie characteristics. Additionally, we will demonstrate how to predict ratings for users who have not rated certain movies, thereby enhancing the accuracy of the recommendation system.

Probabilistic Matrix Factorization: A Toy Example

We will factorize the same matrix used for SVD, Basic Matrix Factorization, and NMF, but this time using PMF. The user-movie rating matrix R is shown below:

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{pmatrix} \end{array}.$$

To factorize this matrix using PMF, we employ an algorithm in Python that utilizes the loss function introduced by R. Salakhutdinov et al. in [8]. For this toy example, we assume two latent factors, "Action" and "Romance," and set $k = 2$. The PMF decomposition yields two matrices: U and V. Below, we will interpret each matrix individually.

User Factors (U Matrix)

As mentioned before, the matrix U illustrates the connection between users and latent factors. Each row in U corresponds to a specific user, while each column represents a latent factor. Higher values within a row signify a stronger association between the user and the respective latent factor.

$$U = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} 1.65 & 0.59 \\ 0.98 & -1.60 \\ 2.09 & 0.43 \\ 2.20 & -0.84 \end{pmatrix} \end{array}.$$

Below, we interpret the matrix U in detail:

- **Alice:** (1.65, 0.59) - Predominantly associated with "Action" and minimally associated with "Romance".
- **Bob:** (0.98, -1.60) - The presence of a negative value makes interpretation difficult.
- **Carol:** (2.09, 0.43) - Strongly associated with "Action" and moderately associated with "Romance".
- **Dave:** (2.20, -0.84) - The presence of a negative value makes interpretation difficult.

Item Factors (V Matrix)

Matrix V depicts the relationship between movies and latent factors. Each row represents a specific latent factor, while each column corresponds to a movie. Higher values in a row indicate a stronger association between the movie and the corresponding latent factor.

$$V = \begin{array}{c} \\ \text{Action} \\ \text{Romance} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 2.35 & 1.42 & 2.35 \\ 0.20 & -2.26 & 0.20 \end{pmatrix} \end{array}.$$

Below, we interpret the matrix V in detail:

- **Matrix (Action movie):** (2.35, 0.2) - Predominantly associated with "Action".

- **Titanic (Romance movie):** (1.42, -2.26) - The presence of a negative value makes interpretation difficult.
- **Inception (Action movie):** (2.35, 0.20) - Predominantly associated with "Action".

Predicting Missing Values:

From the rating matrix R, we can see that Alice did not rate "Inception," and Dave did not rate "Matrix." By using the two obtained matrices U and V and obtain $\hat{R} = UV$, we can predict the missing ratings in matrix R.

$$\hat{R} = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 3.99 & 1 & \mathbf{3.99} \\ 1.98 & 4.99 & 1.98 \\ 4.99 & 1.99 & 4.99 \\ \mathbf{4.99} & 4.98 & 4.99 \end{pmatrix} \end{array}.$$

The predicted rating for "Inception" (Action) for Alice is 3.99, which is high and consistent with her strong association with Action. The predicted rating for "Matrix" (Action) for Dave is 4.99, which is very high, indicating a strong interest in both genres and aligning with our expectations.

Following our overview of the primary matrix factorization techniques, we will embark on a systematic literature review to delve deeper into these methodologies.

2.4 Systematic Literature Review

Jannach et al. [24] were pioneers in applying matrix factorization (MF) techniques to recommendation systems. This model-based collaborative filtering (CF) approach has shown promise in mitigating the issue of data sparsity in recommendation systems [25]. MF is known for its high predictive performance and scalability with large datasets [26].

We conducted a systematic literature review on existing matrix factorization techniques, including Singular Value Decomposition (SVD) [25], Non-negative Matrix Factorization (NMF) Symmetric Nonnegative Matrix Factorization (SNMF), and Probabilistic Matrix Factorization (PMF). These methods represent the main approaches in MF, each with unique strengths and limitations, which we briefly elaborate on below. The goal of this review is to identify gaps that can be addressed by proposing Symmetric Probabilistic Matrix Factorization (SPMF). This thesis aims to tackle the challenges and leverage the opportunities of SPMF by answering the following research questions:

1. What are the comparative advantages of SPMF over existing matrix factorization techniques such as SVD, NMF, and PMF?
2. How can we leverage user-user or item-item interactions into the user-item matrix? And what are the benefits of integrating these knowledge into one single matrix?
3. What are the benefits and trade-offs of transforming the biconvex optimization problem into a convex optimization problem in terms of computational complexity and interpretability?
4. How can the optimal number of latent factors be determined for the SPMF model in recommendation systems, and what impact does this number have on the model's performance?
5. What impact does the choice of probabilistic distribution (e.g., Gaussian and Laplace) in SPMF have on the robustness of recommendations?
6. How to binarize a user-item, item-item, and user-user rating matrix?
7. How to use the developed SPMF for other data structures such as an arbitrary binary matrix?
8. What role does hyperparameter tuning play in the effectiveness of SPMF, and what are the most critical parameters to optimize?
9. How does the proposed SPMF model improve recommendation robustness and interpretability compared to existing methods when evaluated on subsets of the MovieLens 100K and FilmTrust datasets?
10. What are the theoretical and mathematical advantages of SPMF compare to those of state-of-the-art matrix factorization techniques such as NMF and PMF?

Below, we present the keywords used to search databases for relevant articles, dissertations, and conference proceedings.

Keywords:

- Recommendation Systems (RS)
- Symmetric Probabilistic Matrix Factorization (SPMF)
- Symmetric Matrix Factorization (SMF)
- Matrix Factorization (MF)
- Singular Value Decomposition (SVD)
- Non-negative Matrix Factorization (NMF)
- Symmetric Non-negative Matrix Factorization (SNMF)
- Probabilistic Matrix Factorization (PMF)
- Bayesian Probabilistic Matrix Factorization (BPMF)
- Bayesian Inference
- Convex Optimization
- Deep Learning
- Binary Matrix Factorization
- Convergence Rate

To conduct a thorough and systematic literature review, we searched the following databases:

Databases Searched:

- IEEE Xplore
- Inspec
- ProQuest Dissertations & Theses
- Google Scholar
- ScienceDirect

By doing so, we synthesized our findings for SVD, NMF, and PMF. The papers included in the review, the reasons for their inclusion, and their relation to the research questions are discussed below.

Singular Value Decomposition (SVD)

The SVD technique is one of the techniques that generates recommendations for users [27]. It is among the popular MF algorithms that have been applied to solve some CF problems [28]. Guo et al. suggested an approach that applied attribute information to calculate the similarity between items [29]. S. K. Raghuwanshi et al. [30] created an SVD technique that used stochastic gradient descent optimization for speedy convergence of learning parameters to improve prediction accuracy. Al-Sabaawi et al. [31] addressed the sparsity and cold-start issues of CF by suggesting an SVD technique that incorporated social information with a user-rating matrix. To predict unknown relationships with users, they applied Multi-step Resource Allocation (MSRA). They combined the similarity values and user-item rating matrix to obtain the best low-rank linear representation of the user-item matrix. Finally, they enhanced the prediction using Stochastic Gradient Descent (SGD).

Recent advancements in recommendation systems have demonstrated the utility of SVD and its variants across various domains. A book recommendation system utilizing SVD has been effective in handling information overload by capturing latent user preferences and book attributes, resulting in accurate and diverse recommendations [32]. Similarly, a music recommendation system based on SVD and collaborative filtering showed improved performance in making precise recommendations compared to traditional methods [33]. In online course recommendations, SVD was instrumental in predicting user preferences accurately and addressing the cold start problem [34]. Additionally, an improved co-SVD approach for cold-start recommendations enhanced accuracy by reducing data sparsity [35].

Also, SVD is used in binary matrix factorization frameworks. Sahib [36] focused on addressing challenges with massive binary datasets, demonstrating that SVD can manage such data. By converting large binary matrices into low-rank forms, his approach ensures faster data retrieval and processing without information loss. This method was validated on a standard machine, showcasing its practical utility for handling extensive binary data efficiently, and proving the effectiveness of SVD in reducing computational load while maintaining data integrity.

SVD is useful in recommendation systems due to its ability to uncover relationships between items through latent factors. Despite its strengths, SVD has significant drawbacks. It is computationally expensive and slow, particularly with sparse data, making it challenging to find meaningful relationships. Additionally, SVD models can be difficult to interpret because the latent factors they uncover are abstract and do not directly correspond to understandable user or item attributes. This lack of interpretability can impede the ability to explain recommendations to users, which is particularly important in domains like e-commerce where transparency is crucial. The computational cost and interpretability issues collectively affect SVD's scalability and its practical application in recommendation systems [5].

To address these limitations, we introduce SPMF in this study. SPMF handles data sparsity by employing a probabilistic methodology to estimate missing values and generate robust recommendations. Additionally, by transforming the biconvex optimization problem into a convex optimization problem, SPMF reduces the two-factor problem into a single unified low-rank factor, simplifying the factorization process and improving the interpretability of the latent factors. This unified approach not only enhances the understanding of the underlying data structure but also offers several benefits in terms of computational complexity and practical application.

Nonnegative Matrix Factorization (NMF)

NMF has been emerging as a new advancement to both clustering problems and dimension reduction tasks, and it has been investigated and applied for various purposes, especially for high-dimensional data with matrices containing nonnegative values [37]. The ultimate goal of the NMF is to provide a nonnegative low-rank approximation of the original data [37]. The main benefit of this method for collaborative filtering is that, when non-negativity is enforced on non-negative matrices, prediction errors are decreased in comparison to methods like SVD [26].

Additionally, low-rank non-negative matrix factorization allows users to work with compressed dimensional models that typically accelerate efficient statistical classification, clustering, and data organisation, which also accelerates faster searches for patterns or trends [38].

The NMF of the rating matrix has proven to be a successful method in RS [39]. Indeed, NMF characterises the rating matrix by a small number of factors deduced from rating samples, which reduces the dimension of the users' and items' space. This is equivalent to approximating the matrix that contains the observed ratings with a low-rank matrix and using the latter to predict the unobserved ratings.

The non-negativity constraint is a prevalent and significant fact for many data sets. For instance, in image processing, all elements of the data are pixels representing brightness intensities, which are all non-negative values. In text clustering for NLP, each feature is a non-negative element corresponding to the frequency of each keyword in the entire body of the record. Hence, enforcing non-negativity constraints on the resulting factors is necessary to provide meaningful interpretation to the experts in the related fields. Moreover, theoretically it was shown that the extension of the NMF to a symmetric version is equivalent to the variants of other well-known clustering methods such as spectral clustering and K-means algorithms [40].

NMF was first introduced by [41], then it became more popular because of its application to learning the parts of objects, especially for the facial image representation [42]. It is important to note that NMF is a highly ill-posed problem, which means there are infinitely many approximations in homogeneous solution space to the non-negative factorization [7]. Additionally, by considering the NP-hardness of the problem and the existence of non-convex objective functions (Equation 2.20) that result in finding locally optimized solutions U and V, different assumptions have been proposed to alleviate the difficulty of this problem, such as different regularization techniques, sparsity constraints, orthogonality constraints, kernel adoption, and manifold assumptions [43]. Numerical methodologies have shown to be very advantageous to design successful algorithm in many domains related to information searching, retrieval and ranking [44].

Also, NMF is applied in binary matrix factorization. Tomé et al. [45] analyzed binary datasets using Bernoulli statistics and partially NMF of log-odds matrices. Their model imposes constraints that make the estimated basis system strictly nonnegative or even binary, positioning it between logistic PCA and binary NMF. They demonstrated that different model variants yield precise and compact basis systems with toy datasets. Applying the method to the USPS dataset showed good reconstruction quality, even with low-rank binary bases, validating the model's effectiveness for binary data representation. Lumbreras et al. [46] explored binary data matrices in various contexts using NMF with Bernoulli-distributed data. They proposed a Bayesian mean-parameterized framework for NMF without the need for link functions, ensuring interpretability. Three models with different constraints were analyzed, and novel inference methods were derived, including a collapsed Gibbs sampler and a variational algorithm. Their approach also extends to nonparametric settings for automatic latent dimension detection, showing superior performance in dictionary learning and missing data prediction across multiple datasets. Ma et al. [47] proposed a new probabilistic NMF method, decomposing a non-negative matrix into a low-rank factor matrix with binary constraints and a non-negative weight matrix. This approach introduces a deterministic Indian buffet process for variational inference to learn binary features and feature numbers automatically. Additionally, the weight matrix satisfies the exponential prior, and a variational Bayesian exponential Gaussian inference model is used to obtain the posterior distributions of the factor matrices. Comparative experiments on synthetic and real-world datasets demonstrate the method's efficacy.

While NMF is a reliable technique for matrix factorization, offering benefits such as non-negativity constraints, dimensionality reduction capabilities, interpretability, and flexibility in choosing the number of latent factors, it is not without limitations [48]. One significant limitation is that NMF can be sensitive to initialization and may not always converge to a global optimum. Additionally, NMF is a highly ill-posed problem [7] due to the non-existence of a unique solution, leading to unidentifiable factorization.

By introducing SPMF, the two-factor biconvex optimization problem inherent in NMF is transformed into a single-factor convex optimization problem with a unique global optimal solution. This transformation addresses the limitations of NMF by ensuring a unique global optimum, thereby improving the robustness and reliability of the factorization process.

Symmetric Nonnegative Matrix Factorization (SNMF)

Symmetric Nonnegative Matrix Factorization (SNMF) is a variant of NMF that approximates a non-negative symmetric matrix as the product of a non-negative matrix and its transpose. SNMF is particularly useful in clustering tasks, including image processing, semantic analysis, and document categorization. The systematic review by Chen et al. [49] provides a comprehensive overview of the advancements and applications of SNMF, which are summarized below.

Classic SNMF models, such as Orthogonal SNMF, use orthogonal constraints to ensure unique and non-overlapping factors, which is beneficial in applications like graph clustering and document clustering. Sparse SNMF incorporates sparsity constraints to extract local features, valuable in feature selection and image processing. Manifold Structure-Based SNMF combines geometric data with regularization to capture the intrinsic structure of data, useful in genetic data analysis. Pairwise Constraint-Based SNMF integrates must-link and cannot-link constraints to guide clustering based on prior knowledge, enhancing customer segmentation accuracy.

Optimization methods for SNMF include Projected Gradient Descent (PGD), which ensures non-negativity while optimizing, useful in large-scale image processing, and Projected Newton (PNewton), which enhances efficiency in solving large-scale SNMF models, applicable in real-time data clustering like social media analysis. Auxiliary Function Strategies improve robustness and convergence speed in handling complex datasets.

SNMF's applications and performance are evident in various fields: Facial Image Clustering organizes facial images based on features, improving recognition accuracy in security systems. Document Categorization enhances accuracy by extracting meaningful patterns in textual data, aiding in digital libraries' organization. Pattern Clustering in Gene Expression identifies significant gene patterns, contributing to better understanding and treatment strategies in medical research. Additionally, in recommendation systems, SNMF can be used to uncover latent features in user-item interaction matrices, thereby enhancing the quality of recommendations. It helps in addressing issues such as data sparsity and the cold start problem by clustering users and items based on shared features. For instance, in e-commerce, SNMF can improve product recommendations by identifying groups of similar users and products, thus enhancing user satisfaction and loyalty.

SNMF models outperform classical NMF and K-means clustering in accuracy and normalized mutual information (NMI) across various datasets. However, constructing suitable similarity matrices and ensuring scalability remain significant challenges. Moreover, SNMF is limited to nonnegative symmetric matrices and is nonparametric, making it unsuitable for predicting unseen data. In contrast, SPMF introduces a novel method for symmetric matrix factorization through a probabilistic approach. The parametric nature of this approach allows for the prediction of unseen data using the learned model, distinguishing it from nonparametric techniques that merely approximate a given matrix. Additionally, the choice of probabilistic distributions (e.g., Gaussian and Laplace) in SPMF can be tailored to enhance the robustness of recommendations.

Probabilistic Matrix Factorization (PMF)

Probabilistic Matrix Factorization (PMF) [8, 50] is a model-based technique in RS that has been successfully applied in CF. This technique performs well on large, sparse, and imbalanced data and scales linearly with the size of datasets [8, 50]. PMF models have been enhanced by incorporating various advanced techniques to improve recommendation accuracy and address common issues like data sparsity and cold start problems. These enhancements include the integration of deep learning methods, Bayesian approaches, and attention mechanisms [51]. Below we discuss each method briefly.

Deep Learning Integration: One prominent enhancement is the integration of deep learning techniques into PMF. For example, a probabilistic matrix factorization recommendation method based on deep learning (PMFDL) uses convolutional neural networks (CNN) with attention mechanisms to learn item features and long-term and short-term memory networks to learn user features. This combination significantly improves recommendation accuracy compared to traditional PMF and ConvMF methods [51]. Li et al. [52] applied a deep bias probabilistic matrix factorization (DBPMF) to overcome the cold start and sparsity challenges of CF. Indeed, DBPMF uses a CNN to extract the hidden user-item characteristics. Then they incorporated the bias into PMF in order track user rating behavior. Finally, using some datasets, they showed their approach outperformed some of the state-of-the-art models.

Bayesian Approaches: Bayesian Probabilistic Matrix Factorization (BPMF) improves the PMF model by using Gaussian-Wishart priors for the means and covariances of the latent variables instead of the standard zero mean and identity covariance [53]. This modification allows BPMF to calculate distributions that show the range of possible values for the latent variables, which helps in understanding their uncertainty. This approach provides more detailed information about the data, making the model more robust and accurate. As highlighted by Akulwar et al. [54], BPMF avoids the need for parameter tuning and offers better prediction accuracy, especially for sparse datasets. When used in recommendation systems, BPMF, combined with techniques like Cholesky decomposition, Gibbs sampling, and K-nearest neighbor, enhances user satisfaction and loyalty [54]. In BPMF, latent factors are treated as random variables with prior distributions. This allows the model to incorporate uncertainty in the factor estimates, making the model more robust to overfitting and better at handling sparse data. The following techniques have been used in BPMF [55]:

- **Cholesky Decomposition**: This technique is used to decompose the covariance matrix of the latent factors, which simplifies the computation of the posterior distributions. Cholesky decomposition helps in efficiently sampling from multivariate Gaussian distributions, which is essential in the Bayesian framework.
- **Gibbs Sampling**: Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method used to approximate the posterior distributions of the latent factors. It iteratively samples from the conditional distributions of each latent factor, making it possible to handle complex posterior distributions that are difficult to sample from directly.
- **K-Nearest Neighbor Methods**: These methods are used to enhance prediction accuracy by incorporating neighborhood information. In the context of BPMF, they help refine the estimates of the latent factors by considering the similarities between users or items, thereby improving the quality of the recommendations.

These Bayesian techniques collectively enhance the performance of BPMF by providing more reliable and accurate recommendations, particularly in scenarios with sparse and noisy data [54].

Attention Mechanisms: Integrating attention mechanisms with convolutional matrix factorization helps in capturing long-distance dependencies in auxiliary information, such as item comments and categories. This approach alleviates data sparsity and improves the accuracy of rating predictions [51]. Zhang et al. [56] developped an approach which incorporated PMF and attention-based recurrent neural networks to extract item latent feature vectors. They reported significant improvements over existing CF techniques.

Incorporating additional contextual or side information into PMF models has been shown to significantly enhance their performance. This side information can include user reviews, item descriptions, and social trust relationships [57]. Below, we discuss various methods for integrating side information:

- **Textual Data Integration**: Models like Double-ConvMF integrate both item and user textual information using CNNs, extracting features from descriptions and reviews to enhance recommendation accuracy [57].

- **Graph-Based Regularization**: Graph Regularized Probabilistic Matrix Factorization (GRPMF) incorporates expert knowledge through graph-based regularization strategies within an MF framework. This method captures relational information among drugs, which helps in predicting drug-drug interactions [58].

- **Attributes and Finite Mixture Modeling**: Probabilistic matrix factorization models that integrate attribute data using finite mixture modeling can jointly model data and attribute matrices, improving computational efficiency and recommendation accuracy [59].

- **Matrix Factorization with Multimodal Side Information**: Aktukmak et al. [53] address the challenge of enhancing recommendation systems' performance in scenarios with sparse user-item interactions by incorporating side information, such as user demographics and item descriptions. This side information, which includes both numerical and categorical data, is integrated using their Bayesian probabilistic generative framework called MF-MSI (Matrix Factorization with Multimodal Side Information). They develop a scalable optimization method based on variational EM to learn the posterior distributions of latent variables. Experiments on simulated and real datasets demonstrate that incorporating side information improves prediction accuracy and ranking performance, outperforming many popular baseline models. The MF-MSI model consistently produces better results compared to other recent recommendation systems that also utilize side information.

PMF models can be further enhanced by integrating multiple sources of information, such as user ratings, review sentiments, and rating reliability. This comprehensive approach ensures more reliable and accurate recommendations [60]. Below, we discuss methods for integrating multiple sources of information:

- **Sentiment Analysis and Rating Reliability**: Incorporating sentiment analysis and noise detection methods into PMF allows for handling user reviews and determining the reliability of user ratings. This approach aims to enhance the performance of recommendation systems by leveraging multiple types of information [60]. Puja Hari and Sisodia [61] proposed a PMF algorithm to address the sparsity issue of CF and improve its accuracy. Their algorithm accepts the preference relationship as input for organizing items. Additionally, they incorporated other relevant information about the user and item into the algorithm, and reported that their model performed better than some existing models.

- **Trust Relationships and Interest Mining**: Integrating user trust relationships, both direct and indirect, as well as item correlations and user interest mining, addresses data sparsity and cold start problems. This approach enhances recommendation accuracy and diversity by considering comprehensive user and item interactions [62]. To address the cold start problem of the CF, Dong et al. [63] integrate trust relationships between users into the traditional matrix factorization.

Advanced techniques in probabilistic matrix factorization include robust formulations, semi-nonnegative matrix factorization, symmetric nonnegative matrix factorization, and latent semantic analysis [64, 65]. Below, we discuss each method briefly:

- **Robust PCA and Variants**: Robust PCA models decompose data matrices into low-rank components while accounting for sparse and dense noise, enhancing robustness to outliers. Bayesian and variational Bayesian approaches to robust PCA have been proposed to handle different types of noise [64].

- **Semi-Nonnegative Matrix Factorization**: Probabilistic semi-NMF formulations use variational inference techniques to handle noise robustly. These models have shown improved performance in various applications, such as signal processing and machine learning [64].

- **Symmetric Nonnegative Matrix Factorization**: He et al. [66] focus on Symmetric Nonnegative Matrix Factorization (SNMF), a special case of NMF decomposition useful in various applications, including image processing and semantic analysis of documents. They introduce three parallel multiplicative update algorithms using level 3 basic linear algebra subprograms (BLAS) to solve the SNMF problem. The first algorithm minimizes the Euclidean distance, with proven convergence under mild conditions. Building on this, they propose two additional fast parallel methods: α-SNMF and β-SNMF algorithms. These algorithms are used for probabilistic clustering and are shown to work well for facial image clustering, document categorization, and pattern clustering in gene expression. Their study highlights the utility of SNMF in clustering tasks by leveraging efficient and scalable update algorithms.

- **Latent Semantic Analysis and Extensions**: Probabilistic latent semantic analysis (PLSA) and its extensions, including non-parametric formulations, provide probabilistic interpretations of latent factors. These models have been used in various applications in conjunction with techniques like PCA and transfer learning [65].

While PMF offers several advantages, such as handling sparsity, scalability to large datasets, a probabilistic framework that incorporates uncertainty, and often providing interpretable latent factors, it also has some drawbacks. These include computational complexity, sensitivity to initialization, and potential convergence issues, particularly with highly sparse data or poor initialization, which can lead to local minima rather than the global optimum.

SPMF addresses these limitations by transforming the original matrix into a symmetric blockwise matrix. To overcome the non-uniqueness inherent in PMF, SPMF approximates this blockwise matrix by incorporating user-user and item-item information into a single low-rank factor to solve a convex objective function. This approach not only ensures a global optimal solution with a faster convergence rate but also enhances interpretability by modeling user-user and item-item interactions from a probabilistic perspective.

Excluded Papers

The systematic literature review is based on papers that are directly related to the research questions. Below, we list some papers that were found to have no relation to the research questions.

- Ramathulasi, T. and Babu, M. R., 2023, [67], Assessing User Interest in Web API Recommendation using Deep Learning Probabilistic Matrix Factorization.

- Yang, X., Luo, Y., Fu, S., Xu, M., and Chen, Y., 2022, [68], DPMF: Decentralized Probabilistic Matrix Factorization for Privacy-Preserving Recommendation.

- Wei, X., Lin, Z., Liu, T., and Zhang, L., 2022, [69], Probabilistic Matrix Factorization for Visual Tracking.

- Li, X. and Zhang, L., 2022, [70], Probabilistic Matrix Completion.

- Qiu, L., Zhou, W., and Yang, Y., 2023, [71], STPR: Personalized Recommendation via Matrix Factorization with Social Ties.

- Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S., 2019, [72], Neural Graph Collaborative Filtering.

- Li, G., Yang, D., Nobel, A. B., and Shen, H., 2016, [73], Supervised Singular Value Decomposition and its Asymptotic Properties.

- Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., and Zhang, F., 2017, [74], A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems.

- Sedhain, S., Menon, A. K., Sanner, S., and Xie, L., 2015, [75], Autorec: Autoencoders Meet Collaborative Filtering.

- Wang, H., Zhang, F., Xie, X., and Guo, M., 2018, [76], DKN: Deep Knowledge-Aware Network for News Recommendation.

After conducting a systematic literature review of the mentioned articles on matrix factorization, we selected some key papers and performed a synthetic comparison. This comparison is presented in tables to illustrate their relationship with the research questions.

2.5 Synthetic Comparison of Selected Papers

In this section, we present a synthetic and systematic comparison of selected papers included in the literature review. These comparisons are organized into following tables, highlighting their key findings and their relation to the research questions established earlier in this chapter.

Table 2.1 Systematic Literature Review : MF and SVD

Citation	**Focus Area**	**Research Question(s) Addressed**	**Key Findings**	**Relevance to Research Questions**
Jannach et al., 2010 ([24])	MF	What are the comparative advantages of SPMF over existing matrix factorization techniques?	Introduces basic matrix factorization in RS	RQ1, RQ4
Sharifi et al., 2013 ([25])	SVD	What are the comparative advantages of SPMF over existing matrix factorization techniques?	Demonstrates potential in addressing the challenge of data sparsity in RS	RQ1, RQ4
Isinkaye, 2021 ([26])	SVD	What are the comparative advantages of SPMF over existing matrix factorization techniques?	Surveys literature review on all existing MF models	RQ1, RQ3, RQ4
Abdollahi et al., 2013 ([27])	SVD	How can we leverage user-user or item-item interactions into the user-item matrix?	Explores the integration of explainability into matrix factorization techniques used in RS	RQ1, RQ2, RQ3
Balakrishnan et al., 2023 ([32])	SVD	How can the optimal number of latent factors be determined for the SPMF model in recommendation systems, and what impact does this number have on the model's accuracy, efficiency, and overall performance?	Captures latent user preferences and book attributes, resulting in accurate and diverse recommendations	RQ4
Bahar et al., 2023 ([34])	SVD	What are the comparative advantages of SPMF over existing matrix factorization techniques?	SVD was instrumental in predicting user preferences accurately and addressing the cold start problem	RQ1, RQ3

Table 2.2 Systematic Literature Review: SVD and NMF

Citation	Focus Area	Research Question(s) Addressed	Key Findings	Relevance to Research Questions
Ming et al., 2022 ([35])	SVD	How can we leverage user-user or item-item interactions into the user-item matrix?	Improved co-SVD approach for cold-start recommendations enhanced accuracy by reducing data sparsity	RQ2 RQ3
Sahib, 2024 ([36])	SVD	What are the advantages of binarizing the matrix in improving the simplicity and efficiency of the SPMF model?	Application of SVD for binary data matrices in RS to improve recommendation accuracy and computational efficiency	RQ6, RQ7
Tomé et al., 2015, ([45])	NMF	How to use the developed SPMF for other data structures such as an arbitrary binary matrix?	Proposes a logistic NMF method tailored for binary datasets to enhance modeling accuracy and interpretability	RQ6, RQ7
Hinako et al., 2023 ([77])	NMF	How to use the developed SPMF for other data structures such as an arbitrary binary matrix?	Proposes a non-negative/binary matrix factorization method for image classification utilizing quantum annealing techniques	Q6, RQ7
Ma et al., 2021 ([47])	NMF	How to use the developed SPMF for other data structures such as an arbitrary binary matrix?	Proposes a probabilistic NMF approach with binary components to enhance feature learning and model interpretability	Q6, RQ7
Guillaume et al., 2023 ([78])	Convergence Rate	What are the theoretical and mathematical advantages of SPMF compare to those of state-of-the-art matrix factorization techniques such as NMF and PMF	Provides a comprehensive collection of convergence theorems for both stochastic and deterministic gradient methods	Q10

Table 2.3 Systematic Literature Review: NMF and PMF

Citation	**Focus Area**	**Research Question(s) Addressed**	**Key Findings**	**Relevance to Research Questions**
Janecek et al., 2022 ([79])	NMF	What role does hyperparameter tuning play in the effectiveness of SPMF?	Suggested population-based algorithms for initialization	RQ8
Ran et al., 2022 ([39])	NMF	How can the optimal number of latent factors be determined for the SPMF model?	Highlighted NMF's success in RS	RQ1
Gillis, 2012 ([7])	NMF	What are the benefits and trade-offs of transforming the biconvex optimization problem into a convex optimization problem in terms of computational complexity and interpretability?	Highlighted NMF's ill-posed problem	RQ3
Schmidt et al., 2009 ([80])	NMF	What impact does the choice of probabilistic distribution (e.g., Gaussian and Laplace) in SPMF have on the robustness of recommendations?	Develops a Bayesian framework for NMF, enhancing interpretability and model robustness through probabilistic approaches.	RQ1, Q5
Wild et al. 2004 ([81])	NMF	What role does hyperparameter tuning play in the effectiveness of SPMF?	roposes improvements toNMF methods by using structured initialization techniques to enhance performance and convergence	RQ8
Boutsidis et al., 2008 ([82])	NMF	What role does hyperparameter tuning play in the effectiveness of SPMF, and what are the most critical parameters to optimize?	Suggested PCA and SVD initialization for NMF	RQ8

Table 2.4 Systematic Literature Review : PMF and BPMF

Citation	**Focus Area**	**Research Question(s) Addressed**	**Key Findings**	**Relevance to Research Questions**
Salakhutdinov et al., 2007, [8]	PMF	What are the comparative advantages of SPMF over existing matrix factorization techniques?	Achieves notable performance on huge, sparse, and imbalanced data; scales linearly with the number of datasets	RQ1
Gai et al., 2016 [50]	PMF	What are the benefits and trade-offs of transforming the biconvex optimization problem into a convex optimization problem?	Demonstrates high predictive performance and scalability for large datasets	RQ2, RQ3
Xiaoyue et al., 2019, [51]	Deep Learning Integration	How does the SPMF handle sparsity in the user-item rating matrix compared to traditional matrix factorization methods?	Integration of CNN and attention mechanisms improves recommendation accuracy significantly compared to traditional methods	RQ4
Kangkang et al., 2019, [52]	Deep Learning Integration	How does the SPMF handle sparsity in the user-item rating matrix compared to traditional matrix factorization methods?	Overcomes cold start and sparsity challenges using CNN to extract hidden user-item characteristics	RQ4
Pooja et al., 2016, [54]	BPMF	What impact does the choice of probabilistic distribution in SPMF have on the robustness of recommendations?	Introduces probabilistic interpretations of latent factors, enhancing recommendation accuracy	RQ5
Jiarong et al., 2017, [55]	Survey on PMF	What impact does the choice of probabilistic distribution in SPMF have on the robustness of recommendations?	Uses techniques like Cholesky decomposition, Gibbs sampling, and K-nearest neighbor methods for better predictions	RQ5

Table 2.5 Systematic Literature Review : SNMF, PMF and BPMF

Citation	**Focus Area**	**Research Question(s) Addressed**	**Key Findings**	**Relevance to Research Questions**
Stuti et al., 2023, [58]	Graph-Based Regularization	How to use the developed SPMF for other data structures such as an arbitrary binary matrix?	Incorporates expert knowledge through graph-based regularization for drug-drug interaction prediction	RQ6, RQ7
Lifeng et al., 2022, [62]	PMF	How to binarize a user-item, item-item, and user-user rating matrix?	Integrates user trust relationships and item correlations to address data sparsity	RQ6
Junbin et al., 2023, [64]	PMF	What impact do different binarization thresholds have on the performance of binary SPMF in recommendation systems?	Develops a probabilistic semi-nonnegative matrix factorization framework using maximum-likelihood estimation and variational inference techniques	RQ5
Pau et al., 2019, [53]	BPMF	What impact does the choice of probabilistic distribution in SPMF have on the robustness of recommendations?	Proposes a probabilistic framework for matrix factorization incorporating multimodal side information to handle mixed-data type features	RQ5
Zhaoshui et al., 2011, [66]	SNMF	What impact does the choice of probabilistic distribution (e.g., Gaussian and Laplace) in SPMF have on the robustness of recommendations?	Proposes symmetric nonnegative matrix factorization algorithms and their applications to probabilistic clustering	RQ4

After conducting a systematic literature review, we are prepared to highlight the research gaps that this thesis aims to address.

2.6 Research Gaps

The motivation for this research is derived from several key gaps identified in the current literature on matrix factorization techniques for recommendation systems:

- **Limited Robustness in Sparse Data:** Existing matrix factorization methods, such as SVD, NMF, often struggle with sparse datasets, leading to poor performance and inaccurate recommendations. There is a need for models that can maintain robustness in the presence of sparse data.

- **Application Limitations of Symmetric Approaches:** Methods like SMF and SNMF are restricted to symmetric matrices, limiting their applicability. There is a need for models that can handle non-symmetric matrices while leveraging the benefits of symmetric factorization.

- **Inadequate Handling of Binary Ratings:** Traditional matrix factorization approaches often struggle with binary rating matrices, which are common in many practical applications. There is a gap in research addressing the factorization of binary matrices to improve recommendation relevance and interpretability.

- **Lack of Incorporation of Symmetric Interactions:** Most existing methods do not integrate symmetric user-user and item-item interactions into the recommendation process. There is a need for models that can leverage these interactions to enhance the quality of recommendations.

- **Insufficient Exploration of Probabilistic Distributions:** Current approaches often use simplistic probabilistic distributions, such as Gaussian, without exploring alternative distributions that might offer better robustness to noise and outliers. This represents a gap in the exploration of different probabilistic frameworks within matrix factorization.

- **Challenges in Optimization and Convergence:** Many matrix factorization techniques face difficulties in optimization and convergence, especially with non-convex problems. There is a gap in transforming these problems into convex ones to ensure unique optimal solutions and better convergence properties.

- **Hyperparameter Tuning Issues:** Optimizing hyperparameters is a challenge in current matrix factorization methods, often requiring extensive and inefficient trial-and-error approaches. There is a need for systematic methods to optimize hyperparameters.
- **Limited Interpretability of Models:** The interpretability of recommendations is crucial for user trust and satisfaction. Current methods often lack mechanisms to ensure that the generated recommendations are interpretable. There is a gap in creating models that provide clear and understandable recommendations.

Addressing these research gaps will advance the field of matrix factorization in recommendation systems, enabling more robust, and interpretable recommendations. By identifying these gaps, we are prepared to present the objectives of this thesis.

2.7 Research Objectives

The primary objectives of this thesis are outlined as follows:

- **Develop and Validate SPMF Model:** To develop a Symmetric Probabilistic Matrix Factorization (SPMF) model that addresses the limitations of existing matrix factorization techniques such as SVD, NMF, and PMF.
- **Incorporate User-Item Interactions:** To explore the integration of user-user and item-item interactions into a unified user-item matrix, enhancing the interpretability and robustness of recommendations.
- **Transform Optimization Problem:** To transform the biconvex optimization problem into a convex optimization problem, ensuring a unique optimal solution and improved convergence properties.
- **Determine Optimal Latent Factors:** To identify the optimal number of latent factors for the SPMF model, balancing model complexity and performance, and enhancing recommendation accuracy.
- **Utilize Probabilistic Distributions:** To investigate the impact of different probabilistic distributions (e.g., Gaussian and Laplace) on the robustness of the SPMF model in handling noise and outliers.

- **Binarize User-Item Matrix:** To develop methods for binarizing the user-item, item-item, and user-user rating matrices, simplifying the factorization process and improving the relevance of binary data recommendations.
- **Extend SPMF to SPBMF:** To modify the SPMF model to handle binary matrices using the Bernoulli distribution, creating the Symmetric Probabilistic Binary Matrix Factorization (SPBMF) approach.
- **Optimize Hyperparameters:** To identify and optimize critical hyperparameters of the SPMF model, achieving the best possible performance through systematic tuning.
- **Comprehensive Performance Evaluation:** To conduct extensive experiments and evaluations using real-world datasets, demonstrating improvements in recommendation robustness and interpretability over existing methods.
- **Theoretical and Mathematical Analysis:** To analyze the theoretical and mathematical properties of SPMF, proving its advantages in terms of convergence rate and optimization efficiency compared to state-of-the-art matrix factorization techniques.

By highlighting the objectives, we now present the methodology that will be used to achieve the research goals.

2.8 Research Methodologies

The development of the proposed SPMF model involves several innovative steps, which are highlighted below:

- **Constructing a Symmetric Block Matrix**: An innovative paradigm for constructing a symmetric block matrix that integrates the user-item rating matrix with user-user and item-item matrices. This approach combines diverse user-user and item-item interactions within the recommendation systems to facilitate personalized recommendations with interpretable low-rank factorization.
- **Algorithm Design**: Designing a new algorithm for consolidating two distinct factors into a single unified low-rank factor, representing the holistic data matrix. This technique enhances the interpretability of matrix factorization methods, providing a more comprehensive understanding of the underlying data structure.

- **Binarization Framework**: Establishing a new framework for binarizing a data matrix in the context of item recommendation systems. This method translates user preferences into binary ratings, simplifying the complexity of preference data while maintaining interpretability and usefulness of recommendations.

- **SPBMF Development**: Developing a novel method called Symmetric Probabilistic Binary Matrix Factorization (SPBMF) by incorporating the Bernoulli distribution. This innovation is particularly suited for handling arbitrary binary matrices, enhancing the flexibility and applicability of probabilistic matrix factorization techniques in various binary data contexts.

- **Probabilistic Methodologies**: Applying probabilistic methodologies to tackle data sparsity and estimate missing values. This includes exploring Maximum Likelihood Estimation (MLE) with Normal and Laplace distributions, and Bayesian estimation using a normal distribution for the error term and a normal prior distribution for the entries of the low-rank matrix.

- **Theoretical Convergence Analysis**: Conducting an analysis of the theoretical convergence rate of SPMF, providing insights into the efficiency and computational cost of this method in approximating symmetric matrices. This analysis will be compared against state-of-the-art techniques to highlight improvements in convergence rate achieved by SPMF.

- **Experimental Design**: To test and validate the SPMF model, we will design and execute a series of experiments focusing on evaluating its accuracy, efficiency, and robustness. Additionally, we will analyze the impact of various probabilistic distributions. This examination will ensure an understanding of the model's performance and potential areas for improvement.

In the following chapter, we present our proposed SPMF model along with a thorough explanation of its structure and underlying principles.

CHAPTER 3 SYMMETRIC PROBABILISTIC MATRIX FACTORIZATION FOR RS

This chapter outlines the methodologies employed to address the research questions that drive this thesis. Specifically, we focus on the following questions:

- How can SPMF enhance the interpretability and robustness of recommendation systems in sparse data environments?
- How can we leverage user-user or item-item interactions into the user-item matrix? And what are the benefits of integrating these knowledge into one single matrix?
- What are the benefits and trade-offs of transforming the biconvex optimization problem into a convex optimization problem in terms of computational complexity and interpretability?
- How can the optimal number of latent factors be determined for the SPMF model in recommender systems, and what impact does this number have on the model's performance?
- What is the impact of different probabilistic distributions (e.g., Gaussian and Laplace) on the robustness of the SPMF model in handling noise and outliers?
- How to binarize a user-item, item-item, and user-user rating matrix?
- How can the new Symmetric Probabilistic Binary Matrix Factorization (SPBMF) model, using the Bernoulli distribution, handle binary matrices?
- What are the critical hyperparameters for the SPMF model, and how can we optimize them for best performance?
- How can the convergence rate of the SPMF model be improved compared to existing methods?
- What are the implications of using different regularization techniques on the performance and generalizability of the SPMF model?

To answer these questions, this chapter proposes the Symmetric Probabilistic Matrix Factorization (SPMF) model. Our approach includes transforming the biconvex optimization problem into a convex one, ensuring unique optimal solutions and improved convergence properties. A theoretical convergence analysis is conducted to demonstrate the efficiency of SPMF compared to existing methods. Additionally, toy examples for SPMF and SPBMF are provided to illustrate the proposed methodology in a simplified context. Through this methodology, we aim to develop a robust and interpretable recommendation system, providing advancements in the field of matrix factorization for recommendation systems.

Typically, conventional recommendation systems are designed to suggest items to users based solely on a given user-item rating matrix, without considering additional insights from user-user or item-item relationships. This significant limitation motivates the development of a novel approach for recommendation systems, aimed at enriching the information derived from the user-item rating matrix by incorporating interactions from user-user and item-item relationships. The proposed method, called Symmetric Probabilistic Matrix Factorization (SPMF), enhances the interpretability of recommendations.

It's worth noting that the term SPMF is already present in the literature, commonly applied to symmetric matrices, thus limiting its use. However, the SPMF model proposed in this thesis is designed to symmetrize any arbitrary rectangular matrix. This innovation expands the model's applicability beyond symmetric matrices, making it suitable for a broader range of real-world datasets that are typically non-symmetric.

To generate reliable suggestions for users and guide them toward items of interest, our proposed SPMF framework introduces a symmetric block matrix denoted as S. The construction of matrix S is based on the user-item rating matrix R, which captures the preferences or ratings given by users for different items. R is a matrix of size $m \times n$, where m represents the number of users and n represents the number of items. The formulation of S is as follows:

$$S = \begin{bmatrix} R^T R & R^T \\ R & RR^T \end{bmatrix}. \tag{3.1}$$

In this formulation, the upper-left block $R^T R$ captures the relationships between items, while the lower-right block RR^T represents the relationships between users. The off-diagonal blocks R^T and R capture the user-item interactions, providing information on how users are related to items. In Figure 3.1, the construction of the block matrix S is illustrated.

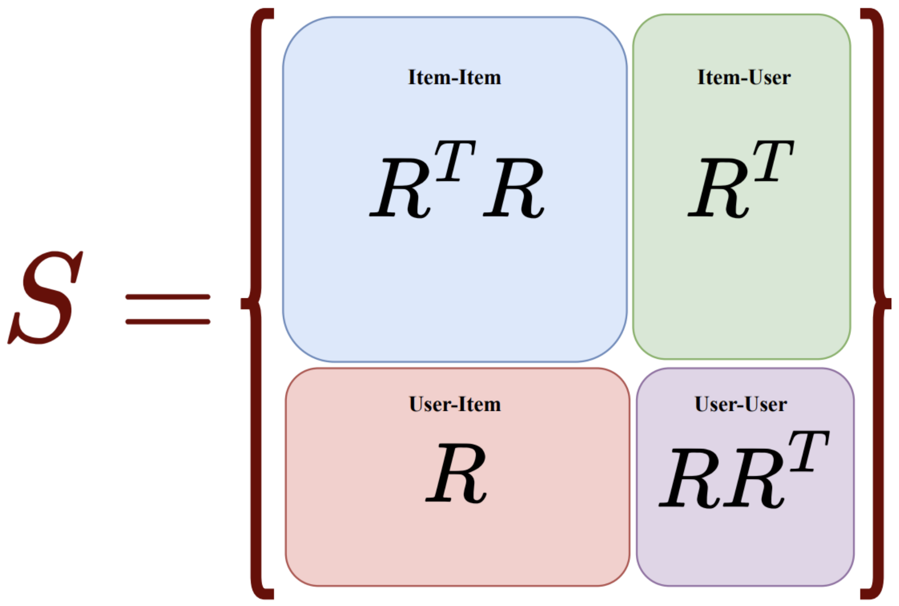

Figure 3.1 SPMF Block Matrix

Specifically, the matrix S serves as a unified representation of the relationships between users, items, and their interactions. By considering user-user and item-item connections, as well as user-item interactions simultaneously, S provides a holistic view of the underlying dynamics in the recommendation system.

The block matrix S is composed of four distinct blocks, such that, the off-diagonal blocks, denoted as R and R^T, represent user-item ratings within the range of 1-5. In contrast, the diagonal blocks, represented by $R^T R$ and RR^T, encapsulate item-item and user-user interactions, respectively. These diagonal blocks often exhibit values that deviate significantly from the user-item rating range, disrupting the consistency between the diagonal and off-diagonal blocks. When employing probabilistic approaches to fit the data to a unified distribution, such as a normal distribution or Laplace distribution, it is crucial to consider the influence of the data's range and scale on the distribution parameters (mean and variance for the normal distribution, or location and scale for the Laplace distribution). Inconsistent scales across different parts of the matrix can lead to inaccurate representation by the fitted distribution. In other words, a mixture model would be required to fit such a block matrix when the scale and range of the data are inconsistent. However, using a mixture model can also introduce an unnecessary level of complexity to the factorization approach, making it more challenging to achieve accurate and interpretable results. Therefore, by normalizing the data across all blocks, we ensure consistency and facilitate the factorization process. To address this, we normalize the diagonal blocks, denoted by $R_{RN}R_{RN}^T$ and $R_{CN}^T R_{CN}$, in a row-wise and column-wise manner, respectively. Furthermore, R and R^T are normalized row-wise denote by R_{RN} and R_{RN}^T , considering their respective user-item and item-user connotations. This normalization harmonizes the scales across all blocks, thereby improving the accuracy and interpretability of the resulting factorization.

By normalizing the user-item rating matrix R column-wise to obtain R_{CN}, as proven in Section 2.1.4, we can consider $R_{CN}^T R_{CN}$ as the item-item similarity matrix and $R_{RN}R_{RN}^T$ as the user-user similarity matrix, with the normalization performed row-wise. It's crucial to note that, as outlined in 2.1.3, before normalization, any missing values in R are replaced by 0. To approximate the block matrix S and capture the relationships between user-user, item-item, and user-item, we use R_{RN} and R_{CN} instead of R when applying the SPMF technique. Specifically, the matrix S is reconstructed as follows:

$$\mathcal{S} = \begin{bmatrix} R_{CN}^T R_{CN} & R_{RN}^T \\ R_{RN} & R_{RN}R_{RN}^T \end{bmatrix}. \tag{3.2}$$

By reconstructing S in this manner, SPMF integrates both the collaborative filtering (CF) aspect (user-user and item-item relationships) and the content-based (CB) aspect (user-item interactions). This combination facilitates a understanding of the recommendation system's dynamics. Below, we explain how SPMF accomplishes this integration:

Collaborative Filtering Aspect:

- The lower-right block $R_{RN}R_{RN}^T$ captures user-user relationships by computing the similarity between users based on their ratings. This is a common approach in user-based collaborative filtering, where the goal is to recommend items liked by similar users.
- The upper-left block $R_{CN}^T R_{CN}$ captures item-item relationships by computing the similarity between items based on the ratings they receive from users. This is a common approach in item-based collaborative filtering, where the goal is to recommend items similar to those liked by the user.

Content-Based Aspect:

- The off-diagonal blocks R_{RN} and R_{RN}^T capture direct user-item interactions. These blocks reflect how users rate specific items, providing information about individual user preferences for specific items. This aspect aligns with content-based filtering, where recommendations are made based on the attributes of the items that a user has shown interest in.

By integrating both CF and CB aspects into a single symmetric block matrix $\mathcal{S}$, SPMF offers a more robust and comprehensive framework for recommendation. This integration allows the model to leverage the strengths of both approaches, improving the interpretability of the recommendations.

To approximate the matrix $\mathcal{S}$, a low-rank matrix approximation approach is employed. Additionally, SPMF addresses the drawbacks of biconvex optimization by transforming it into a single convex cost function, thus eliminating the need for the additional matrix V required in PMF. The matrix U, with dimensions $(n+m) \times k$, represents the low-rank approximation of $\mathcal{S}$. The rank k determines the level of approximation and controls the complexity of the model. This approach involves approximating the matrix S as follows:

$$\mathcal{S} \approx UU^T. \tag{3.3}$$

The approximation of the block matrix $\mathcal{S}$ using the SPMF approach serves two essential purposes within recommendation systems. Firstly, it enables the exploration and understanding of the intricate connections between users and items. Through this approximation, the system gains insights into the underlying patterns governing user-item interactions. By "patterns," we refer to the latent structures and regularities that emerge from user behaviors and item characteristics. These patterns include trends such as users with similar preferences tending to rate items similarly, or certain items being consistently rated higher by particular groups of users. Identifying these patterns helps in predicting user preferences for items they have not yet rated, thereby improving the accuracy and reliability of recommendations. For instance, if a subset of users consistently gives high ratings to a particular genre of movies, this pattern can be used to recommend similar movies to other users with comparable rating behaviors. Similarly, if certain items frequently appear together in user preferences, understanding this pattern can help in suggesting items that align with the user's taste.

Secondly, the existence of missing values within the user-item rating matrix R_{RN} presents a significant challenge. In contrast to methods such as SVD and NMF, which impose no constraints on the prediction range, the probabilistic approach leverages the intrinsic properties of probabilistic distributions in the objective function to predict ratings within an acceptable range for missing values. The goal in approximating the block matrix $\mathcal{S}$ using the SPMF approach is to accurately predict these missing values. By doing so, we address the gaps in the matrix R_{RN}, thus enhancing the system's predictive capabilities and refining its ability to provide precise recommendations.

3.1 Intuition of the Block Matrix $\mathcal{S}$ Construction

As noted in Section 2.3.2, matrix factorization is a biconvex problem, meaning it has two objective functions—one for user features and another for item features—that are convex separately but not jointly convex with respect to both U and V simultaneously due to the product UV [83]. In other words, it is convex in U when V is fixed and convex in V when U is fixed. While matrix factorization is biconvex, finding a global optimum is generally NP-hard due to the non-convexity of the joint optimization problem over both U and V. Moreover, there are infinitely many solutions for the conventional matrix factorization problem, such as basic MF, NMF, or PMF, due to the presence of any arbitrary diagonal matrix D and its inverse. These can be placed in the middle of the approximation $R \approx (UD)(D^{-1}V)$ to account for a new solution $R \approx U'V'$. Therefore, practical algorithms focus on finding local solutions that minimize the objective function [83].

To overcome this, the symmetric block matrix $\mathcal{S}$ is defined and approximated by merit of the inherent symmetry of the problem. In this thesis, we aim to develop a modification that addresses the drawback of biconvexity by consolidating two factors into one single low-rank matrix U with a convex objective function. This approach leads to a global optimal solution for approximating the original block matrix. Additionally, this transformation unfold the existing relationships between user-user, item-item, and user-item interactions. We employ the Bayesian approach and Maximum Likelihood Estimation (MLE) to approximate the block matrix $\mathcal{S}$, as described in the following section.

3.2 Approximating the Block Matrix $\mathcal{S}$

To approximate the block matrix $\mathcal{S}$, we model it as:

$$\mathcal{S} = UU^T + \epsilon, \tag{3.4}$$

where U is an $(m+n) \times k$ matrix of parameters, and ϵ represents a random error matrix. Our goal is to estimate the matrix U, denoted as $\hat{U}$, using a probabilistic approach that minimizes the Frobenius norm $\|\mathcal{S} - UU^T\|_F^2$. Based on Equation (3.4), an approximation for the matrix $\mathcal{S}$ is expressed as:

$$\hat{\mathcal{S}} = \hat{U}\hat{U}^T. \tag{3.5}$$

If we approach the approximation of $\mathcal{S}$ deterministically, we can proceed as follows: Since $\mathcal{S}$ is symmetric, it is diagonalizable. Thus, we can write $\mathcal{S}$ as:

$$\mathcal{S} = Q\Lambda Q^T,$$

where Q is an orthogonal matrix and Λ is a diagonal matrix containing the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n+m}$. We then select the k largest eigenvalues and corresponding eigenvectors, forming the diagonal matrix Λ_k and the matrix Q_k with the corresponding k eigenvectors. Construct U as $U = Q_k\Sigma_k$, where Σ_k is a diagonal matrix containing the square roots of the k largest eigenvalues from Λ_k:

$$\Sigma_k = \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_k}).$$

Given that the sum of the eigenvalues ($\text{tr}(\mathcal{S}) = m + n$) is positive, there must be at least one positive eigenvalue, allowing us to construct Σ_k with the square roots of the positive eigenvalues. The matrix $U = Q_k\Sigma_k$ is an optimal solution to the minimization problem. Substituting U back into the objective function shows that:

$$UU^T = Q_k\Sigma_k\Sigma_k^TQ_k^T = Q_k\Lambda_kQ_k^T.$$

This demonstrates that the rank-k approximation of $\mathcal{S}$ is achieved by UU^T, minimizing the Frobenius norm $\|\mathcal{S} - UU^T\|_F^2$.

However, for a deterministic approach, eigenvalue decomposition has a time complexity of $O(n^3)$ for an $n \times n$ matrix, which can be computationally expensive for large matrices. For extremely large matrices, the computational and memory costs of eigenvalue decomposition make this approach impractical. This approach is not well-suited for dynamic or streaming data where the matrix $\mathcal{S}$ changes over time. Recomputing the eigenvalue decomposition every time $\mathcal{S}$ changes is infeasible for real-time applications. When $\mathcal{S}$ is sparse, eigenvalue decomposition may become more complicated due to the need to handle sparsity effectively.

To address these challenges, our probabilistic approach offers an alternative for estimating the block matrix $\mathcal{S}$, enabling the management of large and dynamic datasets. Probabilistic matrix factorization methods provide several benefits over deterministic approaches that rely on minimizing norm-based error terms through gradient descent. These methods incorporate the concept of data distribution, which allows for better handling of noise and outliers. By modeling the data distribution, probabilistic methods can manage variability in the data, reducing the impact of noisy or anomalous data points on the learned model [8].

Minimizing error solely using gradient descent on the Frobenius norm or L_2 norm often increases the risk of overfitting, especially in high-dimensional data with many parameters. However, probabilistic approaches mitigate overfitting by smoothing the model parameters according to specified prior distributions, leading to better generalization on unseen data. Additionally, probabilistic methods aim to find parameter estimates that conform to a distribution that better fits the entire dataset. This holistic view ensures that the predictions are not merely point estimates minimizing a specific loss function but are representative of the underlying data distribution. This approach provides more reliable predictions, especially in complex and noisy datasets.

The parametric nature of the probabilistic approach enables the utilization of the learned model for the prediction of unseen data, distinguishing it from nonparametric techniques that merely approximate a given matrix. By learning a distribution over the latent factors, probabilistic matrix factorization methods can generate new data points following the learned distribution, making them suitable for tasks such as recommendation systems where predicting future user preferences is crucial. Probabilistic methods also offer flexibility and adaptability by enabling the incorporation of prior knowledge and providing a natural framework for uncertainty quantification [8].

In the context of probabilistic matrix factorization for recommendation systems, the normal distribution is a natural choice for capturing the central tendency of user ratings in our proposed SPMF method. Its symmetry and light tails facilitate the use of statistical techniques such as maximum likelihood estimation and Bayesian inference, ensuring computational efficiency and theoretical soundness. The normal distribution is particularly appropriate when the data adheres closely to the assumption of Gaussian noise. However, its sensitivity to outliers can be a drawback, as extreme values may disproportionately influence the model, potentially distorting the representation of user preferences [84].

In such scenarios, employing the Laplace distribution offers several advantages. The Laplace distribution's robustness to outliers, due to its heavier tails compared to the normal distribution, ensures that extreme values in user ratings do not disproportionately influence the model, which is crucial for accurately capturing user preferences. Additionally, the Laplace distribution enhances the interpretability of the recommendation system by promoting sparsity in the factorized matrices. This is inspired by the L_1 norm as a regularizer, similar to the Lasso sparsity measure, which selects the minimum viable features for prediction. Its mathematical tractability facilitates the implementation of probabilistic models, making it a suitable choice for managing the variability and peculiarities inherent in real-world user rating data [84].

Thus, the choice between the normal and Laplace distributions depends on the specific characteristics of the data and the desired properties of the recommendation system. The normal distribution is preferable for datasets that closely follow Gaussian noise assumptions, whereas the Laplace distribution is advantageous for handling data with outliers and promoting sparsity. Considering this trade-off, the choice of an appropriate distribution can enhance the robustness, and interpretability of our proposed SPMF method. To estimate the matrix U using our proposed SPMF approach, we investigate the following methodologies: Maximum Likelihood Estimation (MLE) with a normal distribution, MLE with a Laplace distribution, and Bayesian estimation with a normal distribution for the error term and a normal distribution as a prior for the entries of U.

1. **Maximum Likelihood Approach using Normal Distribution:** This method assumes that the error term ϵ follows a spherical normal distribution with variance σ^2. It is used when the data exhibits symmetric patterns around the mean, such as symmetry around the mean, a bell-shaped curve, and a single peak, and deviations from the mean are normally distributed. These patterns can be identified using descriptive statistics, histograms, Q-Q plots, and normality tests. This approach leverages the Maximum Likelihood principle to estimate the parameters of the distribution that maximize the likelihood of observing the given data. Specifically, it finds the mean and variance that best fit the observed data under the normal distribution assumption.

2. **Maximum Likelihood Approach using Laplace Distribution:** This method assumes that the error term ϵ follows a Laplace distribution with a scale parameter b.

The Laplace distribution is used for data with heavy tails and possible outliers. By selecting this distribution, we recognize the possibility of the data having a less symmetric distribution compared to the normal distribution. This approach is robust against outliers and often provides more accurate estimates, especially when the data exhibits significant variations.

3. **Bayesian Estimation with Gaussian Priors:** This method assumes that the error term ϵ follows a normal distribution with variance σ^2. Additionally, the matrix of parameters U is subject to identically distributed normal priors with a mean of 0 and variance σ_U^2. Bayesian inference introduces prior knowledge into the estimation process. By employing Gaussian priors, we express our belief that the true values of matrix U are likely to be centered around specific mean values. This approach is used when we have insights or expectations regarding typical user and item preferences. For example, in a recommendation system for movies, if we have prior knowledge that certain genres are generally rated higher by a user group, this information can be encoded in the priors for the latent factors, thereby influencing the final recommendations.

Below, we provide a detailed description of each estimation method for the matrix U using our proposed SPMF technique.

3.2.1 Maximum Likelihood Approach using Normal Distribution

To estimate the parameters of the low-rank matrix approximation for the block matrix $\mathcal{S}$ using our proposed SPMF method, we employ Maximum Likelihood Estimation (MLE). MLE is a statistical technique that seeks to determine the parameter values that maximize the likelihood of observing the given data.

Indeed, the objective of our proposed SPMF is to approximate the symmetric matrix $\mathcal{S}$ by decomposing it into the product of two low-dimensional matrices, U and U^T. This approximation incorporates a noise term ϵ to account for the differences between the true matrix and the low-rank approximation, as stated in Equation (3.4). The noise term ϵ captures the discrepancy between the actual matrix $\mathcal{S}$ and the low-rank approximation UU^T. In this context, an assumption is made that the noise term ϵ follows a Gaussian distribution with a mean of zero and a constant variance of σ^2. This assumption can be expressed as:

$$\mathfrak{s}_{ij} = (UU^T)_{ij} + \epsilon_{ij}, \tag{3.6}$$

where ϵ_{ij} are independently and identically distributed according to a normal distribution with parameters $(0, \sigma^2)$, and $\mathfrak{s}_{ij}$ represents an entry of the matrix $\mathcal{S}$.

Lemma 1. *In terms of probability distribution, it is sufficient to focus on either the lower or upper triangular part of the symmetric matrix $\mathcal{S}$ to enhance computational efficiency while preserving all relevant information. Without loss of generality, we will focus on the lower triangular part.*

Proof. As mentioned before, $\mathcal{S}$ is a symmetric matrix of dimension $(n+m) \times (n+m)$. By definition, $\mathfrak{s}_{ij} = \mathfrak{s}_{ji}$ for all i, j. Thus, the information contained in the lower triangular part (including the diagonal) is identical to that in the upper triangular part.

For instance, $\mathcal{S}$ can be represented as:

$$\mathcal{S} = \begin{bmatrix} \mathfrak{s}_{11} & \mathfrak{s}_{12} & \mathfrak{s}_{13} & \cdots & \mathfrak{s}_{1,n+m} \\ \mathfrak{s}_{21} & \mathfrak{s}_{22} & \mathfrak{s}_{23} & \cdots & \mathfrak{s}_{2,n+m} \\ \mathfrak{s}_{31} & \mathfrak{s}_{32} & \mathfrak{s}_{33} & \cdots & \mathfrak{s}_{3,n+m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathfrak{s}_{n+m,1} & \mathfrak{s}_{n+m,2} & \mathfrak{s}_{n+m,3} & \cdots & \mathfrak{s}_{n+m,n+m} \end{bmatrix}$$

To focus on the lower triangular part, we consider only the entries $\mathfrak{s}_{ij}$ for $i \geq j$:

$$\mathcal{S}_{\text{lower}} = \begin{bmatrix} \mathfrak{s}_{11} & 0 & 0 & \cdots & 0 \\ \mathfrak{s}_{21} & \mathfrak{s}_{22} & 0 & \cdots & 0 \\ \mathfrak{s}_{31} & \mathfrak{s}_{32} & \mathfrak{s}_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathfrak{s}_{n+m,1} & \mathfrak{s}_{n+m,2} & \mathfrak{s}_{n+m,3} & \cdots & \mathfrak{s}_{n+m,n+m} \end{bmatrix}$$

By focusing on the lower triangular part of $\mathcal{S}$, we avoid redundant calculations, thereby improving computational efficiency. Specifically, when evaluating $\mathcal{S}$, we limit our operations to the entries $\mathfrak{s}_{ij}$ for $i \geq j$, halving the number of computations required.

For a matrix of size $(n+m) \times (n+m)$, there are $\frac{(n+m)(n+m+1)}{2}$ unique entries in the lower triangular part. This is in contrast to the $(n+m)^2$ entries in the full matrix. Consequently, this approach reduces the computational complexity from $O((n+m)^2)$ to $O\left(\frac{(n+m)^2}{2}\right)$, thereby enhancing efficiency while retaining all necessary information for further processing. Thus, focusing on the lower triangular part of $\mathcal{S}$ is a strategy to improve computational efficiency without loss of information. $\square$

Method 1: (Maximum Likelihood Estimation for the Parameter Matrix U under Gaussian Noise Distribution) Given the observed data $\mathcal{S}$ and the noise variance σ^2, the goal is to find the parameter matrix U that maximizes the likelihood of the observed data under our proposed SPMF model, assuming a Gaussian noise distribution.

Our objective is to estimate the parameters of U that best align with the entries in the matrix $\mathcal{S}$ under our proposed SPMF model. From Equation (3.6), we can see that for the entry $\mathfrak{s}_{ij}$ in $\mathcal{S}$:

$$\mathfrak{s}_{ij} \sim \mathcal{N}((UU^T)_{ij}, \sigma^2). \tag{3.7}$$

So the probability density function of observing $\mathfrak{s}_{ij}$ given $(UU^T)_{ij}$ and σ^2 is:

$$p(\mathfrak{s}_{ij}|(UU^T)_{ij}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathfrak{s}_{ij} - (UU^T)_{ij})^2}{2\sigma^2}\right). \tag{3.8}$$

Considering the Lemma 1, our goal is to maximize the likelihood of observing the lower triangular and diagonal elements of $\mathcal{S}$ given the parameters U. Therefore, the likelihood function is expressed as:

$$L(U|\mathcal{S}) = \prod_{i=1}^{n+m}\left(\prod_{j=1}^{i} p(\mathfrak{s}_{ij}|(UU^T)_{ij}, \sigma^2)\right) = \prod_{i=1}^{n+m}\left(\prod_{j=1}^{i}\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathfrak{s}_{ij} - (UU^T)_{ij})^2}{2\sigma^2}\right)\right)\right). \tag{3.9}$$

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood. Taking the negative log-likelihood, we get:

$$\mathcal{L}(U) := -\log L(U|\mathcal{S}) = \frac{(n+m)(n+m+1)}{4}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n+m}\sum_{j=1}^{i}(\mathfrak{s}_{ij} - (UU^T)_{ij})^2. \tag{3.10}$$

We are currently aiming to determine the derivative of the negative log-likelihood with respect to U. We employ the following lemma to ascertain the derivative of UU^T with respect to U for any given entry u_{ij}.

Lemma 2. *For any entry* $(UU^T)_{pq}$, *where* $1 \leq p, q \leq n+m$, *The derivative of* $(UU^T)_{pq}$ *with respect to* u_{ij} *is:*

$$\frac{\partial}{\partial u_{ij}}(UU^T)_{pq} = u_{qj}\delta_{pi} + u_{pj}\delta_{qi}, \tag{3.11}$$

where δ_{ij} represents the Kronecker delta, which equals 1 when $i = j$ and 0 otherwise.

Proof. To demonstrate this result, we start by considering the derivative of an element $(UU^T)_{pq}$, where $1 \leq p, q \leq n+m$. This element can be represented as the sum of products:

$$(UU^T)_{pq} = \sum_{l=1}^{k} u_{pl}u_{ql}.$$

Therefore, the derivative of $(UU^T)_{pq}$ with respect to u_{ij} is:

$$\frac{\partial}{\partial u_{ij}}(UU^T)_{pq} = \frac{\partial}{\partial u_{ij}}\sum_{l=1}^{k} u_{pl}u_{ql} = \begin{cases} \frac{\partial}{\partial u_{ij}}(u_{ij}u_{qj}) = u_{qj} & \text{if } i = p,\ p \neq q \\ \frac{\partial}{\partial u_{ij}}(u_{ij}u_{pj}) = u_{pj} & \text{if } i = q,\ p \neq q \\ \frac{\partial}{\partial u_{ij}}(u_{pp}u_{pp}) = 2u_{pp} & \text{if } i = j = p \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Since the Kronecker delta δ_{lj} ensures that the terms are zero unless $l = j$, we can simplify further:

$$\frac{\partial}{\partial u_{ij}}(UU^T)_{pq} = u_{pj}\delta_{qi} + u_{qj}\delta_{pi}. \tag{3.13}$$

And this concludes the proof of the lemma. □

We can now calculate the derivative of the negative log-likelihood with respect to U for any element u_{ij} of U by utilizing the equation (3.11), as shown below:

$$\begin{aligned}
\frac{\partial \mathcal{L}(U)}{\partial u_{ij}} &= \frac{\partial}{\partial u_{ij}} \left[-\log L(U|\mathcal{S}) \right] \\
&= \frac{\partial}{\partial u_{ij}} \left[\frac{1}{2\sigma^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} (\mathfrak{s}_{pq} - (UU^T)_{pq})^2 \right] \\
&= \sum_{p=1}^{n+m} \sum_{q=1}^{p} \frac{\partial}{\partial u_{ij}} \left[\frac{1}{2\sigma^2} (\mathfrak{s}_{pq} - (UU^T)_{pq})^2 \right] \\
&= -\sum_{p=1}^{n+m} \sum_{q=1}^{p} \left[\frac{2}{2\sigma^2} (\mathfrak{s}_{pq} - (UU^T)_{pq}) \frac{\partial}{\partial u_{ij}} (UU^T)_{pq} \right] \\
&= -\frac{1}{\sigma^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} \left[(\mathfrak{s}_{pq} - (UU^T)_{pq}) \frac{\partial}{\partial u_{ij}} (UU^T)_{pq} \right] \\
&= -\frac{1}{\sigma^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} \left[(\mathfrak{s}_{pq} - (UU^T)_{pq}) [u_{qj}\delta_{pi} + u_{pj}\delta_{qi}] \right].
\end{aligned} \tag{3.14}$$

We can use gradient descent to iteratively update the parameters U as follows:

$$U_{t+1} = U_t - \alpha \frac{\partial \mathcal{L}(U_t)}{\partial U_t}, \tag{3.15}$$

where U_t denotes the value of U at iteration t and α is the learning rate. In the equation above, $\frac{\partial \mathcal{L}(U_t)}{\partial U_t}$ represents the derivative of the negative log-likelihood with respect to the matrix of parameters U. By subtracting this derivative from the current values of U scaled by the learning rate α, we can update the values of U in the direction that decreases the negative log-likelihood the most. This is known as the gradient descent algorithm. The learning rate α is a hyperparameter that determines the step size taken in each iteration of gradient descent. Selecting an appropriate learning rate is an important step in the training process, and we can determine the suitable learning rate through cross-validation.

The process stops when a predefined convergence criterion is met. In the context of gradient descent, this typically involves monitoring the change in the objective function or the gradient. The iterations continue until the change in these values becomes sufficiently small, indicating that the algorithm has reached a minimum or a point where further iterations are not significantly improving the results. Once we have learned the values of U, we can use them to predict the similarities between the users and items by computing UU^T.

The primary objective is to iteratively assess the SPMF model's performance by segmenting the observed data into training and validation sets. This process involves partitioning the observed matrix $\mathcal{S}$ using a "speckled" holdout pattern. A binary mask matrix M is created, with entries randomly set to 0 (validation points) or 1 (training points). This ensures no row or column is fully excluded, allowing all parameters to be estimated. The approach for cross-validation was inspired by Alex Williams' method, detailed in his blog post[1].

For instance, if the objective function is $\|\mathcal{S} - UU^T\|_F^2$, the mask matrix M allows us to reformulate the optimization problem as:

$$\min_U \|M \odot (\mathcal{S} - UU^T)\|_F^2$$

where $\odot$ denotes the Hadamard product (element-wise multiplication). After solving this optimization problem, we evaluate the model on the held-out data points using:

$$\text{RMSE} = \sqrt{\frac{1}{|\text{Held-out}|} \sum_{(i,j)\in\text{Held-out}} (\mathfrak{s}_{ij} - (UU^T)_{ij})^2}.$$

This speckled holdout pattern ensures balanced partitioning, prevents overfitting, and improves the generalization of the model.

To better understand how the speckled cross-validation algorithm works, we create a toy example. Please note that all matrices used here are artificial and the goal is solely to elucidate the speckled algorithm.

Speckled Cross-Validation for SPMF: A Toy Example

Cross-validation is a crucial method for evaluating and improving the performance of matrix factorization models. This example illustrates the process using a symmetric matrix $\mathcal{S}$, a mask matrix M, and an initialized factor matrix U. To begin, consider the mask matrix M, where 80% of the entries are 1 (used for training) and 20% are 0 (used for validation):

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

[1]The approach for cross-validation was inspired by the method outlined in a blog post by Alex Williams. `https://alexhwilliams.info/itsneuronalblog/2018/02/26/crossval/`

This random mask matrix M is used to specify which elements of the symmetric matrix $\mathcal{S}$ are used for training and which are held out for validation. Let's the symmetric matrix $\mathcal{S}$ is defined as follows:

$$\mathcal{S} = \begin{bmatrix} 1 & 0.5 & 0.3 & 0.7 & 0.2 \\ 0.5 & 1 & 0.6 & 0.8 & 0.4 \\ 0.3 & 0.6 & 1 & 0.9 & 0.5 \\ 0.7 & 0.8 & 0.9 & 1 & 0.6 \\ 0.2 & 0.4 & 0.5 & 0.6 & 1 \end{bmatrix}.$$

Next, we initialize the factor matrix U with random values, and set the number of latent factors $k = 2$:

$$U = \begin{bmatrix} 0.5 & 0.8 \\ 0.9 & 0.4 \\ 0.3 & 0.2 \\ 0.4 & 0.5 \\ 0.7 & 0.6 \end{bmatrix}.$$

Using this U, we compute the product UU^T:

$$UU^T = \begin{bmatrix} 0.89 & 0.77 & 0.31 & 0.6 & 0.83 \\ 0.77 & 0.97 & 0.35 & 0.56 & 0.87 \\ 0.31 & 0.35 & 0.13 & 0.22 & 0.33 \\ 0.6 & 0.56 & 0.22 & 0.41 & 0.58 \\ 0.83 & 0.87 & 0.33 & 0.58 & 0.85 \end{bmatrix}.$$

For the training set, we optimize the objective function by minimizing the difference between the observed ratings and the predicted ratings, using only the entries specified by the mask matrix M:

$$\min_U \|M \odot (\mathcal{S} - UU^T)\|_F^2.$$

Here, $\odot$ denotes the Hadamard product. The training set elements are:

$$M \odot (\mathcal{S} - UU^T) = \begin{bmatrix} 1 - 0.89 & 0 & 0.3 - 0.31 & 0.7 - 0.6 & 0.2 - 0.83 \\ 0.5 - 0.77 & 1 - 0.97 & 0 & 0.8 - 0.56 & 0.4 - 0.87 \\ 0 & 0.6 - 0.35 & 1 - 0.13 & 0.9 - 0.22 & 0.5 - 0.33 \\ 0.7 - 0.6 & 0.8 - 0.56 & 0.9 - 0.22 & 1 - 0.41 & 0 \\ 0.2 - 0.83 & 0 & 0.5 - 0.33 & 0.6 - 0.58 & 1 - 0.85 \end{bmatrix}.$$

To find the optimal U, we use an iterative approach to minimize the Frobenius norm of the above matrix. After training, we evaluate the model on the held-out data by calculating the

RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{|\text{Held-out}|} \sum_{(i,j)\in\text{Held-out}} (\mathfrak{s}_{ij} - (UU^T)_{ij})^2}.$$

The validation set elements are:

$$(1 - M) \odot (\mathcal{S} - UU^T) = \begin{bmatrix} 0 & 0.5 - 0.77 & 0 & 0 & 0 \\ 0 & 0 & 0.6 - 0.35 & 0 & 0 \\ 0.3 - 0.31 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6 - 0.58 \\ 0 & 0.4 - 0.87 & 0 & 0 & 0 \end{bmatrix}.$$

Therefore, the sum of squared differences is:

$$(0.5 - 0.77)^2 + (0.6 - 0.35)^2 + (0.3 - 0.31)^2 + (0.6 - 0.58)^2 + (0.4 - 0.87)^2 = 0.3568.$$

Finally,

$$\text{RMSE} = \sqrt{\frac{0.3568}{5}} = 0.267.$$

This is just a toy example to demonstrate the process. A complete example using real-world data will be discussed in Chapter 4.

In the context of K-fold cross-validation, we apply the speckled cross-validation approach by constructing the binary mask matrix M. This involves dividing the lower triangular part and diagonal of the matrix $\mathcal{S}$ (based on Lemma 1) randomly into K approximately equal-sized folds. Each entry in M, aligned with the matrix $\mathcal{S}$, indicates whether the corresponding element in $\mathcal{S}$ belongs to the training set (represented as 1) or the validation set (represented as 0).

The process of creating M for K-fold cross-validation involves:

- Randomly dividing the lower triangular part and diagonal matrix $\mathcal{S}$ into K folds.
- Setting the corresponding elements in M to 0 (indicating validation) for each fold and keeping the rest as 1 (indicating training).
- Rotating which fold is used for validation in each iteration until all folds have been used as validation once.

The algorithm 2 describes the MLE approach using normal distributions for estimating the matrix U in a recommendation system. Here are the key parts of the algorithm explained briefly:

- Data Preprocessing:
  - Replace missing values with 0 in the user-item matrix R.
  - Calculate the normalized matrices R_{RN} and R_{CN} from R.
- Block Matrix Calculation:
  - Compute the block matrix $\mathcal{S}$.
- Cross-Validation:
  - Employ K-fold cross-validation by randomly partitioning the elements of the lower triangular part and diagonal matrix $\mathcal{S}$ into K folds. A detailed explanation of this will be provided in 4.3.
- Training and Validation:
  - Designate $\mathcal{S}_{train}$ for training and $\mathcal{S}_{val}$ for validation purposes.
  - Explore various combinations of latent factors k and learning rates α.
  - Initialize matrix U with random values and fine-tune it through multiple iterations using gradient descent.
  - Keep track of the validation loss, calculated as the RMSE between the $\mathcal{S}_{val}$ and the optimized U using the optimization approach detailed earlier.
- Determine the Best Latent Factor:
  - Find the latent factor k_{best} that minimizes the average validation loss across all folds.
- Optimal Learning Rate:
  - Identify the learning rate α^* that minimizes the validation loss for the best latent factor k_{best}.
- Training with Optimal Parameters:
  - Initialize U with random values using the optimal latent factor k_{best} and optimize U using the best learning rate α^*.

- Performance Evaluation:
  - Calculate the approximated $\hat{\mathcal{S}}$ using $\hat{U}$.
  - Compute the RMSE.
- Output:
  - Provide the estimated matrix $\hat{U}$, the best latent factor k_{best}, and the RMSE based on k_{best}.

These steps summarize the process of estimating the matrix U using maximum likelihood estimation with normal distribution and evaluating the model's performance through cross-validation and RMSE calculations.

Algorithm 2 Maximum Likelihood Approach using Normal Distribution with Cross-Validation

```
Require: User-item matrix $R$, a fixed $\sigma^2$, list of learning rates $\alpha$, number of iterations $T$, number of folds $F$
Ensure: Estimated matrix $U$, best number of latent factors $k_{\text{best}}$, RMSE based on $k_{\text{best}}$
 1: Replace missing value with 0 in $R$
 2: Calculate the normalized matrix $R_{RN}$ and $R_{CN}$
 3: Calculate the block matrix $\mathcal{S}$
 4: Randomly partitioning the elements of the lower triangular part and diagonal matrix $\mathcal{S}$ into $F$ folds
 5: Designate the data into training and validation sets: $\mathcal{S}_{train}$ and $\mathcal{S}_{val}$
 6: for $f = 1$ to $F$ do
 7:   for $k = 1$ to $K_{\max}$ do
 8:     Initialize a list of validation losses $L_{val}[f][k]$ for each potential latent factor $k$
 9:     for $\alpha$ in $\alpha$ do
10:       Initialize $U$ with random values
11:       for $t = 1$ to $T$ do
12:         Compute the gradient of the loss function with respect to $U$ using $\mathcal{S}_{train}$ and the equation (3.14)
13:         Update $U$ using gradient descent: $U \leftarrow U - \alpha\frac{\partial\mathcal{L}(U)}{\partial U}$
14:       end for
15:       Compute the validation loss using the original $\mathcal{S}_{val}$ : $L_{val}[f][k] \leftarrow L_{val}[f][k] + \frac{1}{F}L_{RMSE}(\mathcal{S}_{val}, U)$
16:     end for
17:   end for
18: end for
19: Find the best $k_{\text{best}}$ that minimizes the average validation loss across folds: $k_{\text{best}} = \text{argmin}_k\left(\frac{1}{F}\sum_{f=1}^{F} L_{val}[f][k]\right)$
20: Find the best learning rate $\alpha^*$ for $k_{\text{best}}$: $\alpha^* = \arg\min_\alpha L_{val}[f][k_{\text{best}}]$
21: Initialize $U$ with random values and the obtained latent factor $k_{\text{best}}$
22: for $t = 1$ to $T$ do
23:   Compute the gradient of the loss function with respect to $U$ using the equation (3.14)
24:   Update $U$ using the best learning rate: $U \leftarrow U - \alpha^*(\frac{\partial\mathcal{L}(U)}{\partial U})$
25: end for
26: Calculate the estimated $\hat{\mathcal{S}}$
27: Compute the RMSE
28: return $\hat{U}$, $k_{\text{best}}$, RMSE
```

3.2.2 Maximum Likelihood Approach using Laplace Distribution

In this section, we further explore the Maximum Likelihood Estimation (MLE) for the parameters of U in the matrix factorization approach applied to the block matrix $\mathcal{S}$ using our proposed SPMF method. This approach utilizes the Laplace distribution as an alternative to the normal distribution.

As stated in (3.4), we assume that the symmetric matrix $\mathcal{S}$ can be approximated by the product of two low-dimensional matrices, U and U^T, along with a noise term that follows a Laplace distribution:

$$\mathfrak{s}_{ij} = (UU^T)_{ij} + \epsilon_{ij}, \tag{3.16}$$

where ϵ_{ij} are independently and identically distributed according to Laplace distribution with parameters $(0, b)$, and $\mathfrak{s}_{ij}$ represents an entry of the matrix $\mathcal{S}$.

Method 2: (Maximum Likelihood Estimation for the matrix U - Laplace Noise Distribution) Given the observed data $\mathcal{S}$ and the scale parameter b, the objective is to find the parameter matrix U that maximizes the likelihood of the observed data under our proposed SPMF model, assuming a Laplace noise distribution.

To acquire the parameters within matrix U, we aim to maximize the likelihood of the observed entries in $\mathcal{S}$. For each entry $\mathfrak{s}_{ij}$ of $\mathcal{S}$, we have:

$$\mathfrak{s}_{ij} \sim \text{Laplace}((UU^T)_{ij}, b). \tag{3.17}$$

The probability density function of a Laplacian distribution with location parameter μ and scale parameter b is given by:

$$p(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right). \tag{3.18}$$

Thus, the probability density function of $\mathfrak{s}_{ij}$ given $(UU^T)_{ij}$ and b is:

$$p(\mathfrak{s}_{ij}|(UU^T)_{ij}, b) = \frac{1}{2b} \exp\left(-\frac{|\mathfrak{s}_{ij} - (UU^T)_{ij}|}{b}\right). \tag{3.19}$$

Lemma 3. *For the Laplace distribution with the scale parameter b, the derivative of the* *log-likelihood with respect to U is given by:*

$$\frac{\partial}{\partial U}\log L(U|\mathcal{S}) = -\frac{1}{b}\sum_{i=1}^{n+m}\sum_{j=1}^{i} sgn(\mathfrak{s}_{ij} - (UU^T)_{ij})\frac{\partial}{\partial U}(-(UU^T)_{ij}) \tag{3.20}$$

Proof. Referring to the Lemma 1, our goal is to maximize the likelihood of observing the lower triangular and diagonal elements of $\mathcal{S}$ given the parameters U. Therefore, the likelihood function is expressed as:

$$L(U|\mathcal{S}) = \prod_{i=1}^{n+m}\prod_{j=1}^{i} p(\mathfrak{s}_{ij}|(UU^T)_{ij}, b) = \prod_{i=1}^{n+m}\prod_{j=1}^{i}\frac{1}{2b}\exp\left(-\frac{|\mathfrak{s}_{ij} - (UU^T)_{ij}|}{b}\right). \tag{3.21}$$

To find the values of U that maximize the likelihood function, we can take the logarithm of the function and maximize it instead. Therefore, the log-likelihood function can be written as:

$$\log L(U|\mathcal{S}) = \sum_{i=1}^{n+m}\sum_{j=1}^{i}\log\left(\frac{1}{2b}\exp\left(-\frac{|\mathfrak{s}_{ij} - (UU^T)_{ij}|}{b}\right)\right). \tag{3.22}$$

We can simplify this expression further:

$$\log L(U|\mathcal{S}) = -\frac{1}{b}\sum_{i=1}^{n+m}\sum_{j=1}^{i}|\mathfrak{s}_{ij} - (UU^T)_{ij}| + \text{constant}. \tag{3.23}$$

Here, the constant term does not affect the optimization problem and can be ignored. Now, let's compute the gradient with respect to the elements of U:

$$\begin{aligned}\frac{\partial}{\partial U}\log L(U|\mathcal{S}) &= \frac{\partial}{\partial U}\left(-\frac{1}{b}\sum_{i=1}^{n+m}\sum_{j=1}^{i}|\mathfrak{s}_{ij} - (UU^T)_{ij}| + \text{constant}\right)\\ &= -\frac{1}{b}\frac{\partial}{\partial U}\left(\sum_{i=1}^{n+m}\sum_{j=1}^{i}|\mathfrak{s}_{ij} - (UU^T)_{ij}|\right).\end{aligned}$$

The gradient with respect to the elements of U can be computed by considering the subgradients of the absolute value function. Let's denote the subgradient of $|x|$ as $\text{sgn}(x)$, which takes the value -1, 0, or 1 depending on the sign of x. Finally, we can express the gradient as follows:

$$\frac{\partial}{\partial U}\log L(U|\mathcal{S}) = -\frac{1}{b}\sum_{i=1}^{n+m}\sum_{j=1}^{i}\text{sgn}(\mathfrak{s}_{ij} - (UU^T)_{ij})\frac{\partial}{\partial U}(\mathfrak{s}_{ij} - (UU^T)_{ij}), \tag{3.24}$$

where$\frac{\partial}{\partial U}(\mathfrak{s}_{ij} - (UU^T)_{ij}) = \frac{\partial}{\partial U}(-(UU^T)_{ij})$ can be computed in a similar way as we described in (3.11). □

To maximize the log-likelihood function, we can apply gradient descent. In gradient descent, we update the values of U iteratively in the opposite direction of the gradient:

$$U_{t+1} = U_t - \alpha \frac{\partial}{\partial U_t} \log L(U_t|\mathcal{S}),$$

where U_t denotes the value of U at iteration t and α is the learning rate that controls the step size in each iteration.

By iteratively updating the values of U according to the gradient descent update rules, we can gradually maximize the log-likelihood function and find the optimal values of U that maximize it. The algorithm for this process is described in Algorithm 3.

Algorithm 3 Maximum Likelihood Approach using Laplace Distributions

```
Require: User-item matrix $R$, a fix $b$, list of learning rates $\alpha$, number of iterations $T$, number of folds $F$
Ensure: Estimated matrix $U$, best number of latent factors $k_{\text{best}}$, RMSE based on $k_{\text{best}}$
 1: Replace missing value with 0 in $R$
 2: Calculate the normalized matrix $R_{RN}$ and $R_{CN}$
 3: Calculate the block matrix $\mathcal{S}$
 4: Randomly partitioning the elements of the lower triangular part and diagonal matrix $\mathcal{S}$ into $F$ folds
 5: Designate the data into training and validation sets: $\mathcal{S}_{train}$ and $\mathcal{S}_{val}$
 6: for $f = 1$ to $F$ do
 7:     for $k = 1$ to $K_{\max}$ do
 8:         Initialize a list of validation losses $L_{val}[f][k]$ for each potential latent factor $k$
 9:         for $\alpha$ in $\alpha$ do
10:             Initialize $U$ with random values
11:             for $t = 1$ to $T$ do
12:                 Compute the gradient of the loss function with respect to $U$ using $\mathcal{S}_{train}$ and the equation (3.20)
13:                 Update $U$ using gradient descent
14:             end for
15:             Compute the validation loss using the original $\mathcal{S}_{val}$ : $L_{val}[f][k] \leftarrow L_{val}[f][k] + \frac{1}{F} L_{RMSE}(\mathcal{S}_{val}, U)$
16:         end for
17:     end for
18: end for
19: Find the best $k_{\text{best}}$ that minimizes the average validation loss across folds: $k_{\text{best}} = \text{argmin}_k \left(\frac{1}{F}\sum_{f=1}^{F} L_{val}[f][k]\right)$
20: Find the best learning rate $\alpha^*$ for $k_{\text{best}}$: $\alpha^* = \arg\min_\alpha L_{val}[f][k_{\text{best}}]$
21: Initialize $U$ with random values and the obtained latent factor $k_{\text{best}}$
22: for $t = 1$ to $T$ do
23:     Compute the gradient of the loss function with respect to $U$ using the equation (3.20)
24:     Update $U$ using the best learning rate
25: end for
26: Calculate the estimated $\hat{\mathcal{S}}$
27: Compute the RMSE
28: return $\hat{U}$, $k_{\text{best}}$, RMSE
```

3.2.3 Bayesian Estimation with Gaussian Priors

In this section, we leverage a Bayesian framework to estimate the matrix U using our proposed SPMF method by incorporating prior knowledge about the data. Establishing a prior distribution over U enables us to estimate user preferences within this informed framework.

To initiate the Bayesian estimation, one approach is to presume that the entries of U follow an identical Gaussian distribution with a mean of zero and a variance of σ_U^2 [85]:

$$p(U) = \prod_{p=1}^{n+m} \prod_{q=1}^{k} \mathcal{N}(u_{pq} \mid 0, \sigma_U^2).$$

Here, $p(U)$ represents the prior probability distribution governing U, allowing integration of our prior knowledge or assumptions about U into the model. Specifically, within this context, we adopt a Gaussian prior distribution for the individual elements of U. This prior assumes that entries u_{pq} of the matrix U are independent and identically distributed according to a Gaussian distribution with a mean of zero and a variance of σ_U^2. The notation $\mathcal{N}(u_{pq} \mid 0, \sigma_U^2)$ signifies the Gaussian probability density function characterizing each element u_{pq}, featuring a mean of zero and a variance of σ_U^2.

Next, referring to the Equation (3.4), we need to specify a likelihood function for the matrix data $\mathcal{S}$ given the parameters U and the noise variance σ^2, similar to what we described in Equation (3.9).

Method 3: (Estimating the Matrix U through Bayesian Inference) Given the observed data $\mathcal{S}$ and parameters σ_U^2 and σ^2, our goal is to estimate the matrix U using Bayesian inference.

Using Bayes' theorem, the posterior distribution of U given the data $\mathcal{S}$ and the parameters σ_U^2 and σ^2 is proportional to the product of the prior and the likelihood:

$$p(U|\mathcal{S}, \sigma_U^2, \sigma^2) \propto p(U)p(\mathcal{S}|U, \sigma^2). \tag{3.25}$$

Referring to the Lemma 1, our goal is to maximize the likelihood $p(U|\mathcal{S}, \sigma_U^2, \sigma^2)$, which involves observing the lower triangular and diagonal elements of $\mathcal{S}$ given the parameters U.

This is equivalent to minimizing the negative logarithm of the posterior distribution:

$$\begin{aligned}\mathcal{L}(U) &= -\log p(U|\mathcal{S}, \sigma_U^2, \sigma^2) = -\log p(U) - \log p(\mathcal{S}|U, \sigma^2) + \text{constant} \\ &= \frac{||U||_F^2}{2\sigma_U^2} + \frac{1}{2\sigma^2} \sum_{i=1}^{n+m} \sum_{j=1}^{i} (\mathfrak{s}_{ij} - (UU^T)_{ij})^2\end{aligned} \tag{3.26}$$

where the constant term does not depend on U and can be ignored. To minimize the loss function $\mathcal{L}(U)$ using gradient descent, we first need to compute its derivative with respect to U:

$$\frac{\partial \mathcal{L}(U)}{\partial U} = \frac{\partial}{\partial U}\left(\frac{||U||_F^2}{2\sigma_U^2} + \frac{1}{2\sigma^2} \sum_{i=1}^{n+m} \sum_{j=1}^{i} (\mathfrak{s}_{ij} - (UU^T)_{ij})^2\right). \tag{3.27}$$

To do that we calculate the derivative with respect to any entry u_{ij} of the matrix U.

Lemma 4. *For any entry u_{ij} of U:*

$$\frac{\partial \mathcal{L}(U)}{\partial u_{ij}} = -\frac{1}{\sigma^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} \Big[(\mathfrak{s}_{pq} - (UU^T)_{pq})[u_{qj}\delta_{pi} + u_{pj}\delta_{qi}]\Big] + \frac{1}{\sigma_U^2} u_{ij}. \tag{3.28}$$

Proof. As we computed in Equation (3.14):

$$\frac{\partial}{\partial u_{ij}}\left(\frac{1}{2\sigma^2} \sum_{i=1}^{n+m} \sum_{j=1}^{i} (\mathfrak{s}_{ij} - (UU^T)_{ij})^2\right) = -\frac{1}{\sigma^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} \Big[(\mathfrak{s}_{pq} - (UU^T)_{pq})[u_{qj}\delta_{pi} + u_{pj}\delta_{qi}]\Big]. \tag{3.29}$$

Next, we calculate the derivative of the expression $\frac{||U||_F^2}{2\sigma_U^2}$ with respect to the matrix U for each entry u_{ij}. Starting with the Frobenius norm squared expression:

$$||U||_F^2 = \sum_{p=1}^{n+m} \sum_{q=1}^{p} u_{pq}^2. \tag{3.30}$$

The derivative of $\frac{||U||_F^2}{2\sigma_U^2}$ with respect to u_{ij} is:

$$\frac{\partial}{\partial u_{ij}}\left(\frac{||U||_F^2}{2\sigma_U^2}\right) = \frac{\partial}{\partial u_{ij}}\left(\frac{1}{2\sigma_U^2} \sum_{p=1}^{n+m} \sum_{q=1}^{p} u_{pq}^2\right) = \frac{1}{\sigma_U^2} u_{ij}. \tag{3.31}$$

By combining equations (3.29) and (3.31), the desired result is obtained.

□

Now, we can utilize the gradient calculated in the previous lemma to iteratively update the parameters of U using the gradient descent algorithm:

$$U_{t+1} = U_t - \alpha \frac{\partial \mathcal{L}(U_t)}{\partial U_t}, \tag{3.32}$$

where U_t denotes the value of U at iteration t, α is the learning rate, and $\frac{\partial \mathcal{L}(U_t)}{\partial U_t}$ is the gradient of the loss function with respect to U evaluated at U_t. The gradient descent algorithm works by iteratively updating the parameters U in the direction of steepest descent of the loss function until convergence. The learning rate controls the size of the steps taken in each iteration and should be chosen carefully to ensure convergence. The complete algorithm for approximating the block matrix $\mathcal{S}$ using Bayesian inference is described in Algorithm 4.

Algorithm 4 SPMF using Bayesian Inference with Cross-Validation

```
Require: User-item matrix $R$, a fix $\sigma^2$, a fix $\sigma_U^2$, list of learning rates $\alpha$, number of iterations $T$, number of folds $F$
Ensure: Estimated matrix $U$, best number of latent factors $k_{\text{best}}$, RMSE based on $k_{\text{best}}$
 1: Replace missing value with 0 in $R$
 2: Calculate the normalized matrix $R_{RN}$ and $R_{CN}$
 3: Calculate the block matrix $\mathcal{S}$
 4: Randomly partitioning the elements of the lower triangular part and diagonal matrix $\mathcal{S}$ into $F$ folds
 5: Designate the data into training and validation sets: $\mathcal{S}_{train}$ and $\mathcal{S}_{val}$
 6: for $f = 1$ to $F$ do
 7:   for $k = 1$ to $K_{\max}$ do
 8:     Initialize a list of validation losses $L_{val}[f][k]$ for each potential latent factor $k$
 9:     for $\alpha$ in $\alpha$ do
10:       Initialize $U$ with random values
11:       for $t = 1$ to $T$ do
12:         Compute the gradient of the loss function with respect to $U$ using $\mathcal{S}_{train}$ and the equation (3.28)
13:         Update $U$ using gradient descent: $U \leftarrow U - \alpha \frac{\partial \mathcal{L}(U)}{\partial U}$
14:       end for
15:       Compute the validation loss using the original $\mathcal{S}_{val}$ : $L_{val}[f][k] \leftarrow L_{val}[f][k] + \frac{1}{F} L_{RMSE}(\mathcal{S}_{val}, U)$
16:     end for
17:   end for
18: end for
19: Find the best $k_{\text{best}}$ that minimizes the average validation loss across folds: $k_{\text{best}} = \text{argmin}_k \left( \frac{1}{F} \sum_{f=1}^{F} L_{val}[f][k] \right)$
20: Find the best learning rate $\alpha^*$ for $k_{\text{best}}$: $\alpha^* = \arg\min_\alpha L_{val}[f][k_{\text{best}}]$
21: Initialize $U$ with random values and the obtained latent factor $k_{\text{best}}$
22: for $t = 1$ to $T$ do
23:   Compute the gradient of the loss function with respect to $U$ using the equation (3.28)
24:   Update $U$ using the best learning rate: $U \leftarrow U - \alpha^* (\frac{\partial \mathcal{L}(U)}{\partial U})$
25: end for
26: Calculate the estimated $\hat{\mathcal{S}}$
27: Compute the RMSE based
28: return $\hat{U}$, $k_{\text{best}}$, RMSE
```

Following our explanation of the block matrix estimation process, we illustrate its practical application through a toy example that demonstrates the real-world utility of SPMF.

Application of SPMF: A Practical Toy Example

Below, we provide a numerical example illustrating how ratings are predicted using SPMF. This example elucidates how we derive insights from the approximated block matrix $\mathcal{S}$. Among the proposed SPMF methods—MLE with a normal noise distribution, MLE with a Laplace noise distribution, and Bayesian estimation with a normal distribution for the error term and a normal distribution as a prior for the entries of U—we opt for Bayesian estimation. It is important to note that the block matrix $\mathcal{S}$ can also be approximated using the other SPMF methods mentioned above.

We will factorize the same matrix used for SVD, Basic Matrix Factorization, and NMF and PMF, but this time using SPMF. The user-movie rating matrix R is shown below:

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \left(\begin{array}{ccc} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{array}\right) \end{array}.$$

Our objective is to demonstrate the relationships between users and items by estimating the matrix U and predicting the missing ratings within the provided user-item rating matrix R. Specifically, we aim to predict Alice's rating for "Inception" movie and Dave's rating for "Matrix" movie. These predictions leverage the latent factors obtained from the SPMF process.

Upon deriving these interpretations and predictions, we delve deeper into understanding the dynamics that underpin the relationships between movies, users, and their preferences. This exploration includes examining:

- **Movie-Movie Similarities**: Understanding how different movies relate to each other based on their latent factor representations.
- **User-User Similarities**: Analyzing the similarities between users in terms of their preferences and behaviors.
- **User-Movie Interactions**: Investigating the nuanced interactions between users and

movies, which influence their ratings and preferences.

Therefore, through this analysis, we aim to uncover the intricate patterns and dynamics that govern user behavior and item characteristics, thereby enhancing the interpretability and reliability of our proposed SPMF method.

First, we describe the methodology for deriving the optimal estimated matrix U by employing the Bayesian framework outlined in Algorithm 4.

Matrix U Estimation

The Bayesian estimation approach offers several advantages, including the incorporation of prior knowledge and the ability to quantify uncertainty in the predictions. To estimate the optimal matrix U using the proposed SPMF, we utilize a Bayesian inference algorithm with cross-validation, implemented in Python based on Algorithm 4. Below, we explain how this matrix is obtained using the SPMF method with Bayesian inference.

Given that there are just two genres, "Action" and "Romance," the maximum number of latent factors is $k_{\max} = 2$. The learning rates α are set to 0.1 and 0.01, and the number of iterations T is set to 30, 50, and 100. The main steps of this algorithm to estimate the optimal matrix U using the proposed SPMF, are outlined below:

1. **Initialization and Data Preparation**:
   - The user-item matrix R is prepared by replacing missing values with 0.
   - The normalized matrices R_{RN} and R_{CN} are calculated.
   - The block matrix $\mathcal{S}$ is computed from these normalized matrices.
   - The elements of the lower triangular part and diagonal of $\mathcal{S}$ are partitioned into 5 folds for cross-validation using the speckled method.
2. **Cross-Validation for Latent Factors and Learning Rate**:
   - For each fold and each potential number of latent factors $k = 1, 2$, the algorithm initializes a list of validation losses.
   - For each learning rate $\alpha = 0.1, 0.01$, the matrix U is initialized with random values.

- For each number of iterations $T = 30, 50$ and 100, the gradient of the SPMF objective function with Bayesian inference with respect to U is computed and used to update U using gradient descent.
- The validation loss is computed using the original validation set $\mathcal{S}_{val}$, and averaged across folds to identify the best number of latent factors k_{best} and the best learning rate α^*.

3. **Final Estimation of Matrix U:**

- With the identified $k_{\text{best}} = 2$ and $\alpha^* = 0.1$, the matrix U is re-initialized and updated over $T = 50$ iterations using gradient descent.
- The estimated matrix $\hat{\mathcal{S}}$ is computed, and the RMSE is calculated to evaluate the model's accuracy.

User Preferences and Latent Factors:

The estimated optimal matrix $\hat{U}$, obtained using Algorithm 4, is illustrated below.

$$\hat{U} = \begin{array}{c} \\ \text{Matrix} \\ \text{Titanic} \\ \text{Inception} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} 0.93 & 0.35 \\ 0.41 & 0.9 \\ 0.8 & 0.58 \\ 0.9 & 0.43 \\ 0.43 & 0.89 \\ 0.95 & 0.3 \\ 0.71 & 0.7 \end{pmatrix} \end{array}.$$

Interpreting the matrix $\hat{U}$ reveals insights into user preferences. The matrix $\hat{U}$ represents the latent factors for movies and users after the estimation process. Each row corresponds to a movie or user, and each column corresponds to a latent factor associated with a genre ("Action" and "Romance" in this case). The values in the matrix indicate the strength of the association between each movie/user and each genre. Below, we elaborate on how we can derive meaningful insights from the obtained matrix $\hat{U}$ and utilize these insights to enhance our understanding of user preferences and movie characteristics:

- **Movies relation with latent factors**
  - **Matrix:** Strongly associated with the "Action" genre (0.93) and moderately with "Romance" (0.35).
  - **Titanic:** Strongly associated with the "Romance" genre (0.9) and weakly with "Action" (0.41).
  - **Inception:** Moderately associated with both "Action" (0.8) and "Romance" (0.58).
- **User Preferences:**
  - **Alice:** Prefers "Action" (0.9) more than "Romance" (0.43).
  - **Bob:** Strongly prefers "Romance" (0.89) with a moderate association with "Action" (0.43).
  - **Carol:** Strong preference for "Action" (0.95) and low preference for "Romance" (0.3).
  - **Dave:** Balanced preference for both "Action" (0.71) and "Romance" (0.7).

By leveraging the latent factors in $\hat{U}$, we can not only enhance user experience by providing personalized recommendations but also drive strategic decisions in marketing and content creation to better meet the preferences of the audience.

Predicting Missing Ratings:

Based on the construction of the block matrix $\mathcal{S}$, the off-diagonal blocks serve to represent R_{RN} and R_{RN}^T, encapsulating the interactions between users and items. After obtaining the matrix $\hat{U}$ through the SPMF process and subsequently calculating the product $\hat{U}\hat{U}^T$, we are equipped to make predictions for missing ratings. In our specific scenario, we arrived at matrix $\hat{U}$ via factorization and computed $\hat{U}\hat{U}^T$ to yield the following representation:

$$\hat{U}\hat{U}^T = \begin{array}{c|ccccccc} & \text{Matrix} & \text{Titanic} & \text{Inception} & \text{Alice} & \text{Bob} & \text{Carol} & \text{Dave} \\ \hline \text{Matrix} & 0.99 & 0.70 & 0.96 & 0.99 & 0.71 & 0.99 & 0.91 \\ \text{Titanic} & 0.70 & 0.99 & 0.86 & 0.76 & 0.98 & 0.66 & 0.93 \\ \text{Inception} & 0.96 & 0.86 & 0.99 & 0.98 & 0.87 & 0.94 & 0.98 \\ \text{Alice} & 0.99 & 0.76 & 0.98 & 0.99 & 0.77 & 0.98 & 0.94 \\ \text{Bob} & 0.71 & 0.98 & 0.87 & 0.77 & 0.98 & 0.68 & 0.93 \\ \text{Carol} & 0.99 & 0.66 & 0.94 & 0.98 & 0.68 & 0.99 & 0.88 \\ \text{Dave} & 0.91 & 0.93 & 0.98 & 0.94 & 0.93 & 0.88 & 0.99 \end{array}$$

To better visualize the four blocks of the approximated matrix $\hat{\mathcal{S}} = \hat{U}\hat{U}^T$, we color-code it as shown below:

	Matrix	Titanic	Inception	Alice	Bob	Carol	Dave
Matrix	0.99	0.70	0.96	0.99	0.71	0.99	0.91
Titanic	0.70	0.99	0.86	0.76	0.98	0.66	0.93
Inception	0.96	0.86	0.99	0.98	0.87	0.94	0.98
Alice	0.99	0.76	0.98	0.99	0.77	0.98	0.94
Bob	0.71	0.98	0.87	0.77	0.98	0.68	0.93
Carol	0.99	0.66	0.94	0.98	0.68	0.99	0.88
Dave	0.91	0.93	0.98	0.94	0.93	0.88	0.99

Upon comparing the resulting $\hat{U}\hat{U}^T$ with the previously constructed $\mathcal{S}$ matrix, we discern that the yellow block corresponds to the approximated R_{RN}. As a result, we can deduce that the predicted rating in standardized form for the user Alice and the movie Inception stands at **0.98**. To convert the predicted rating into a range between 0 and 5, the following formula is used [1]:

$$\text{Predicted rating} = \text{predicted standardized rating} \times (\text{max rate in } R - \text{min rate in } R). \quad (3.33)$$

The Equation (3.33) scales the standardized prediction by the range of ratings in the original dataset. Consequently, this formula rescales the prediction to fit within the original range of ratings, specifically between 0 and 5.

Therefore, the predicted rating for the user Alice and the movie Inception is $0.98 \times (5 - 0) =$ **4.9**. This inference implies a resemblance between Alice's cinematic preferences and the observed similarity between The Matrix and Inception. Similarly, we project a rating of **0.91** for Emily and The Matrix, indicating that Dave's movie preferences align with the shared affinity between herself and The Inception, which will be **4.6**.

It is worth noting that we observe that $\hat{U}\hat{U}^T$ serves as an approximate for $\mathcal{S}$, and this relationship is reflected in the diagonal values of the matrix. These diagonal elements, which are not equal to 1, result from the estimation process and indicate some underestimation in item or user similarity.

Understanding Relationships:

Blue Block (Movie-Movie Similarity Matrix): The blue-colored area corresponds to the item-item similarity matrix. Each cell in this block indicates the similarity between two movies based on their shared characteristics. Higher values signify stronger similarity.

For instance:

- **Matrix and Titanic:** The similarity score of 0.70 suggests that "Matrix" and "Titanic" share some common latent factors, although the similarity is moderate.
- **Matrix and Inception:** A high similarity score of 0.96 indicates that "Matrix" and "Inception" have a strong association, likely sharing many action-related latent factors.
- **Titanic and Inception:** With a score of 0.86, "Titanic" and "Inception" have some significant shared attributes, possibly related to a blend of romance and action genres.

Red Block (User-User Similarity Matrix): The red-highlighted region corresponds to the user-user similarity matrix. Each cell represents the similarity between two users based on their movie preferences. Larger values indicate greater similarity.

For instance:

- **Alice and Bob:** The similarity score of 0.77 suggests that Alice and Bob share some common latent factors, although the similarity is moderate.
- **Alice and Carol:** A high similarity score of 0.98 indicates that Alice and Carol have a strong association, likely sharing many similar preferences in both action and romance genres.
- **Alice and Dave:** The score of 0.94 shows that Alice and Dave have a substantial similarity in their preferences, indicating shared attributes or characteristics.

- **Bob and Carol:** With a score of 0.68, Bob and Carol have some shared attributes, though less strong compared to others.
- **Bob and Dave:** The similarity score of 0.93 suggests that Bob and Dave share a high degree of common latent factors, indicating similar tastes.
- **Carol and Dave:** With a score of 0.88, Carol and Dave have a high level of similarity in their preferences.

This interpretation helps in understanding the relationships between users, which can be leveraged for making recommendations based on the similarity of user preferences.

Upon factorizing the user-item rating matrix R using SPMF, we present our contributions of introducing SPMF and compare it with other state-of-the-art models like SVD, NMF, and PMF.

Contributions of Proposed SPMF vs. Other RS Methods

After factorizing the user-item rating matrix R using SPMF with Bayesian inference in the toy example, we summarize the key contributions of our proposed SPMF method and highlight how it differentiates from traditional recommendation system techniques like SVD, NMF, and PMF. The following points elucidate the innovations and advantages of SPMF:

1. **Transformation from Biconvex to Convex Optimization**:
    - **Unified Matrix Interpretation**: By transforming the optimization problem from biconvex to convex, SPMF allows for the interpretation of a single matrix U rather than two matrices (U and V) as in traditional matrix factorization methods. This unified matrix U simultaneously represents both items and users, capturing their relations with latent factors more coherently.
    - **Simplified Model Structure**: The convex optimization framework simplifies the model structure, making it more robust and easier to analyze mathematically.
2. **Interpretability and Practical Insights**:
    - **Enhanced Interpretability**: The obtained matrix U is more interpretable, allowing for clear insights into user preferences and item characteristics. This interpretability is crucial for making actionable recommendations and strategic decisions based on the model's output.

- **Application of Bayesian Inference**: By incorporating Bayesian inference, SPMF quantifies the uncertainty in predictions, providing a probabilistic measure of confidence that is valuable for decision-making.

3. **Optimal Solution and Latent Factors**:

   - **Unique Optimal Solution**: SPMF guarantees the finding of a unique optimal solution U, providing stability and reliability in the recommendations generated.
   - **Determination of Latent Factors**: The method allows for the identification of the optimal number of latent factors k, ensuring that the model maximizes its predictive power.

4. **Simultaneous Prediction and Similarity Calculation**:

   - **Predicting Missing Values**: By calculating UU^T, SPMF enables the prediction of missing values within the user-item matrix.
   - **Item-Item and User-User Similarity**: The matrix UU^T also provides insights into item-item and user-user similarities, which are crucial for recommendation tasks. This dual functionality streamlines the recommendation process.

5. **Probabilistic Framework and Cross-Validation**:

   - **Data Splitting for Training and Validation**: Unlike SVD and NMF, SPMF leverages a probabilistic framework that allows for data splitting into training and validation sets. This capability ensures that the model can be generalized well to unseen data.
   - **Model Adaptability**: The probabilistic nature of SPMF makes it adaptable for use with new, unseen data, enhancing its practical utility in dynamic environments.

These contributions demonstrate the innovative aspects of SPMF and its advantages over traditional matrix factorization methods, positioning it as a tool for recommendation systems. By addressing key limitations of existing methods and introducing new capabilities, SPMF advances the field of recommendation systems, providing more robust and interpretable recommendations.

3.3 Symmetric Probabilistic Binary Matrix Factorization (SPBMF)

In the continually evolving landscape of Artificial Intelligence (AI)-enabled digital entertainment, platforms like YouTube, Amazon, and Netflix have revolutionized video content consumption, functioning as major advertising channels generating substantial revenue through targeted advertisements. The effectiveness of their recommendation systems is critical for maximizing revenue, as these systems capture and retain user attention by proposing videos or items tailored to individual preferences. Accurate predictions of user preferences extend user engagement and enhance advertisement delivery potential. A significant challenge lies in recommending the most suitable content from overwhelmingly vast repositories, where user preferences are often simplified into binary ratings representing likes or dislikes. Developing a robust recommendation system for binary user-item rating matrices is thus crucial, as it improves user satisfaction by delivering more relevant content, thereby optimizing advertisement exposure and engagement.

In this section, we introduce a subtle modification to the objective function of SPMF to demonstrate its applicability in binary recommendation systems, specifically addressing the item recommendation challenge in the binary regime. Binary recommendation systems represent an application of SPMF where the Bernoulli distribution is pivotal in guiding the optimization process to accommodate the binary data matrix. We call this modified approach Symmetric Probabilistic Binary Matrix Factorization (SPBMF). By leveraging the core principles of SPMF, SPBMF interprets and anticipates user preferences based on binary ratings, thereby offering a robust solution to the challenge of video recommendations.

To this end, we transform each entry of the block matrix S in Equation 3.1 into a binary representation, referred to as the binary matrix $\mathcal{S}_{\text{bin}}$. This transformation involves two main steps:

Firstly, we normalize the off-diagonal blocks of matrix S row-wise (R_{RN} and R_{RN}^T), where all entries are normalized to be within the range 0 to 1. So for any given element $\mathfrak{s}_{ij}$ in the off-diagonal blocks of S, we define:

$$\mathcal{S}_{\text{bin}} = \begin{cases} 0, & \text{if } \mathfrak{s}_{ij} < t \\ 1, & \text{otherwise} \end{cases} \tag{3.34}$$

where t is a threshold typically set to 0.5 but can be adjusted to different levels according to the domain context. Secondly, the diagonal blocks of $\mathcal{S}_{\text{bin}}$ are defined by performing a correlation test on R^TR and RR^T, respectively. For entries where the p-value of the correlation test is below the significance level of 0.05, we assign a value of 1; otherwise, we assign a value of 0.

Below, we demonstrate the construction of the binary matrix $\mathcal{S}_{\text{bin}}$ using the following user-item rating matrix R introduced in Equation 2.14.

$$R = \begin{array}{c} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Matrix} & \text{Titanic} & \text{Inception} \end{array} \\ \begin{pmatrix} 4 & 1 & \text{NA} \\ 2 & 5 & 2 \\ 5 & 2 & 5 \\ \text{NA} & 5 & 5 \end{pmatrix} \end{array} \tag{3.35}$$

First, we replace NA with 0 and create the block matrix S:

$$S = \begin{bmatrix} R^TR & R^T \\ R & RR^T \end{bmatrix} = \begin{bmatrix} 45 & 24 & 29 & 4 & 2 & 5 & 0 \\ 24 & 55 & 45 & 1 & 5 & 2 & 5 \\ 29 & 45 & 54 & 0 & 2 & 5 & 5 \\ 4 & 1 & 0 & 17 & 13 & 22 & 5 \\ 2 & 5 & 2 & 13 & 33 & 30 & 35 \\ 5 & 2 & 5 & 22 & 30 & 54 & 35 \\ 0 & 5 & 5 & 5 & 35 & 35 & 50 \end{bmatrix}$$

Next, we use the L_2 norm to normalize the off-diagonal blocks of matrix S row-wise (R_{RN} and R_{RN}^T). By applying the binarization with $t = 0.5$ in the Equation 3.34, we obtain:

$$R_{RN} = \begin{bmatrix} 0.97 & 0.24 & 0 \\ 0.35 & 0.87 & 0.35 \\ 0.68 & 0.27 & 0.68 \\ 0 & 0.71 & 0.71 \end{bmatrix} \xrightarrow{\text{binarization with } t=0.5} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

In the second step, we calculate the correlation for the matrices R^TR and RR^T using the Spearman correlation with a significance level of 0.05. The resulting binary correlation matrices, $\mathrm{cor}(RR^T)$ and $\mathrm{cor}(R^TR)$, are as follows:

$$\mathrm{cor}(RR^T) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \mathrm{cor}(R^TR) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Finally, the binary matrix $\mathcal{S}_{\text{bin}}$ is constructed as follows:

$$\mathcal{S}_{\text{bin}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The primary objective of designing the binary matrix $\mathcal{S}_{\text{bin}}$ is to approximate it using the SPBMF approach. Our goal is to find a low-rank matrix U such that $\mathcal{S}_{\text{bin}}$ is approximated by UU^T. Therefore, we adopt a probabilistic approach to define the interactions between user-user, item-item, or user-item pairs, derived from the rows of U. This probability that captures the relationship between a pair of rows (i, j) in the matrix U, denoted by p_{ij}, can be modeled by a logistic function parameterized by the inner product $u_i^T u_j$ between the pair of rows (i, j). The sigmoid function transforms this dot product into a value within the $[0, 1]$ range, serving as a probability-like measure. This transformed value indicates the likelihood of interaction between the attributes of the user/item associated with u_i and those associated with u_j. This transformation is expressed as:

$$p_{ij} = \sigma(u_i^T u_j) = \frac{1}{1 + \exp(-u_i^T u_j)} \quad i, j = 1, \ldots, m + n, \tag{3.36}$$

where $\sigma(u_i^T u_j)$ represents the sigmoid function.

Referring to the Lemma 1, our focus is on the lower triangular and diagonal elements within $\mathcal{S}_{bin}$. Considering this, we assume that the distribution of each entry in the binary matrix $\mathcal{S}_{bin}$ follows a Bernoulli distribution, independent of other entries. This implies that the joint probability mass function of $\mathcal{S}_{bin}$ is derived from the product of interaction probabilities across the lower triangular and diagonal elements in matrix $\mathcal{S}_{bin}$. In other words:

$$p(\mathcal{S}_{bin}|U) = \prod_{i=1}^{n+m} \left(\prod_{j=1}^{i} p_{ij}^{\mathfrak{s}_{ij}} (1 - p_{ij})^{1-\mathfrak{s}_{ij}} \right),$$

where $\mathfrak{s}_{ij}$ represents the entries of $\mathcal{S}_{bin}$ associated with the lower triangular and diagonal elements. We also assume a Gaussian prior $p(U)$ with zero mean and variance σ^2 on the rows of matrix U.

SPBMF Method : (SPBMF for Recommendation Systems) Given the observed data $\mathcal{S}_{bin}$ and parameter σ^2, our goal is to approximate the binary block matrix $\mathcal{S}_{bin}$ using Bayesian inference.

Using Bayes' theorem, we can obtain the posterior distribution of the matrix U given the observed binary matrix $\mathcal{S}_{bin}$ and the model parameters σ^2. The posterior probability is given by:

$$p(U|\mathcal{S}_{bin}, \sigma^2) = \frac{p(\mathcal{S}_{bin}|U)p(U)}{p(\mathcal{S}_{bin}|\sigma^2)}, \tag{3.37}$$

where $p(\mathcal{S}_{bin}|U)$ is the likelihood function, $p(U)$ is the prior distribution, and $p(\mathcal{S}_{bin}|\sigma^2)$ is the normalization constant. The quantity $p(U|\mathcal{S}_{bin}, \sigma^2)$ is the posterior distribution of the matrix U given the observed block matrix $\mathcal{S}_{bin}$ and the model parameters σ^2. To find the maximum a posteriori (MAP) estimate of U, we need to maximize the logarithm of the posterior distribution:

$$\log p(U|\mathcal{S}_{bin}, \sigma^2) = \log p(\mathcal{S}_{bin}|U) + \log p(U) - \log p(\mathcal{S}_{bin}|\sigma^2). \tag{3.38}$$

Lemma 5. *The logarithm of the posterior distribution $p(U|\mathcal{S}_{bin}, \sigma^2)$ can be calculated as follows:*

$$\begin{aligned}\log p(U|\mathcal{S}_{bin}, \sigma^2) = &\sum_{i=1}^{n+m} \sum_{j=1}^{i} \left[\mathfrak{s}_{ij} u_i^T u_j - \log(1 + e^{u_i^T u_j}) \right] \\ &- \frac{n+m}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n+m} u_i^T u_i - \log p(\mathcal{S}_{bin}|\sigma^2).\end{aligned}$$

Proof. By substituting $p_{ij} = \sigma(u_i^T u_j)$ in Equation (3.38), we get:

$$\begin{aligned}\log p(U|\mathcal{S}_{bin}, \sigma^2) = &\sum_{i=1}^{n+m}\sum_{j=1}^{i}[\mathfrak{s}_{ij}\log\sigma(u_i^T u_j) + (1-\mathfrak{s}_{ij})\log(1-\sigma(u_i^T u_j))]\\ &-\frac{n+m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n+m} u_i^T u_i - \log p(\mathcal{S}_{bin}|\sigma^2).\end{aligned}$$

Using the fact that $\log\sigma(x) = -\log(1+e^{-x})$ and $\log(1-\sigma(x)) = -\log(1+e^{x})$, we can rewrite the above equation as:

$$\begin{aligned}\log p(U|\mathcal{S}_{bin}, \sigma^2) = &\sum_{i=1}^{n+m}\sum_{j=1}^{i}\left[\mathfrak{s}_{ij}\log\frac{1}{1+e^{-u_i^T u_j}} + (1-\mathfrak{s}_{ij})\log\frac{1}{1+e^{u_i^T u_j}}\right]\\ &-\frac{n+m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n+m} u_i^T u_i - \log p(\mathcal{S}_{bin}|\sigma^2)\\ = &\sum_{i=1}^{n+m}\sum_{j=1}^{i}\left[\mathfrak{s}_{ij}u_i^T u_j - \log(1+e^{u_i^T u_j})\right]\\ &-\frac{n+m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n+m} u_i^T u_i - \log p(\mathcal{S}_{bin}|\sigma^2).\end{aligned}$$

□

Lemma 6. *The gradient of the log-likelihood function* $\log p(U|\mathcal{S}_{bin}, \sigma^2)$ *with respect to* u_k *is given by:*

$$\frac{\partial}{\partial u_k}\log p(U|\mathcal{S}_{bin}, \sigma^2) = \sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}(u_j\delta_{ik} + u_i\delta_{jk}) + \sum_{i=1}^{n+m}\sum_{j=1}^{i}\frac{e^{u_i^T u_j}}{1+e^{u_i^T u_j}}(u_j\delta_{ik} + u_i\delta_{jk}) - \frac{1}{\sigma^2}u_k,$$

where δ_{ij} *represents the Kronecker delta, which equals 1 when* $i = j$ *and 0 otherwise.*

Proof. The gradient of the log-likelihood function with respect to u_k can be expressed as:

$$\begin{aligned}\frac{\partial}{\partial u_k}\log p(U|\mathcal{S}_{bin}, \sigma^2) = &\frac{\partial}{\partial u_k}\sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}u_i^T u_j - \frac{\partial}{\partial u_k}\sum_{i=1}^{n+m}\sum_{j=1}^{i}\log(1+e^{u_i^T u_j})\\ &-\frac{\partial}{\partial u_k}(\frac{n+m}{2}\log(2\pi\sigma^2)) - \frac{\partial}{\partial u_k}(\frac{1}{2\sigma^2}\sum_{i=1}^{n+m} u_i^T u_i) - \frac{\partial}{\partial u_k}(\log p(\mathcal{S}_{bin}|\sigma^2)).\end{aligned}$$

We have:

$$\begin{aligned}\frac{\partial}{\partial u_k}\sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}u_i^Tu_j &= \frac{\partial}{\partial u_k}\left(\sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}u_i^Tu_j\right)\\ &= \sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}\frac{\partial}{\partial u_k}(u_i^Tu_j)\\ &= \sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}(u_j\delta_{ik}+u_i\delta_{jk}).\end{aligned}$$

Similarly:

$$\begin{aligned}\frac{\partial}{\partial u_k}\sum_{i=1}^{n+m}\sum_{j=1}^{i}\log(1+e^{u_i^Tu_j}) &= \sum_{i=1}^{n+m}\sum_{j=1}^{i}\frac{\partial}{\partial u_k}\log(1+e^{u_i^Tu_j})\\ &= \sum_{i=1}^{n+m}\sum_{j=1}^{i}\frac{e^{u_i^Tu_j}}{1+e^{u_i^Tu_j}}\frac{\partial}{\partial u_k}(u_i^Tu_j)\\ &= \sum_{i=1}^{n+m}\sum_{j=1}^{i}\frac{e^{u_i^Tu_j}}{1+e^{u_i^Tu_j}}(u_j\delta_{ik}+u_i\delta_{jk}).\end{aligned}$$

Finally:

$$-\frac{\partial}{\partial u_k}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n+m}u_i^Tu_i\right) = -\frac{1}{2\sigma^2}\frac{\partial}{\partial u_k}(u_k^Tu_k) = -\frac{1}{2\sigma^2}(2u_k) = -\frac{1}{\sigma^2}u_k.$$

Now, we've derived a formula for the gradient of the log posterior probability $\frac{\partial}{\partial u_k}\log p(U|\mathcal{S}_{bin},\sigma^2)$ with respect to a row vector u_k:

$$\frac{\partial}{\partial u_k}\log p(U|\mathcal{S}_{bin},\sigma^2) = \sum_{i=1}^{n+m}\sum_{j=1}^{i}\mathfrak{s}_{ij}(u_j\delta_{ik}+u_i\delta_{jk}) + \sum_{i=1}^{n+m}\sum_{j=1}^{i}\frac{e^{u_i^Tu_j}}{1+e^{u_i^Tu_j}}(u_j\delta_{ik}+u_i\delta_{jk}) - \frac{1}{\sigma^2}u_k. \tag{3.39}$$

□

After finding the gradient of the logarithm of the log posterior probability, we apply the following steps to estimate the matrix U:

1. Initialization: Start by initializing the rows of matrix U with some initial values.

2. Iterative Update: This step refines the rows of matrix U iteratively to enhance its approximation of the binary matrix $\mathcal{S}_{bin}$.

$$u_k^{(t+1)} = u_k^{(t)} - \alpha \left(\sum_{i=1}^{n+m} \sum_{j=1}^{i} \mathfrak{s}_{ij}(u_j\delta_{ik} + u_i\delta_{jk}) + \sum_{i=1}^{n+m} \sum_{j=1}^{i} \frac{e^{u_i^T u_j}}{1 + e^{u_i^T u_j}}(u_j\delta_{ik} + u_i\delta_{jk}) - \frac{1}{\sigma^2}u_k \right). \tag{3.40}$$

Here:

- $u_k^{(t)}$ is the current value of the k-th row vector of U at iteration t.
- α is the learning rate, determining the step size in each iteration.

3. Convergence: Repeat the iterative updates for all row vectors in U for a fixed number of iterations or until a convergence criterion is met. Convergence is typically declared when the changes in the elements of U become very small.

4. Objective Achievement: The final matrix U obtained after these iterations will represent an estimate $\hat{U}$ of the matrix U that best approximate the binary matrix $\mathcal{S}_{bin}$.

- **Threshold Application:** An approach to convert $\hat{U}\hat{U}^T$ into $\hat{\mathcal{S}}_{bin}$ involves applying a threshold. This threshold operation serves as a means to interpret the continuous values generated by $\hat{U}\hat{U}^T$ into discrete binary values. To identify the optimal threshold for binarizing a predicted matrix, we compute the Area Under the Curve (AUC) from the False Positive Rate (FPR) and True Positive Rate (TPR) using the AUC function. The optimal threshold, denoted as t, is determined by maximizing the difference between the TPR and FPR. Subsequently, this threshold is utilized to convert the continuous predicted values into binary values: values exceeding t are set to 1, while values below t are set to 0.

- **Utilization of Obtained Binary Matrix $\hat{\mathcal{S}}_{bin}$:**
  - *Recommendation System:* With $\hat{\mathcal{S}}_{bin}$ generated, recommendations can be provided to users based on items corresponding to a value of 1. These items represent potential preferences or interests for users.
  - *Determining Similarities:* $\hat{\mathcal{S}}_{bin}$ also enables the determination of similarities between users or items corresponding to a value of 1. By analyzing the binary matrix, patterns emerge that enable the identification of similar user preferences or similar item characteristics, aiding in clustering or recommendation enhancement.

This process culminates in not only recommending items to users based on their preferences but also unveiling associations between users or items, thus contributing to the system's functionality and enhancing user experiences. Adjusting the threshold is key to achieving accurate and meaningful recommendations and associations.

In Algorithm 5 we explain a part of script to implement SPMF using the Bayesian inference with Bernoulli distribution.

Algorithm 5 Bayesian Estimation with Bernoulli Distribution using AUC

```
Require: User-item matrix $R$, a fixed $\sigma^2$, list of learning rates $\alpha$, number of iterations $T$,
    number of folds $F$
Ensure: Estimated matrix $\hat{U}$, best number of latent factors $k_{\text{best}}$, AUC based on $k_{\text{best}}$
 1: Replace missing values with 0 in $R$
 2: Calculate the normalized matrix $R_{RN}$ and $R_{CN}$
 3: Calculate the block matrix $\mathcal{S}_{bin}$
 4: Randomly partition the elements of the lower triangular part and diagonal of $\mathcal{S}_{bin}$ into
    $F$ folds
 5: Designate the data into training and validation sets: $\mathcal{S}_{bintrain}$ and $\mathcal{S}_{binval}$
 6: for $f = 1$ to $F$ do
 7:   for $k = 1$ to $K_{\max}$ do
 8:     Initialize a list of AUC scores $AUC_{val}[f][k]$ for each potential latent factor $k$
 9:     for $\alpha$ in $\alpha$ do
10:       Initialize $U$ with random values
11:       for $t = 1$ to $T$ do
12:         Compute the gradient of the loss function with respect to $U$ using $\mathcal{S}_{bintrain}$ and
            the equation (3.39)
13:         Update $U$ using gradient descent
14:       end for
15:       Compute the AUC using the original $\mathcal{S}_{binval}$
16:       Store the computed AUC in $AUC_{val}[f][k]$
17:     end for
18:   end for
19: end for
20: Find the best $k_{\text{best}}$ that maximizes the average AUC across folds: $k_{\text{best}} =$
    $\text{argmax}_k \left(\frac{1}{F}\sum_{f=1}^{F} AUC_{val}[f][k]\right)$
21: Find the best learning rate $\alpha^*$ for $k_{\text{best}}$: $\alpha^* = \arg\max_\alpha AUC_{val}[f][k_{\text{best}}]$
22: Initialize $U$ with random values and the obtained latent factor $k_{\text{best}}$
23: for $t = 1$ to $T$ do
24:   Compute the gradient of the loss function with respect to $U$ using the equation (3.39)
25:   Update $U$ using the best learning rate $\alpha^*$
26: end for
27: Calculate the estimated $\hat{\mathcal{S}}_{bin}$
28: Compute the AUC based on $k_{\text{best}}$
29: return $\hat{U}$, $k_{\text{best}}$, AUC
```

3.3.1 Applications of SPBMF

Graph algorithms and their applications play a central role in representing relationships. The fundamental nature of graphs, often captured through adjacency matrices, provides an intuitive way to represent connections and interactions. However, due to the symmetric nature of adjacency matrices, traditional matrix factorization techniques used in recommendation systems cannot analyze graph-induced datasets properly. SPBMF, by incorporating this symmetric constraint, offers an innovative approach to utilizing user-user and item-item interactions.

Social media platforms, where users are connected through friendships and shared interests in various items, such as books, movies, and products, provide an excellent source of graph-structured data. The SPBMF model intrinsically encodes these interactions into a graph, where an edge between two nodes denotes a meaningful relationship—whether a friendship or a shared interest. The absence of an edge implies a lack of interaction, enabling the model to distinguish between connected and disconnected nodes clearly.

Expanding this concept further, user-item recommendations can be visualized through a bipartite graph. In this graph, users and items form two distinct sets of nodes. An edge between a user node and an item node signifies a recommendation, providing a comprehensive network of preferences and suggestions. This bipartite graph not only captures the direct interactions between users and items but also highlights the intricate network of recommendations.

By merging these graph structures—user-user, item-item, and user-item interactions—into the binary matrix $\mathcal{S}_{bin}$, SPBMF constructs a robust and interpretable recommendation system. This system integrates complementary sources of information, building an integrative model for highly customized recommendations. In the social media domain, where user connectivity is paramount, this novel approach to binary matrix factorization allows the incorporation of friendship information and personalized user data, such as following favorite celebrities or influencers, to provide targeted recommendations of interest.

The result is a user-aware recommendation system that goes beyond typical suggestions. By focusing on the unique preferences and behaviors of each user, SPBMF optimizes recommended products and content to align perfectly with user interests. This not only enhances user experience but also boosts engagement and satisfaction by delivering highly relevant recommendations.

We demonstrate that graph factorization can be regarded as one of the applications of SPBMF, merging social media's connectivity network with personalized recommendations. This approach opens new horizons in the field of recommendation systems, making SPBMF a pivotal tool for optimizing and integrating user-centric recommendation systems.

3.3.2 Graph Factorization as an Application of SPBMF: A Toy Example

We demonstrate the application of SPBMF through a toy example that merges social media connectivity networks with personalized recommendations. This graph factorization approach highlights how SPBMF can integrate user-user and item-item interactions to provide a unified, interpretable representation of the data.

In Figure 3.2, the top left plot represents the user-user interaction graph for five users $U_1, U_2, \ldots, U_5$. In this graph, an edge between two nodes denotes a friendship, while the absence of an edge implies a lack of interaction. For example, there is an edge between the nodes U_1 and U_2, indicating that U_1 and U_2 are friends. Conversely, there is no edge between U_1 and U_3, signifying that U_1 and U_3 are not friends.

Similarly, the top right plot in Figure 3.2 depicts the item-item interaction graph for three movies. In this graph, an edge between two nodes represents a shared genre or significant similarity. For instance, the movies "action_1" (I_1) and "action_2" (I_2) share the same genre and are connected by an edge, indicating their similarity. On the other hand, the movie "romance" (I_3) shows no interactions with the other two movies, and hence, no edges are created between them.

The user-item interaction graph is depicted in the bottom left of Figure 3.2. In this graph, solid edges represent movies that have been recommended to users, the absence of an edge indicates that a movie has not been recommended to a user, and dotted edges signify that it is unknown whether a movie has been recommended to a user, thus requiring prediction using SPBMF. For example, User U_4 has been recommended I_3 and has not been recommended I_2. However, it is unclear whether User U_4 has been recommended I_1, and this recommendation needs to be predicted using SPBMF.

These graphs are integral to understanding the relationships and similarities between users and items. The user-user interaction graph helps in identifying social connections and influence among users, which can be crucial for recommendation systems. The item-item interaction graph, on the other hand, assists in understanding the similarities and differ-

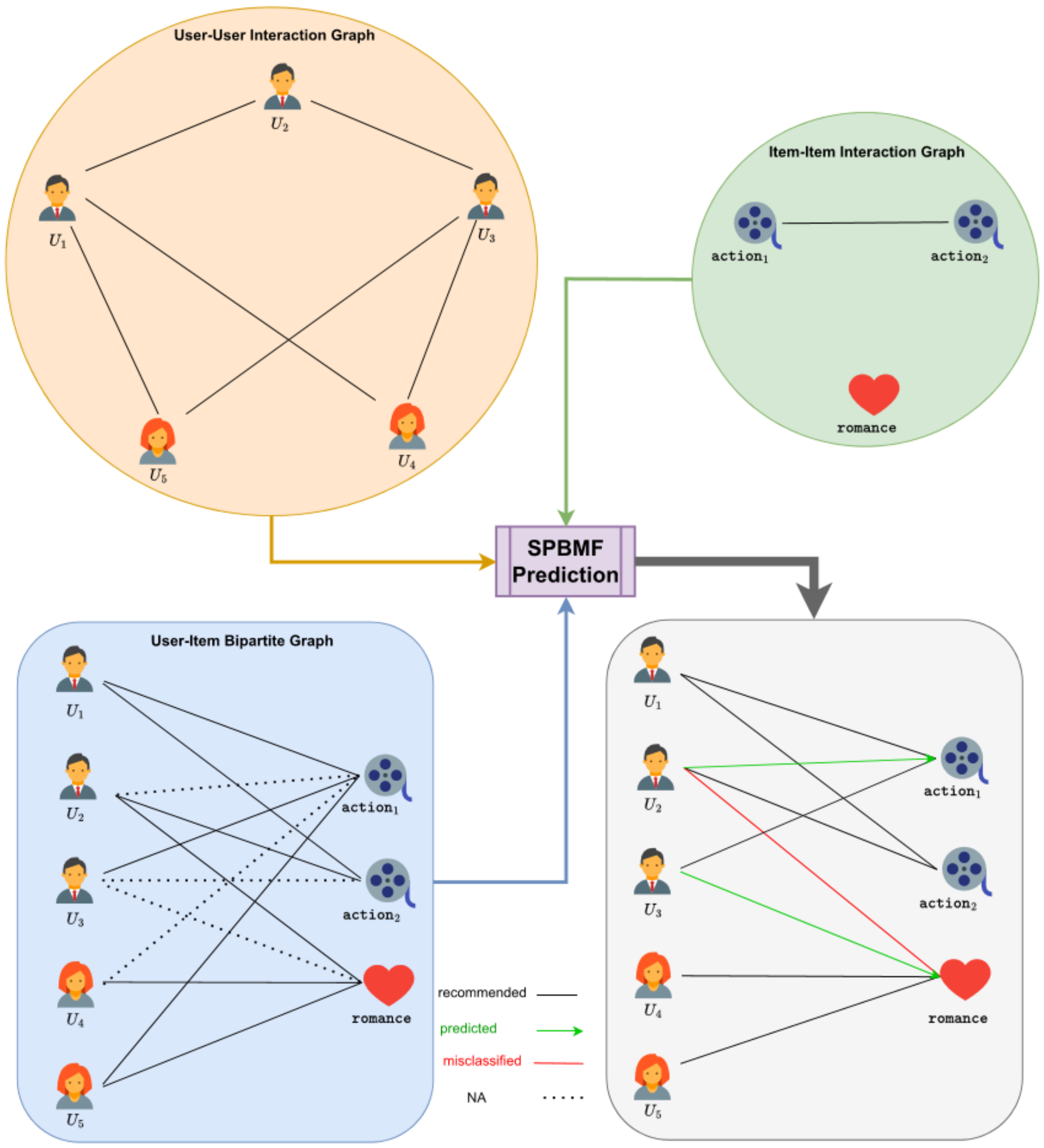

Figure 3.2 SPBMF Prediction

ences between items, aiding in more accurate item-based recommendations. The user-item interaction graph represents the direct interactions between users and items, capturing the preferences and behaviors of users towards specific items. This graph is essential for modeling the interactions in a recommendation system as it combines the insights from both user-user and item-item interactions.

We now present the binary matrices associated with each interaction graph. These matrices provide a clear representation of the interactions among users and items, facilitating the analysis and understanding of the underlying relationships.

The user-user interaction matrix, corresponding to the user-user interaction graph, is as follows:

$$\text{User-User Interaction} = \begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \begin{array}{c} \begin{array}{ccccc} U_1 & U_2 & U_3 & U_4 & U_5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}.$$

The item-item interaction matrix, corresponding to the item-item interaction graph, is illustrated as follows:

$$\text{Item-Item Interaction} = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \end{array} \begin{array}{c} \begin{array}{ccc} I_1 & I_2 & I_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}.$$

The user-item interaction matrix R, associated with the user-item interaction graph, is presented below:

$$\text{User-Item Interaction } R = \begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \begin{array}{c} \begin{array}{ccc} I_1 & I_2 & I_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ \text{NA} & 1 & 1 \\ 1 & \text{NA} & \text{NA} \\ \text{NA} & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \end{array}.$$

Having all these three matrices, we are able to construct the block matrix $\mathcal{S}_{bin}$ as below:

$$\mathcal{S}_{\text{bin}} = \begin{bmatrix} \text{item-item interaction matrix} & \text{item-user interaction matrix} \\ \text{user-item interaction matrix} & \text{user-user interaction matrix} \end{bmatrix}.$$

Or equivalently,

$$\mathcal{S}_{\text{bin}} = \begin{array}{c|cccccccc} & I_1 & I_2 & I_3 & U_1 & U_2 & U_3 & U_4 & U_5 \\ \hline I_1 & 1 & 1 & 0 & 1 & \text{NA} & 1 & \text{NA} & 1 \\ I_2 & 1 & 1 & 0 & 1 & 1 & \text{NA} & 0 & 0 \\ I_3 & 0 & 0 & 1 & 0 & 1 & \text{NA} & 1 & 1 \\ U_1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ U_2 & \text{NA} & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ U_3 & 1 & \text{NA} & \text{NA} & 0 & 1 & 1 & 1 & 1 \\ U_4 & \text{NA} & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ U_5 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}.$$

After constructing the block matrix $\mathcal{S}_{\text{bin}}$, we apply Algorithm 5 of SPBMF to estimate the matrix $\hat{U}$. By executing this algorithm in Python, we obtain the following estimated matrix $\hat{U}$:

$$\hat{U} = \begin{array}{c|cc} & \text{Action} & \text{Romance} \\ \hline I_1 & 0.93 & 0.43 \\ I_2 & 1.04 & 0 \\ I_3 & 0 & 1.02 \\ U_1 & 0.96 & 0.27 \\ U_2 & 0.83 & 0.38 \\ U_3 & 0.46 & 0.89 \\ U_4 & 0.19 & 0.83 \\ U_5 & 0.31 & 0.80 \end{array}.$$

Consequently, we compute the matrix $\hat{U}\hat{U}^T$ as follows:

$$\hat{U}\hat{U}^T = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} I_1 & I_2 & I_3 & U_1 & U_2 & U_3 & U_4 & U_5 \end{array} \\ \begin{pmatrix} 1.04 & 0.96 & 0.44 & 1.01 & \mathbf{0.93} & 0.82 & \mathbf{0.54} & 0.64 \\ 0.96 & 1.09 & 0.00 & 1.01 & 0.86 & \mathbf{0.48} & 0.20 & 0.33 \\ 0.44 & 0.00 & 1.04 & 0.28 & 0.39 & \mathbf{0.91} & 0.84 & 0.81 \\ 1.01 & 1.01 & 0.28 & 1.01 & 0.90 & 0.69 & 0.41 & 0.52 \\ \mathbf{0.93} & 0.86 & 0.39 & 0.90 & 0.83 & 0.73 & 0.48 & 0.57 \\ 0.82 & \mathbf{0.48} & \mathbf{0.91} & 0.69 & 0.73 & 1.01 & 0.83 & 0.86 \\ \mathbf{0.54} & 0.20 & 0.84 & 0.41 & 0.48 & 0.83 & 0.73 & 0.72 \\ 0.64 & 0.33 & 0.81 & 0.52 & 0.57 & 0.86 & 0.72 & 0.74 \end{pmatrix} \end{array}.$$

To determine the optimal threshold for binarizing a predicted matrix, the Area Under the Curve (AUC) is calculated from the False Positive Rate (FPR) and True Positive Rate (TPR) values using the AUC function. This metric provides a single measure of the overall performance of the prediction model. For our model, the AUC is 0.91. The optimal threshold, denoted as t, is found by maximizing the difference between the TPR and FPR. For our model, t is 0.725. This threshold is then used to convert the continuous predicted values into binary values: values above t are set to 1, and values below t are set to 0. In Figure 3.3, we present the Receiver Operating Characteristic (ROC) curve along with the AUC value. The ROC curve illustrates the trade-off between the TPR and FPR across different thresholds and demonstrates the performance of our prediction model. The point where the difference between the TPR and FPR is maximized corresponds to the optimal threshold t.

After determining the optimal threshold that maximizes the AUC (.725), we convert the matrix $\hat{U}\hat{U}^T$ to obtain $\hat{\mathcal{S}}_{\text{bin}}$ as follows:

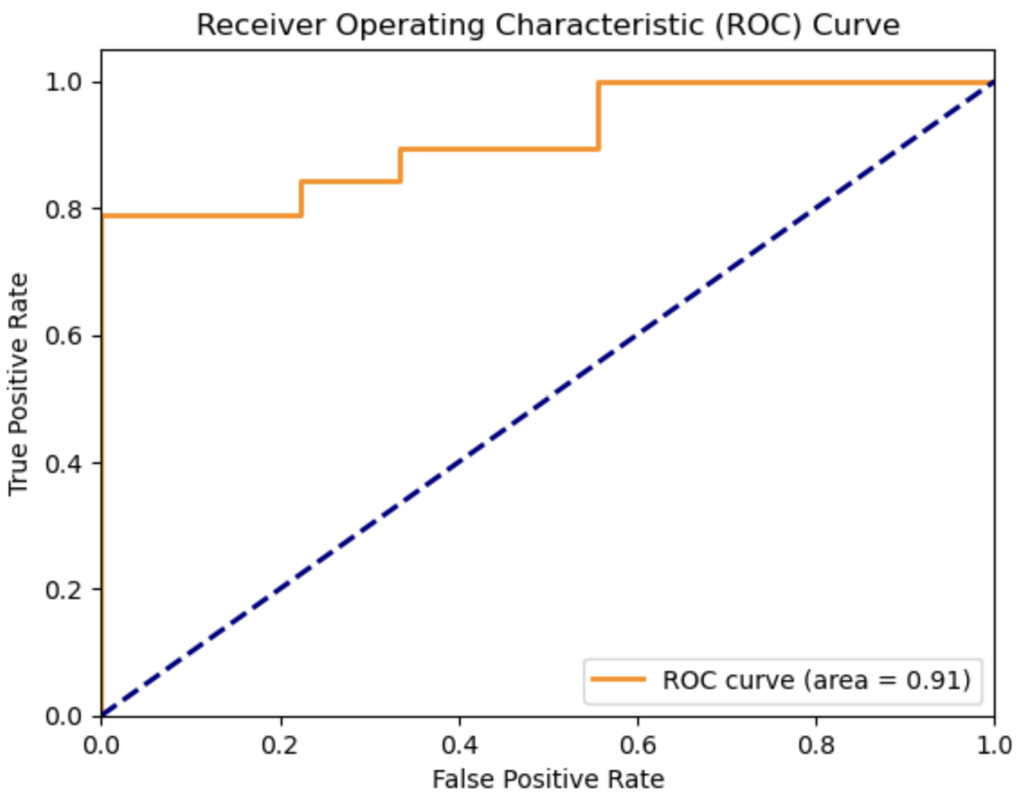

Figure 3.3 ROC Curve with AUC = 0.91. The optimal threshold t is 0.725.

$$\hat{\mathcal{S}}_{bin} = \begin{array}{c|cccccccc} & I_1 & I_2 & I_3 & U_1 & U_2 & U_3 & U_4 & U_5 \\ \hline I_1 & 1 & 1 & 0 & 1 & \mathbf{1} & 1 & \mathbf{0} & 0 \\ I_2 & 1 & 1 & 0 & 1 & 1 & \mathbf{0} & 0 & 0 \\ I_3 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 1 & 1 \\ U_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ U_2 & \mathbf{1} & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ U_3 & 1 & \mathbf{0} & \mathbf{1} & 0 & 1 & 1 & 1 & 1 \\ U_4 & \mathbf{0} & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ U_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}.$$

Therefore, the user-item prediction using the SPBMF method is as follows:

$$\text{SPBMF User-Item Interaction } \hat{R} = \begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \begin{array}{c} \begin{array}{ccc} I_1 & I_2 & I_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ \mathbf{1} & 1 & 0 \\ 1 & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

As illustrated by the numerical toy example, the key strength of SPBMF lies in its ability to seamlessly incorporate the symmetric nature of adjacency matrices, which is intrinsic to many real-world datasets, particularly those involving social networks and other graph-structured data. Traditional methods such as NMF, PMF, and SVD often struggle with such datasets due to their inability to handle symmetric constraints and binary interactions.

SPBMF extends these capabilities to binary datasets, making it suitable for applications where interactions are inherently binary, such as user-item recommendations based on likes or follows. By integrating various types of interactions—user-user, item-item, and user-item—into a unified model, SPBMF provides a framework for generating personalized recommendations.

Therefore, SPBMF provides an interpretable, flexible, and integrative solution for recommendation systems, particularly in the context of social media. It surpasses traditional unconstrained matrix factorization techniques by utilizing graph-structured data, making it a practical and innovative tool in user-centric recommendation technologies.

Interpretation of Estimated Matrix $\hat{U}$

Interpreting the matrix $\hat{U}$ provides valuable insights into user preferences and item characteristics. The matrix $\hat{U}$ represents the latent factors for movies and users following the estimation process. Each row corresponds to a movie or user, and each column corresponds to a latent factor associated with a genre ("Action" and "Romance" in this case). The values in the matrix indicate the strength of the association between each movie/user and each genre. Below, we elaborate on how meaningful insights can be derived from the obtained matrix $\hat{U}$ to enhance our understanding of user preferences and movie characteristics:

$$\hat{U} = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{array} \begin{array}{c} \begin{array}{cc} \text{Action} & \text{Romance} \end{array} \\ \begin{pmatrix} 0.93 & 0.43 \\ 1.04 & 0 \\ 0 & 1.02 \\ 0.96 & 0.27 \\ 0.83 & 0.38 \\ 0.46 & 0.89 \\ 0.19 & 0.83 \\ 0.31 & 0.80 \end{pmatrix} \end{array}.$$

First, we interpret the relations between each item and the latent factors.

Interpretation of Item Characteristics

- I_1: This item has a high preference score of 0.93 for Action and a moderate preference score of 0.43 for Romance. This suggests that I_1 is primarily an Action-oriented item but also has some elements that could appeal to fans of the Romance genre.
- I_2: This item has a very high preference score of 1.04 for Action and 0 for Romance. This indicates that I_2 is an exclusively Action-oriented item with no appeal to fans of the Romance genre.
- I_3: This item has a high preference score of 1.02 for Romance and 0 for Action. This indicates that I_3 is an exclusively Romance-oriented item with no appeal to fans of the Action genre.

Next, we interpret the relations between users and the latent factors.

Interpretation of Users' Preferences

- U_1: This user has a high preference score of 0.96 for Action and a low preference score of 0.27 for Romance. This suggests that U_1 prefers Action items over Romance.
- U_2: This user has a preference score of 0.83 for Action and 0.38 for Romance. This indicates a stronger preference for Action but still some interest in Romance.

- U_3: This user has a balanced preference with a score of 0.46 for Action and a high preference score of 0.89 for Romance. This suggests a moderate interest in Action and a strong preference for Romance.
- U_4: This user has a preference score of 0.19 for Action and 0.83 for Romance. This indicates a stronger preference for Romance with little interest in Action.
- U_5: This user has a preference score of 0.31 for Action and 0.80 for Romance. This suggests a higher preference for Romance but some interest in Action.

Missing Value Prediction

To interpret the matrix $\hat{R}$, which includes predicted values for missing entries, we have obtained the missing values using SPBMF and interpret them as follows:

$$\text{SPMF User-Item Interaction } \hat{R} = \begin{array}{c|ccc|} & \text{action}_1 & \text{action}_2 & \text{romance} \\ U_1 & 1 & 1 & 0 \\ U_2 & \mathbf{1} & 1 & 0 \\ U_3 & 1 & \mathbf{0} & \mathbf{1} \\ U_4 & \mathbf{0} & 0 & 1 \\ U_5 & 0 & 0 & 1 \end{array}$$

The values in $\hat{R}$ marked in bold represent the predicted values for the previously missing entries in R.

- U_2: The missing value for action$_1$ is predicted to be 1, indicating a strong preference for this item.
- U_3: The missing values for action$_2$ and Romance are predicted to be 0 and 1, respectively, suggesting no preference for action$_2$ but a strong preference for Romance.
- U_4: The missing value for action$_1$ is predicted to be 0, indicating no preference for this item.

3.4 Convergence Analysis for SPMF

In the vast landscape of recommendation systems, speed and interpretability are paramount. Traditional matrix factorization techniques, such as ALS, and SGD, rely on decomposing complex optimization problems into two parts, each with its own biconvex objective function. These methods can be slow due to the iterative back-and-forth required to optimize both parts simultaneously. Additionally, another inherent challenge of traditional matrix factorization techniques, such as NMF and PMF, is the rank deficiency issue. Rank deficiency can lead to numerical instability and inaccuracies in the factorization process, hindering the ability to find precise solutions. These rank deficiencies often require additional regularization or constraints, complicating the optimization process and potentially slowing convergence.

Our suggested SPMF offers a single, convex objective function such that this simplification leads to a more direct optimization process, resulting in faster convergence rates. Unlike ALS and SGD, which navigate a complex, two-part optimization landscape, SPMF with its unique single-factor optimization promises quicker and more interpretable results. The key to understanding SPMF's efficiency lies in its mathematical properties, particularly strong convexity and Lipschitz continuity. Strong convexity ensures that the objective function has a unique global minimum and converges faster under reasonable conditions. Lipschitz continuity of the gradient guarantees that the optimization steps are stable and smooth, preventing erratic changes that could slow down convergence.

In this chapter, we will analyze these properties in mathematical detail, demonstrating how they contribute to the faster convergence of SPMF. Specifically, we show that SPMF achieves a logarithmic convergence rate, denoted as $\mathcal{O}(\log(\frac{1}{\epsilon}))$, under conditions of strong convexity and Lipschitz continuity of the gradient. Moreover, we demonstrate the criteria under which the objective function of SPMF remains strongly convex. By setting an upper bound on the sum of eigenvalues of the approximation matrix UU^T, we ensure that the strong convexity condition holds. This analysis is essential for proving that SPMF not only converges faster but does so under rigorous mathematical guarantees.

Through this convergence analysis, we aim to highlight the mathematical properties of SPMF, providing a solid theoretical foundation for practical usage in recommendation systems. This chapter lay the groundwork for understanding why SPMF represents a step forward in matrix factorization techniques, promising faster and more interpretable recommendations.

In the following, we show the conditions under which the SPMF objective function is strongly convex and its gradient is Lipschitz continuous.

3.4.1 Strong Convexity of the SMPF Objective Function

To establish that the objective function of SPMF is μ-strongly convex, we first define μ-strong convexity. We then present two lemmas and provide a detailed proof of a theorem that demonstrates the μ-strong convexity of the SPMF objective function under certain conditions.

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be μ-strongly convex for $\mu > 0$ if for all $x, y \in \mathbb{R}^n$:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2.$$

This inequality indicates that f grows at least quadratically away from its minimizer [86].

Lemma 7. *Let $f(U) = \|\mathcal{S} - UU^T\|_F^2$ be the objective function of SPMF, where U is an $(n+m) \times k$ matrix that approximates $\mathcal{S}$, and $\mathcal{S}$ is the $(n+m) \times (n+m)$ symmetric block matrix defined in Equation (3.2). Under the condition:*

$$trace(UU^T) > \frac{1}{3} trace(\mathcal{S}),$$

the Hessian $\nabla^2 f(U)$ is positive definite.

Proof. Let's consider the objective function:

$$f(U) = \|\mathcal{S} - UU^T\|_F^2,$$

where U is an $(n+m) \times k$ matrix, and $\mathcal{S}$ is an $(n+m) \times (n+m)$ symmetric matrix. First, we compute the gradient of $f(U)$ with respect to U:

$$\nabla f(U) = \frac{\partial}{\partial U}\|\mathcal{S} - UU^T\|_F^2.$$

Using the result from matrix calculus for the gradient of the Frobenius norm squared, we have:

$$\nabla f(U) = -4(\mathcal{S} - UU^T)U.$$

Next, we compute the Hessian $H(U)$, which is the second derivative of $f(U)$ with respect to U:

$$H(U) := \nabla^2 f(U) = \frac{\partial}{\partial U}\left(-4(\mathcal{S} - UU^T)U\right).$$

Using matrix calculus and the product rule, we get:

$$H(U) = -4\left(\frac{\partial(\mathcal{S} - UU^T)}{\partial U}U + (\mathcal{S} - UU^T)\frac{\partial U}{\partial U}\right).$$

We now calculate each term separately. First, consider the term $\frac{\partial(\mathcal{S}-UU^T)}{\partial U}U$:

$$\frac{\partial(\mathcal{S} - UU^T)}{\partial U} = -2U^T.$$

Thus,

$$-4\left(\frac{\partial(\mathcal{S} - UU^T)}{\partial U}U\right) = 8UU^T.$$

Next, consider the term $(\mathcal{S} - UU^T)\frac{\partial U}{\partial U}$. This term simplifies to:

$$-4(\mathcal{S} - UU^T).$$

Combining these results, we get the Hessian:

$$H(U) = 12UU^T - 4\mathcal{S}.$$

To ensure $H(U)$ is positive definite, we need all eigenvalues of $12UU^T - 4\mathcal{S}$ to be positive:

$$12\lambda_i(UU^T) - 4\lambda_i(\mathcal{S}) > 0 \quad \text{for all } i.$$

This simplifies to:

$$3\lambda_i(UU^T) > \lambda_i(\mathcal{S}) \quad \text{for all } i.$$

Given that the trace of a matrix is the sum of its eigenvalues, we have:

$$\text{trace}(\mathcal{S}) = \sum_{i=1}^{n+m} \lambda_i(\mathcal{S}) = m + n.$$

We need to ensure:

$$\text{trace}(UU^T) > \frac{1}{3}\text{trace}(\mathcal{S}).$$

This condition is already provided in the lemma and guarantees that the Hessian $H(U)$ is positive definite. □

Lemma 8. *Let $f(U) = \|\mathcal{S} - UU^T\|_F^2$ be the objective function of SPMF, and suppose the condition trace$(UU^T) > \frac{1}{3}$trace$(\mathcal{S})$ is satisfied. Then, for any vector $x \in \mathbb{R}^{n+m}$ and the smallest eigenvalue $\lambda_{\min}$ of the Hessian $H(U)$, the following inequality holds:*

$$x^T H(U)x \geq \lambda_{\min}\|x\|^2.$$

Proof. Given that the condition $\text{trace}(UU^T) > \frac{1}{3}\text{trace}(\mathcal{S})$ holds, it follows from Lemma 7 that $H(U)$ is positive definite. This implies that for any non-zero vector x, the quadratic form $x^T H(U)x$ is always positive:

$$x^T H(U)x > 0 \quad \text{for all } x \neq 0.$$

Since $H(U) = 12UU^T - 4\mathcal{S}$ is symmetric, it can be diagonalized. Therefore, we can write it as:

$$H(U) = Q\Lambda Q^T,$$

where Q is an orthogonal matrix and Λ is a diagonal matrix containing the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n+m}$. Any vector x can be expressed in terms of the eigenvectors of $H(U)$. Let $x = Qy$, where y is a vector of coefficients. Then:

$$x^T H(U)x = (Qy)^T Q\Lambda Q^T Qy = y^T \Lambda y,$$

because $Q^T Q = I$. Since Λ is a diagonal matrix with eigenvalues λ_i, the quadratic form can be written as:

$$y^T \Lambda y = \sum_{i=1}^{n+m} \lambda_i y_i^2.$$

The smallest eigenvalue $\lambda_{\min}$ provides a lower bound:

$$\sum_{i=1}^{n+m} \lambda_i y_i^2 \geq \lambda_{\min} \sum_{i=1}^{n+m} y_i^2.$$

The term $\sum_{i=1}^{n+m} y_i^2$ is just the squared norm of y:

$$\sum_{i=1}^{n+m} y_i^2 = \|y\|^2.$$

Since $x = Qy$ and Q is an orthogonal matrix, it preserves the norm:

$$\|y\| = \|x\|.$$

Combining these results, we obtain:

$$x^T H(U)x = y^T \Lambda y \geq \lambda_{\min}\|y\|^2 = \lambda_{\min}\|x\|^2.$$

This inequality, $x^T H(U)x \geq \lambda_{\min}\|x\|^2$, shows that $x^T H(U)x$ is bounded below by the smallest eigenvalue of the Hessian matrix scaled by the squared norm of x. This is a fundamental property that helps us prove the strong convexity of the function $f(U)$. □

Theorem 1. *The objective function of SPMF $f(U) = \|\mathcal{S} - UU^T\|_F^2$ under the condition trace$(UU^T) > \frac{1}{3}$trace$(\mathcal{S})$ is μ-strongly convex, where $\mu > 0$ is a constant. Specifically, for any matrices $U, V \in \mathbb{R}^{(m+n)\times k}$ and for some $\mu > 0$, we have:*

$$f(V) \geq f(U) + \nabla f(U)^T(V - U) + \frac{\mu}{2}\|V - U\|_F^2.$$

Proof. Since the condition $\mathcal{S} < 3UU^T$ holds, $H(U)$ is positive definite, it implies that for any non-zero vector x:

$$x^T H(U)x > 0.$$

Let $\lambda_{\min}$ be the smallest eigenvalue of $H(U)$. Then from Lemma 8 for any vector x:

$$x^T H(U)x \geq \lambda_{\min}\|x\|^2.$$

Substituting $x = V - U$, we get:

$$(V - U)^T H(U)(V - U) \geq \mu\|V - U\|_F^2,$$

where $\mu = \lambda_{\min}$. For the function $f(U)$, the second-order Taylor expansion around a point U and evaluated at V can be written as:

$$f(V) \approx f(U) + \nabla f(U)^T(V - U) + \frac{1}{2}(V - U)^T H(U)(V - U).$$

Substituting this inequality into the second-order Taylor expansion, we get:

$$f(V) \geq f(U) + \nabla f(U)^T(V - U) + \frac{\mu}{2}\|V - U\|_F^2.$$

This completes the proof.

□

Now its the time to show under which condition the gradiant of SPMF objective funtion is Lipschitz continuous.

3.4.2 L-Lipschitz Continuity of the gradient of SPMF Objective Function

To demonstrate that the gradient of the SPMF objective function is L-Lipschitz continuous, we begin by defining L-Lipschitz continuity. We then present a theorem accompanied by a detailed proof.

Definition 2. *The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-Lipschitz continuous for $L > 0$ if for all $x, y \in \mathbb{R}^n$:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

This condition ensures that the gradient of f does not change too rapidly [86].

Theorem 2. *The gradient $\nabla f(U)$ of the SPMF objective function $f(U) = \|\mathcal{S} - UU^T\|_F^2$ is Lipschitz continuous. That is, there exists a constant $L > 0$ such that for any matrices U and V:*

$$\|\nabla f(U) - \nabla f(V)\|_F \leq L\|U - V\|_F.$$

Proof. The gradient of the function $f(U) = \|\mathcal{S} - UU^T\|_F^2$ is given by:

$$\nabla f(U) = -4(\mathcal{S} - UU^T)U.$$

To determine when this gradient is Lipschitz continuous, we analyze the difference in the gradient at two points U and V:

$$\nabla f(U) = -4(\mathcal{S} - UU^T)U,$$

$$\nabla f(V) = -4(\mathcal{S} - VV^T)V.$$

The difference in the gradients is:

$$\nabla f(U) - \nabla f(V) = -4\left[(\mathcal{S} - UU^T)U - (\mathcal{S} - VV^T)V\right]$$

$$= -4\left[(\mathcal{S}U - UU^TU) - (\mathcal{S}V - VV^TV)\right]$$

$$= -4\left[\mathcal{S}(U - V) + VV^TV - UU^TU\right].$$

To establish the Lipschitz continuity, we need to bound the Frobenius norm of this difference:

$$\|\nabla f(U) - \nabla f(V)\|_F.$$

The Frobenius norm satisfies the triangle inequality, which states that for any two matrices U and V:

$$\|U + V\|_F \leq \|U\|_F + \|V\|_F.$$

Applying this to our expanded form:

$$\|\mathcal{S}(U - V) + (VV^TV - UU^TU)\|_F \leq \|\mathcal{S}(U - V)\|_F + \|(VV^TV - UU^TU)\|_F.$$

The Frobenius norm is submultiplicative, meaning that for any two matrices U and V:

$$\|UV\|_F \leq \|U\|_F\|V\|_F.$$

Applying this property to the two terms we have:

$$\|\mathcal{S}(U - V)\|_F \leq \|\mathcal{S}\|_F\|U - V\|_F$$

and

$$\begin{aligned}\|VV^TV - UU^TU\|_F &= \|VV^TV - VU^TU + VU^TU - UU^TU\|_F \\ &= \|V(V^TV - U^TU) + (V - U)U^TU\|_F \\ &\leq \|V\|_F\|V^TV - U^TU\|_F + \|V - U\|_F\|U^TU\|_F.\end{aligned}$$

To bound $\|V^TV - U^TU\|_F$, we use:

$$\|V^TV - U^TU\|_F \leq \|V^T(V - U) + (V^T - U^T)U\|_F$$

$$\leq \|V\|_F\|V - U\|_F + \|V - U\|_F\|U\|_F.$$

Combining these bounds, we get:

$$\|\nabla f(U) - \nabla f(V)\|_F \leq 4\|\mathcal{S}\|_F\|U - V\|_F + 4\left(\|V\|_F(\|V\|_F + \|U\|_F) + \|U\|_F^2\right)\|U - V\|_F.$$

Let $C = \|\mathcal{S}\|_F + \|V\|_F(\|V\|_F + \|U\|_F) + \|U\|_F^2$, then:

$$\|\nabla f(U) - \nabla f(V)\|_F \leq 4C\|U - V\|_F.$$

Thus, $\nabla f(U)$ is Lipschitz continuous with Lipschitz constant $L = 4C$. The gradient $\nabla f(U) = -4(\mathcal{S} - UU^T)U$ is Lipschitz continuous if the Frobenius norm of $\mathcal{S}$ and U are bounded. The Lipschitz constant L depends on the Frobenius norms of $\mathcal{S}$, U, and the changes in U. Specifically, $L = 4(\|\mathcal{S}\|_F + \|V\|_F(\|V\|_F + \|U\|_F) + \|U\|_F^2)$. $\square$

Given that the SPMF objective function is μ-strongly convex and its gradient $\nabla f(U)$ is L-Lipschitz continuous, we can apply Theorem 3 from the appendix. Theorem 3 asserts that for any strongly convex function with a Lipschitz continuous gradient, the convergence rate of optimization algorithm is $\mathcal{O}(\log(1/\epsilon))$. Therefore, under the established conditions of strong convexity and Lipschitz continuity, the SPMF objective function achieves a convergence rate of $\mathcal{O}(\log(1/\epsilon))$. This result underscores the efficiency and robustness of the SPMF approach in reaching the global minimum, thereby providing a solid theoretical foundation for its practical application in recommendation systems.

3.5 Regularization and Strong Convexity in SPMF

Regularization is a technique in matrix factorization models to prevent overfitting and to improve generalization by incorporating additional information or constraints into the model. In the context of SPMF, regularization can be achieved by imposing a prior distribution on the low-rank matrix U. One common approach is to incorporate a Gaussian prior, which introduces a penalty term in the objective function, thereby promoting simpler and more robust solutions. This not only aids in controlling the complexity of the model but also ensures that the optimization problem remains well-behaved. Specifically, incorporating a Gaussian prior $N(0, \sigma^2)$ on U introduces a regularization term that contributes to the strong convexity of the objective function, a desirable property for ensuring convergence to a unique minimum. The following lemma formalizes this effect.

Lemma 9. *Incorporating a Gaussian prior $N(0, \sigma^2)$ on the low-rank matrix U in the objective function introduces a regularization term $\frac{\|U\|_F^2}{2\sigma^2}$. This regularization term ensures that the objective function is strongly convex.*

Proof. To demonstrate why the regularization term $\frac{\|U\|_F^2}{2\sigma^2}$ ensures strong convexity, we need to show that it adds a positive definite quadratic component to the objective function. Consider the objective function:

$$\mathcal{L}(U) = \frac{1}{2\sigma^2}\|\mathcal{S} - UU^T\|_F^2 + \frac{\|U\|_F^2}{2\sigma^2}.$$

The regularization term $\frac{\|U\|_F^2}{2\sigma^2}$ can be interpreted as adding a ridge regression penalty to the optimization problem. Specifically, $\|U\|_F^2 = \sum_{i,j} u_{ij}^2$ is a sum of squared elements of U, which is always non-negative. To see why this term ensures strong convexity, consider the Hessian of the regularization term. The Hessian of $\frac{\|U\|_F^2}{2\sigma^2}$ with respect to U is:

$$\nabla^2 \left(\frac{\|U\|_F^2}{2\sigma^2} \right) = \frac{1}{\sigma^2} I_{(n+m)k},$$

where $I_{(n+m)k}$ is the $(n+m)k \times (n+m)k$ identity matrix. This indicates that the Hessian is positive definite and hence adds a strictly positive term to the overall Hessian of the objective function. For a function $f(U)$ to be μ-strongly convex, its Hessian must satisfy:

$$\nabla^2 f(U) \geq \mu I$$

for some $\mu > 0$. In our case, the Hessian of the regularization term $\frac{\|U\|_F^2}{2\sigma^2}$ provides a constant $\frac{1}{\sigma^2}$, which acts as μ. Combining the data fitting term and the regularization term, we get:

$$\mathcal{L}(U) = \frac{1}{2\sigma^2} \|\mathcal{S} - UU^T\|_F^2 + \frac{\|U\|_F^2}{2\sigma^2}.$$

The term $\frac{\|U\|_F^2}{2\sigma^2}$ ensures that the objective function $\mathcal{L}(U)$ is $\frac{1}{\sigma^2}$-strongly convex. This strong convexity is essential for ensuring that optimization algorithms such as gradient descent converge to a unique minimum. Thus, incorporating the Gaussian prior $N(0, \sigma^2)$ on U imposes a strong convexity constraint on the objective function, facilitating reliable and robust optimization. $\square$

3.6 Error Bound for SPMF

To understand the lower bound of the error term in matrix approximation, we need to clarify what constitutes the error and why determining its lower bound is essential. The error term $J = \|\mathcal{S} - UU^T\|_F^2$ represents the discrepancy between the original symmetric matrix $\mathcal{S}$ and its low-rank approximation UU^T. This error quantifies how well U captures the essential features of $\mathcal{S}$. By investigating the lower bound of this error, we aim to identify the minimal unavoidable error that arises due to the inherent properties of $\mathcal{S}$. Specifically, if $\mathcal{S}$ contains negative eigenvalues, these contribute to the error because a low-rank approximation cannot fully capture all negative eigenvalues if the rank k is less than the number of negative eigenvalues r. In Lemma 10 we provide an insight into the minimum error that any low-rank approximation must incur, which is a consideration for matrix factorization techniques. By establishing this lower bound, we can better understand the limitations and performance of our matrix approximation methods, guiding us in selecting appropriate ranks and assessing the quality of our approximations.

Lemma 10. *Let $\mathcal{S}$ be an $(n+m)\times(n+m)$ symmetric matrix that is not positive semidefinite, with $r \leq n+m$ negative eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$. The objective function*

$$J = \|\mathcal{S} - UU^T\|_F^2$$

is always at least $\sum_{p=1}^{r} \lambda_p^2$, irrespective of the value of k in the $(n+m) \times k$ matrix U. The minimum value of k at which this lower bound for the error is guaranteed is $k = r$.

Proof. Since $\mathcal{S}$ is symmetric, it is diagonalizable. Therefore, we can write $\mathcal{S}$ as:

$$\mathcal{S} = Q\Lambda Q^T,$$

where Q is an orthogonal matrix and Λ is a diagonal matrix containing the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n+m}$. The Frobenius norm of $\mathcal{S}$ can be expressed in terms of its eigenvalues:

$$\|\mathcal{S}\|_F^2 = \sum_{i=1}^{n+m} \lambda_i^2.$$

For any rank-k approximation UU^T, its Frobenius norm squared can be expressed in terms of the eigenvalues of $\mathcal{S}$. To minimize $\|\mathcal{S}-UU^T\|_F^2$, the rank-k approximation UU^T must capture the largest k eigenvalues of $\mathcal{S}$. This is because, as explained in Section 3.2, the eigenvalue decomposition provides the optimal low-rank approximation in the Frobenius norm sense. However, since $\mathcal{S}$ has r negative eigenvalues, these cannot be fully captured by any rank-k approximation if $k < r$. The error in the approximation is given by the sum of the squares of the eigenvalues that are not captured by UU^T. Therefore, the minimum error must include at least the contribution from these negative eigenvalues:

$$J \geq \sum_{p=1}^{r} \lambda_p^2.$$

This holds true irrespective of the value of k because at least these r negative eigenvalues will contribute to the error. To determine the minimum value of k that guarantees this lower bound for the error, we note that k must be at least r to fully capture the r negative eigenvalues. If $k < r$, some of these negative eigenvalues will be excluded, and the error will be higher than $\sum_{p=1}^{r} \lambda_p^2$. Therefore, the minimum value of k at which this lower bound for the error is guaranteed is $k = r$.

□

This lemma highlights the necessity of choosing k appropriately in matrix approximation problems to ensure minimal error, especially when dealing with matrices that are not positive semidefinite.

CHAPTER 4 EXPERIMENTAL RESULTS

In this chapter, we present the key results obtained in Chapter 3 and demonstrate the originality and contributions of this thesis through experimental results. Chapter 3 introduced the SPMF model, highlighting its comparative advantages over existing techniques like SVD, NMF, and PMF. We demonstrated SPMF's ability to convert the original matrix into a symmetric blockwise matrix, providing interpretability advantages by modeling user-user and item-item interactions from a probabilistic perspective.

We will implement the SPMF algorithm using Maximum Likelihood Estimation and Bayesian inference, as discussed in Chapter 3 (Sections 3.2.1, 3.2.2, and 3.2.3). For clarity, we denote the low-rank estimation approaches as "SPMF Normal" and "SPMF Laplace," representing Maximum Likelihood Estimation using Normal and Laplacian distributions, respectively. Similarly, we use "SPMF Bayes" to indicate estimation performed using Bayesian inference with Gaussian prior distributions.

To evaluate the performance of these models, we will conduct experiments using subsets of the MovieLens 100K and FilmTrust datasets. We will demonstrate how to determine the optimal number of latent factors (k) for the SPMF model in recommendation systems. To identify the optimal k, we will employ K-fold cross-validation to evaluate the SPMF model's performance across different values of k. Model performance is assessed using RMSE, which provides a measure of the model's accuracy and recommendation relevance. By systematically varying k and recording the corresponding performance metrics, we will identify the k value that minimizes RMSE. Additionally, we will assess model complexity and potential overfitting by comparing training and validation performance. This methodology will enable us to determine the optimal number of latent factors for the SPMF model, ensuring a balance between model complexity and performance.

We will highlight the role of hyperparameter tuning in the overall capability of SPMF. Identifying the most important parameters to optimize will provide a framework for achieving the best possible model performance.

By transforming a biconvex optimization problem into a convex one, SPMF ensures a unique optimal solution and improved convergence properties. This transformation consolidates two distinct factors into a single unified low-rank factor, enhancing interpretability. We will show how SPMF simplifies the factorization process by using a single unified matrix that incorporates both user-user and item-item relationships. This approach alleviates rank deficiency problems and improves the factorization process. By comparing the actual and approximated user-user and item-item similarity matrices, we will demonstrate how SPMF provides this information in a single matrix without computing them separately. We will also show how SPMF can be used for Collaborative Filtering RS and Content-Based RS, and how SPMF can predict missing values. Additionally, by applying SPMF on subsets of the MovieLens 100K and FilmTrust datasets, we will show that SPMF converges faster than SVD, NMF, and PMF.

Finally, we will undertake a comparative analysis of the performance of three SPMF models—"SPMF Normal", "SPMF Laplace", and "SPMF Bayes"—against SVD, NMF, and PMF. This evaluation will be conducted through a series of experiments utilizing subsets of the MovieLens 100K and FilmTrust datasets.

The following sections detail the experimental methodology, performance metrics, and results of each model.

4.1 Data Description

In this Section, we provide a summarized description of the two datasets, MovieLens 100K and FilmTrust, that we used for our experiments in this chapter.

MovieLens 100K Dataset

The MovieLens 100K dataset, created by the GroupLens Research Project at the University of Minnesota, comprises 100,000 ratings (ranging from 1 to 5 stars) from 943 users on 1682 movies. Each user has rated a minimum of 20 movies, and the dataset includes simple demographic information such as age, gender, occupation, and zip code. The data was collected from the MovieLens website between September 19th, 1997, and April 22nd, 1998. A cleaning process removed users with fewer than 20 ratings or incomplete demographic information. This dataset has been utilized in several publications, including [11], [10], and [87].

FilmTrust Dataset

The FilmTrust dataset, collected from the FilmTrust website, contains 35,497 ratings (ranging from 0.5 to 4.0 stars) from 1,508 users on 2,071 movies. The ratings reflect users' preferences, with each rating incremented by 0.5 stars. The dataset is particularly sparse, as many users have rated only a small subset of the available movies. The dataset was collected from FilmTrust, an online platform where users rate movies and express trust in other users. Also, this dataset has been used in various research studies focused on recommender systems, collaborative filtering, and trust-based recommendation models. Some studies that have utilized this dataset include: [88], [89], and [90].

4.2 Data Preprocessing

Due to the volume of the two datasets, MovieLens 100K and FilmTrust, we randomly selected 50 users and 20 movies from the MovieLens 100K dataset, and 100 users and 50 movies from the FilmTrust dataset. To ensure robustness and generalizability, we generated 20 samples from each dataset using this random selection approach. This method allowed us to assess the consistency and reliability of the models across different subsets of data.

Following this, we computed the normalized matrices R_{RN} and R_{CN}. To prepare the input for SPMF, we formed the block matrix $\mathcal{S}$ as described below. The randomly selected datasets, comprising m users and n movies, are organized into the user-item rating matrix R as follows with missing values replaced by 0.

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & r_{jk} & \vdots \\ r_{m1} & \cdots & r_{mn} \end{bmatrix}$$

The row-normalized matrix R_{RN} for R can be calculated as follows:

$$R_{RN} = \begin{bmatrix} \frac{r_{11}}{\sqrt{r_{11}^2+\cdots+r_{1n}^2}} & \cdots & \frac{r_{1n}}{\sqrt{r_{11}^2+\cdots+r_{1n}^2}} \\ \vdots & \ddots & \vdots \\ \frac{r_{m1}}{\sqrt{r_{m1}^2+\cdots+r_{mn}^2}} & \cdots & \frac{r_{mn}}{\sqrt{r_{m1}^2+\cdots+r_{mn}^2}} \end{bmatrix}.$$

Similarly, the column-normalized matrix R_{CN} for R can be calculated as follows:

$$R_{CN} = \begin{bmatrix} \frac{r_{11}}{\sqrt{r_{11}^2+\cdots+r_{m1}^2}} & \cdots & \frac{r_{1n}}{\sqrt{r_{1n}^2+\cdots+r_{mn}^2}} \\ \vdots & \ddots & \vdots \\ \frac{r_{m1}}{\sqrt{r_{11}^2+\cdots+r_{m1}^2}} & \cdots & \frac{r_{mn}}{\sqrt{r_{1n}^2+\cdots+r_{mn}^2}} \end{bmatrix}.$$

Finally, the matrix $\mathcal{S}$ is constructed as follows:

$$\mathcal{S} = \begin{bmatrix} R_{CN}^T R_{CN} & R_{RN}^T \\ R_{RN} & R_{RN} R_{RN}^T \end{bmatrix}.$$

Because $\mathcal{S}$ is symmetric, we transform it into a lower triangular matrix to optimize the computation process. This transformation allows us to approximate the upper-side elements, which are initially set to 0, using their counterparts from the lower side after the approximation process is completed.

$$\mathcal{S}_{\text{lower}} = \begin{bmatrix} \mathfrak{s}_{11} & 0 & 0 & \cdots & 0 \\ \mathfrak{s}_{21} & \mathfrak{s}_{22} & 0 & \cdots & 0 \\ \mathfrak{s}_{31} & \mathfrak{s}_{32} & \mathfrak{s}_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathfrak{s}_{n+m,1} & \mathfrak{s}_{n+m,2} & \mathfrak{s}_{n+m,3} & \cdots & \mathfrak{s}_{n+m,n+m} \end{bmatrix}.$$

After constructing the block matrix $\mathcal{S}$, we provide details about data split and cross-validation in the following section.

4.3 Data Split and K-Fold Cross-Validation

When evaluating recommender systems or any machine learning model, it is essential to divide the datasets into training and testing sets to assess their performance. In this thesis, we will use K-fold cross-validation, a widely adopted approach for model evaluation that provides a more reliable estimate of the model's performance. K-fold cross-validation involves randomly dividing the dataset into K subsets, or folds, of equal size. The model is then trained on $K-1$ folds and tested on the remaining fold. This process is repeated K times, with each fold serving as the test set exactly once. The final evaluation score in K-fold cross-validation is calculated as the average of the K evaluation scores obtained. In our analysis, the measure of performance is RMSE.

In this thesis, we partition the lower triangular portion and diagonal of the matrix $\mathcal{S}$ into training and validation folds based on Alex Williams' approach[1], as explained in Section 3.2.1. Initially, the elements within the lower triangular part and diagonal of the matrix $\mathcal{S}$ are randomly distributed across multiple folds, with our choice being 5 ($K = 5$) folds—aiming for a balance between computational efficiency and statistical significance.

It is important to note that a fixed seed is utilized throughout this random partitioning process to ensure reproducibility. Subsequently, during the training and validation stages, we designate $\mathcal{S}_{\text{train}}$ for training purposes and $\mathcal{S}_{\text{val}}$ for validation. Cross-validation is employed solely to determine the optimal number of latent factors k and the best learning rate α.

4.4 Parameter Tuning

In our optimization process, the goal is to fine-tune three crucial parameters: the number of latent factors k, the learning rate α, and the number of iterations T used in the gradient descent process to estimate $\hat{U}$. Each of these parameters plays a role in shaping the performance of our model. Determining the optimal number of iterations in the gradient descent algorithm involves a process known as hyperparameter tuning. This method entails experimenting with different values and selecting the set that yields the best performance on a validation set. Common approaches to hyperparameter tuning include grid search, which systematically explores a range of hyperparameter values.

In our specific case, we have predefined lists for the learning rate (α), the number of iterations (T), and the number of latent factors (k), namely $\alpha \in \{10^{-3}, 10^{-2}, 10^{-1}\}$, $T \in \{50, 100, 150, 200, 250\}$, and $k \in [2, k_{\max}]$, where $k_{\max}$ is the number of genres. These values were chosen to balance stability and convergence speed, avoiding the risks associated with very small learning rates (α) and preventing overfitting with the chosen values for T. Practical constraints and computational feasibility were also considered, with the predefined grid representing a compromise. Hyperparameter tuning is iterative, allowing adjustments based on initial insights, ensuring an exploration of reasonable values for optimal model performance.

To determine the combination of α, T, and the number of latent factors k that minimizes the RMSE on the validation set $\mathcal{S}_{\text{val}}$, we employ three nested loops—one for each parameter. Within these loops, we iterate over the defined ranges for α, T, and k, recording the

[1]The approach for cross-validation was inspired by the method outlined in a blog post by Alex Williams. `https://alexhwilliams.info/itsneuronalblog/2018/02/26/crossval/`

RMSE on the validation set $\mathcal{S}_{\text{val}}$ for each combination. The set of α, T, and k resulting in the lowest RMSE is then returned, representing the optimal configuration for our model. This exploration ensures that our model is fine-tuned to deliver the best possible performance.

In the next step, we fit the three SPMF models ("SPMF Normal", "SPMF Laplace," and "SPMF Bayes") on the same preprocessed data, utilizing a fixed seed of 42 as described in Section 4.2. We initiate our experiments with a subset of the MovieLens 100K dataset, followed by a subset of the FilmTrust dataset. For each SPMF model, we detail the fitting procedure on the dataset and the prediction of missing values. To evaluate the performance of the three SPMF models against SVD, NMF, and PMF, we generate 20 random subsets from each dataset for comparison.

4.5 Experiment Results Using the MovieLens 100K Dataset

We analyzed the performance of "SPMF Normal," "SPMF Laplace," and "SPMF Bayes" on a subset of the MovieLens 100K dataset, which has been preprocessed as detailed in Section 4.2. By identifying the champion model among these SPMF variants, we compared the user-user similarity matrix, item-item similarity matrix, and their associated heatmaps, both for the original data and their approximations. Using the results, we implemented Collaborative Filtering and Content-Based Filtering within the framework of SPMF to make movie recommendations and predict missing values using the approximated $\hat{\mathcal{S}}$. Moreover, we compared the convergence rate of the champion SPMF model with SVD, NMF, and PMF, consistently utilizing the same randomly selected dataset.

To ensure robustness and generalizability, we generated 20 samples from the MovieLens 100K dataset using this random selection approach. This method allowed us to assess the consistency and reliability of the models across different subsets of data. We compared the performance of the three SPMF models with SVD, NMF, and PMF in terms of RMSE. The comparison of the models' performance on these samples provided insights into their predictive accuracy and stability in various scenarios, ultimately guiding us in recommending the best approach for movie recommendations and missing value prediction.

First, we implement the "SPMF Normal" model as outlined in the following sections.

SPMF Normal

In this section, we apply SPMF on the lower triangular part and diagonal matrix $\mathcal{S}$ using the "SPMF Normal" model. We implement the algorithm detailed in Algorithm 2, utilizing the loss function described in Equation 3.14. The lower triangular part and diagonal matrix $\mathcal{S}$ is divided into $\mathcal{S}_{\text{train}}$ for training purposes and $\mathcal{S}_{\text{val}}$ for validation, as outlined in Section 4.3 and 4.4 respectively.

The algorithm is run with different combinations of hyperparameters discussed in Section 4.4, and we record the lowest RMSE on the validation set $\mathcal{S}_{\text{val}}$. For the "SPMF Normal" model, the optimal hyperparameters yielding the lowest RMSE=1.52 are $\alpha = 0.1$, $T = 100$, and $k = 11$, indicating higher RMSE values using other combinations. Figure 4.1 illustrates how the RMSE varies with the number of latent factors, using $\alpha = 0.1$ and $T = 100$.

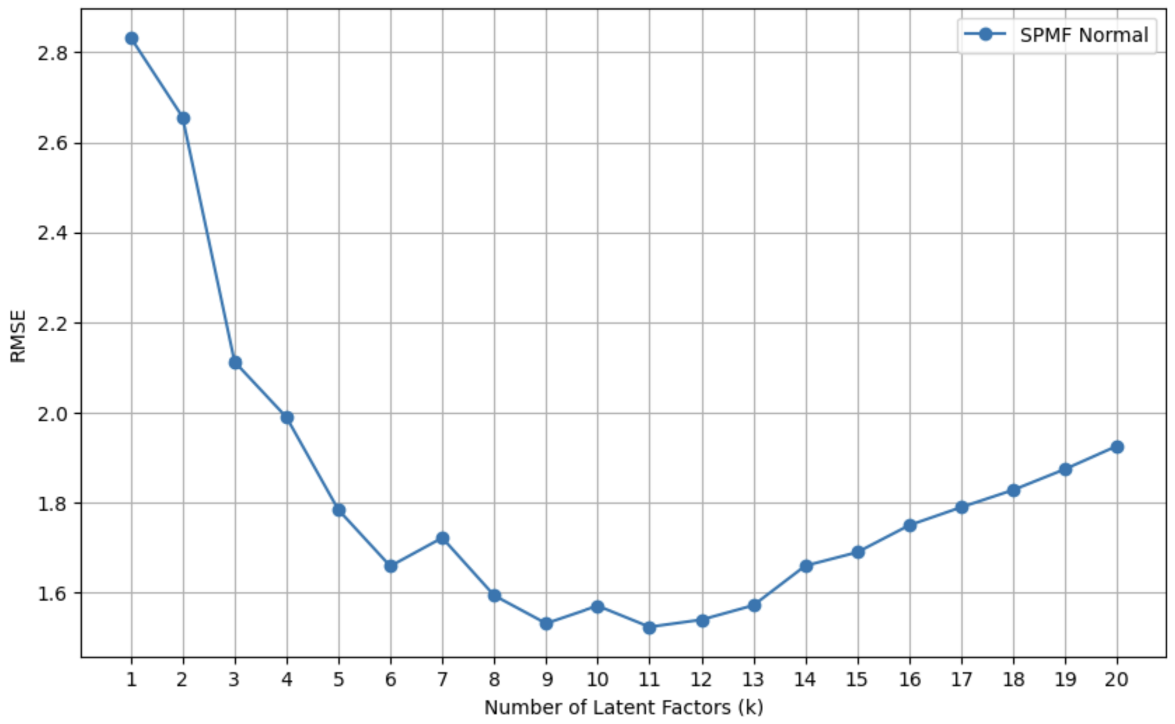

Figure 4.1 RMSE vs Number of Latent Factors - "SPMF Normal" Model for $\alpha = 0.1$ and $T = 100$

SPMF Laplace

In this section, we apply SPMF on the lower triangular part and diagonal matrix $\mathcal{S}$ utilizing the "SPMF Laplace" model. We implement the algorithm specified in Algorithm 3 and utilize the loss function described in Equation 3.20. To partition $\mathcal{S}$ into training and validation sets, as well as for parameter tuning, we follow the procedures outlined in 4.3 and 4.4 respectively.

For the "SPMF Laplace" model, as discussed in 4.4, we conduct an optimization process to determine the combination of α, T, and the number of latent factors k that minimizes the RMSE on the validation set $\mathcal{S}_{\text{val}}$. Through three nested loops—one for each parameter—the optimal hyperparameters resulting in the lowest RMSE=1.54 are $\alpha = 0.1$, $T = 100$, and $k = 10$. Figure 4.2 illustrates how the RMSE varies with the number of latent factors, using $\alpha = 0.1$ and $T = 100$.

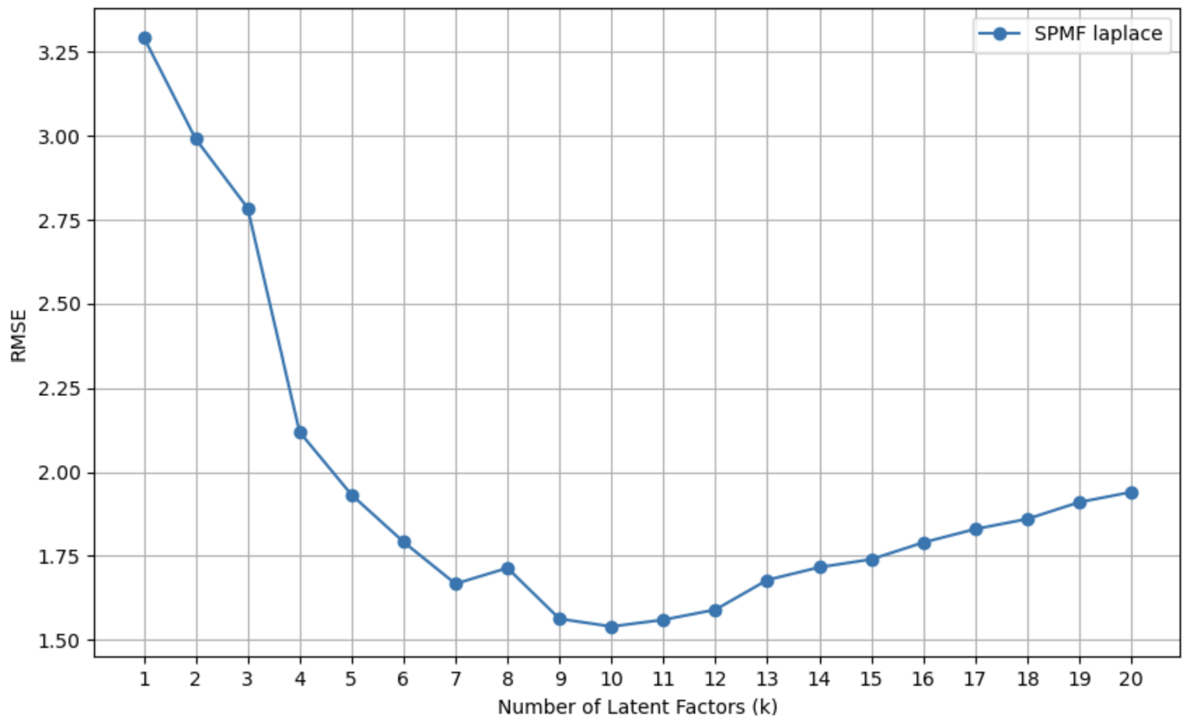

Figure 4.2 RMSE vs Number of Latent Factors - "SPMF Laplace" Model for $\alpha = 0.1$ and $T = 100$

SPMF Bayes

In this section, we employ SPMF on the lower triangular part and diagonal matrix $\mathcal{S}$ using the "SPMF Bayes" model. The algorithm specified in Algorithm 4 is implemented, incorporating the loss function described in Equation 3.28. To partition $\mathcal{S}$ into training and validation sets, as well as for parameter tuning, we adhere to the procedures outlined in 4.3 and 4.4 respectively.

For the "SPMF Bayes" model, as discussed in 4.4, we determine the optimal hyperparameters resulting in the lowest RMSE=1.49, namely $\alpha = 0.1$, $T = 150$, and $k = 12$. Figure 4.3 illustrates how the RMSE varies with the number of latent factors, using $\alpha = 0.1$ and $T = 150$.

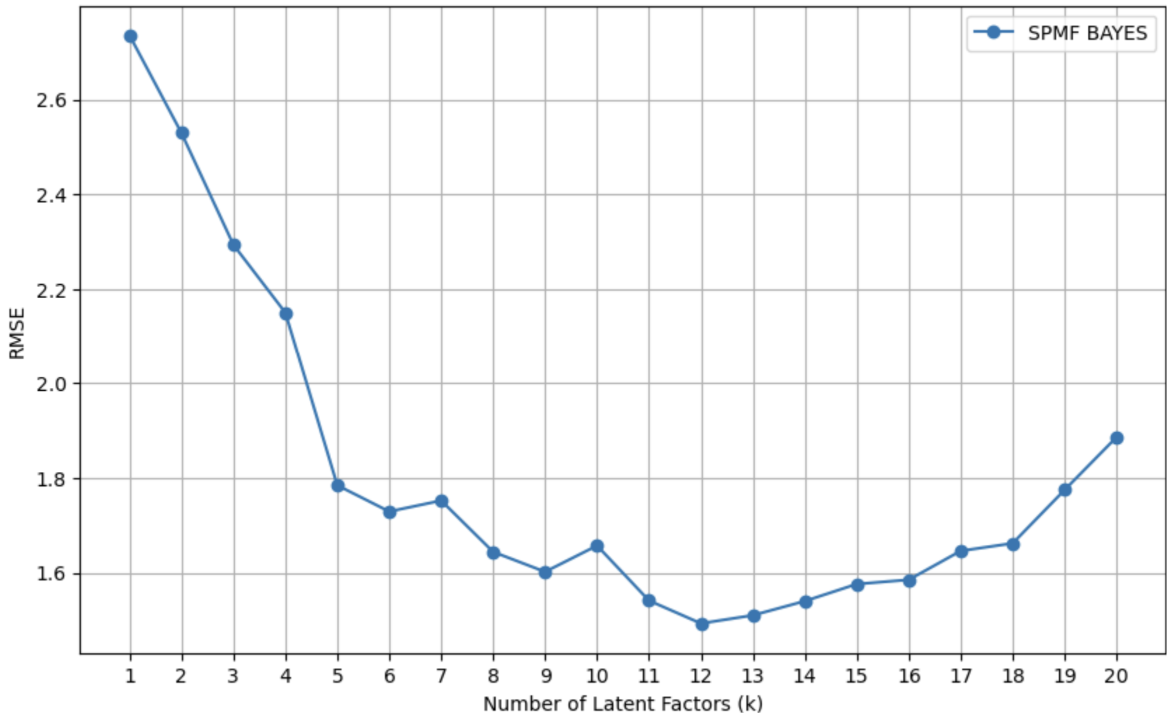

Figure 4.3 RMSE vs Number of Latent Factors - "SPMF Bayes" Model for $\alpha = 0.1$ and $T = 150$

Discussion of Results

One of the primary benefits derived from constructing the matrix $\mathcal{S}$ and subsequently approximating it to obtain $\hat{\mathcal{S}}$ lies in uncovering relationships among user-user, item-item, and user-item interactions. Given that "SPMF Bayes" demonstrated the lowest RMSE compared to the other two SPMF models, we employed "SPMF Bayes" to approximate $\mathcal{S}$, thereby facilitating the prediction of recommendations. To better understand the obtained results, we present the partial user-item rating matrix used for modeling as follows:

$$R = \begin{array}{c} \\ \mathbf{10} \\ \mathbf{136} \\ \mathbf{145} \\ \mathbf{258} \\ \mathbf{270} \\ \mathbf{301} \\ \mathbf{308} \\ \mathbf{474} \\ \mathbf{576} \\ \mathbf{598} \\ \mathbf{699} \\ \mathbf{730} \\ \mathbf{829} \\ \mathbf{896} \\ \mathbf{932} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathbf{123} & \mathbf{124} & \mathbf{280} & \mathbf{311} & \mathbf{541} & \mathbf{792} & \mathbf{898} & \mathbf{1012} \end{array} \\ \begin{pmatrix} \text{NA} & 5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & 5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 1 & 4 \\ \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 \\ 3 & 4 & \text{NA} & \text{NA} & \text{NA} & 3 & \text{NA} & \text{NA} \\ \text{NA} & 5 & \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} & \text{NA} \\ \text{NA} & 4 & 5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} \\ \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 5 \\ \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 3 & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & \text{NA} & \text{NA} & 1 & \text{NA} & \text{NA} & \text{NA} \end{pmatrix} \end{array}. \tag{4.1}$$

After estimating the matrix $\hat{U}$ using the "SPMF Bayes" model and obtaining the approximated matrix $\hat{\mathcal{S}} = \hat{U}\hat{U}^T$, we can derive insights into user-item behavior. As previously mentioned, the lower right block matrix of $\hat{\mathcal{S}}$ ($\hat{R}_{\text{RN}}\hat{R}_{\text{RN}}^T$) represents the user-user similarity matrix. The approximated user-user similarity matrix for some users is presented below.

$$\begin{array}{c} \\ \mathbf{10} \\ \mathbf{136} \\ \mathbf{145} \\ \mathbf{258} \\ \mathbf{270} \\ \mathbf{301} \\ \mathbf{308} \\ \mathbf{474} \\ \mathbf{576} \\ \mathbf{598} \\ \mathbf{699} \\ \mathbf{730} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} \mathbf{10} & \mathbf{136} & \mathbf{145} & \mathbf{258} & \mathbf{270} & \mathbf{301} & \mathbf{308} & \mathbf{474} & \mathbf{576} & \mathbf{598} & \mathbf{699} & \mathbf{730} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & -0.01 & -0.01 & 0.01 & 0.69 & 0.80 & 0.62 & 0 & 1 & 0.02 \\ 1 & 1.05 & 0.01 & 0 & 0.01 & -0.01 & 0.67 & 0.76 & 0.64 & 0.01 & 0.98 & 0.01 \\ 0.01 & 0.01 & 0.99 & 0.02 & 0.68 & 0.97 & 0.34 & -0.04 & -0.03 & 0.17 & 0 & 0.69 \\ -0.01 & 0 & 0.02 & 1 & 0 & 0 & -0.01 & -0.01 & 0 & 0 & -0.01 & 0.02 \\ -0.01 & 0.01 & 0.68 & 0 & 1.08 & 0.69 & 0.45 & 0.03 & 0 & 0.01 & -0.01 & -0.03 \\ 0.01 & -0.01 & 0.97 & 0 & 0.69 & 1.02 & 0.34 & -0.04 & 0.01 & -0.01 & 0.04 & 0.68 \\ 0.69 & 0.67 & 0.34 & -0.01 & 0.45 & 0.34 & 1.02 & 0.75 & 0.39 & 0.01 & 0.69 & -0.04 \\ 0.80 & 0.76 & -0.04 & -0.01 & 0.03 & -0.04 & 0.75 & 1.02 & 0.47 & 0.02 & 0.82 & 0 \\ 0.62 & 0.64 & -0.03 & 0 & 0 & 0.01 & 0.39 & 0.47 & 0.99 & -0.03 & 0.63 & -0.02 \\ 0 & 0.01 & 0.17 & 0 & 0.01 & -0.01 & 0.01 & 0.02 & -0.03 & 1.05 & -0.01 & 0.01 \\ 1 & 0.98 & 0 & -0.01 & -0.01 & 0.04 & 0.69 & 0.82 & 0.63 & -0.01 & 1.07 & -0.01 \\ 0.02 & 0.01 & 0.69 & 0.02 & -0.03 & 0.68 & -0.04 & 0 & -0.02 & 0.01 & -0.01 & 1.05 \end{pmatrix} \end{array}.$$

Additionally, as demonstrated in Section 2.1.4, to obtain the user-user similarity matrix for the matrix R, it is sufficient to calculate $R_{\mathrm{RN}}R_{\mathrm{RN}}^{T}$. This matrix is presented as below:

$$
\begin{array}{c|cccccccccccc}
 & \mathbf{10} & \mathbf{136} & \mathbf{145} & \mathbf{258} & \mathbf{270} & \mathbf{301} & \mathbf{308} & \mathbf{474} & \mathbf{576} & \mathbf{598} & \mathbf{699} & \mathbf{730} \\
\mathbf{10} & 1 & 1 & 0 & 0 & 0 & 0 & 0.69 & 0.78 & 0.62 & 0 & 1 & 0 \\
\mathbf{136} & 1 & 1 & 0 & 0 & 0 & 0 & 0.69 & 0.78 & 0.62 & 0 & 1 & 0 \\
\mathbf{145} & 0 & 0 & 1 & 0 & 0.70 & 0.98 & 0.36 & 0 & 0 & 0.17 & 0 & 0.70 \\
\mathbf{258} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{270} & 0 & 0 & 0.70 & 0 & 1 & 0.71 & 0.51 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{301} & 0 & 0 & 0.98 & 0 & 0.71 & 1 & 0.36 & 0 & 0 & 0 & 0 & 0.71 \\
\mathbf{308} & 0.69 & 0.69 & 0.36 & 0 & 0.51 & 0.36 & 1 & 0.86 & 0.43 & 0 & 0.69 & 0 \\
\mathbf{474} & 0.78 & 0.78 & 0 & 0 & 0 & 0 & 0.86 & 1 & 0.49 & 0 & 0.78 & 0 \\
\mathbf{576} & 0.62 & 0.62 & 0 & 0 & 0 & 0 & 0.43 & 0.49 & 1 & 0 & 0.62 & 0 \\
\mathbf{598} & 0 & 0 & 0.17 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{699} & 1 & 1 & 0 & 0 & 0 & 0 & 0.69 & 0.78 & 0.62 & 0 & 1 & 0 \\
\mathbf{730} & 0 & 0 & 0.70 & 0 & 0 & 0.71 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}.
$$

To visually compare the original user-user similarity matrix with the approximated one, we present both matrices using heatmaps, as illustrated in 4.4.

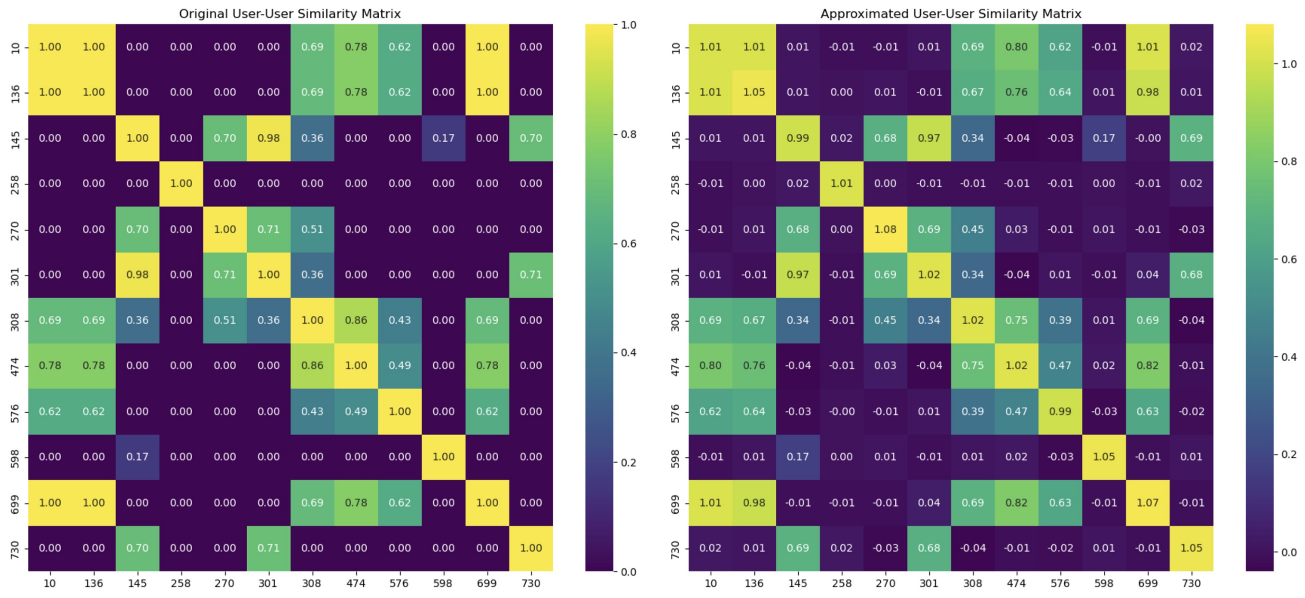

Figure 4.4 User-User Similarity: Heatmap comparison of the original and approximated user-user similarity matrices.

The heatmap comparison of the original and approximated user-user similarity matrices in Figure 4.4 reveals that the general structure of the user-user similarity matrix is preserved in the approximation. This indicates that the "SPMF Bayes" model captures the relationships between users. In both heatmaps, certain user pairs exhibit high similarity values (close to 1), consistently identified in both the original and approximated matrices, suggesting that the approximation retains strong user-user relationships. Additionally, both heatmaps show pairs of users with low or zero similarity values, indicating that the model approximates the lack of relationship between certain users and maintains the integrity of the original data. While some minor variations are observed in the similarity values between the original and approximated matrices, these differences are generally small, underscoring the robustness of the approximation. The heatmaps also reveal clusters of users with higher similarities, indicating user groups with similar preferences. These clusters are visible in both the original and approximated matrices, demonstrating that the approximation successfully retains the clustering patterns within the user data. Overall, the "SPMF Bayes" model provides a reliable approximation of the user-user similarity matrix, capturing the relationships and patterns present in the original data.

Similarly, the top left block matrix of $\hat{\mathcal{S}}$ $(\hat{R}_{\text{CN}}^T \hat{R}_{\text{CN}})$ pertains to the item-item similarity matrix. Below, we present a subset of this matrix for 8 movies.

$$\begin{array}{c|cccccccc}
 & \mathbf{123} & \mathbf{124} & \mathbf{280} & \mathbf{311} & \mathbf{541} & \mathbf{792} & \mathbf{898} & \mathbf{1012} \\
\mathbf{123} & 0.99 & 0.09 & 0.01 & -0.00 & -0.02 & 0.18 & 0.42 & 0.31 \\
\mathbf{124} & 0.09 & 1.06 & 0.21 & -0.02 & 0.01 & 0.39 & -0.01 & -0.03 \\
\mathbf{280} & 0.01 & 0.21 & 1.00 & -0.00 & -0.02 & 0.04 & -0.02 & 0.01 \\
\mathbf{311} & -0.00 & -0.02 & -0.00 & 1.01 & 0.00 & -0.02 & 0.01 & 0.01 \\
\mathbf{541} & -0.02 & 0.01 & -0.02 & 0.00 & 1.04 & 0.01 & 0.02 & -0.03 \\
\mathbf{792} & 0.18 & 0.39 & 0.04 & -0.02 & 0.01 & 1.04 & -0.02 & -0.01 \\
\mathbf{898} & 0.42 & -0.01 & -0.02 & 0.01 & 0.02 & -0.02 & 1.04 & 0.49 \\
\mathbf{1012} & 0.31 & -0.03 & 0.01 & 0.01 & -0.03 & -0.01 & 0.49 & 1.06
\end{array}.$$

Also, as shown in Section 2.1.4, computing $R_{\text{CN}}^T R_{\text{CN}}$ is adequate to derive the item-item similarity matrix for the matrix R. The resulting similarity matrix is shown below.

$$\begin{array}{c|cccccccc}
 & \mathbf{123} & \mathbf{124} & \mathbf{280} & \mathbf{311} & \mathbf{541} & \mathbf{792} & \mathbf{898} & \mathbf{1012} \\
\hline
\mathbf{123} & 1 & 0.22 & 0 & 0 & 0 & 0.21 & 0.46 & 0.41 \\
\mathbf{124} & 0.22 & 1 & 0.32 & 0 & 0.00 & 0.51 & 0 & 0 \\
\mathbf{280} & 0 & 0.32 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{311} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{541} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{792} & 0.21 & 0.51 & 0 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{898} & 0.46 & 0 & 0 & 0 & 0 & 0 & 1 & 0.44 \\
\mathbf{1012} & 0.41 & 0 & 0 & 0 & 0 & 0 & 0.44 & 1
\end{array}.$$

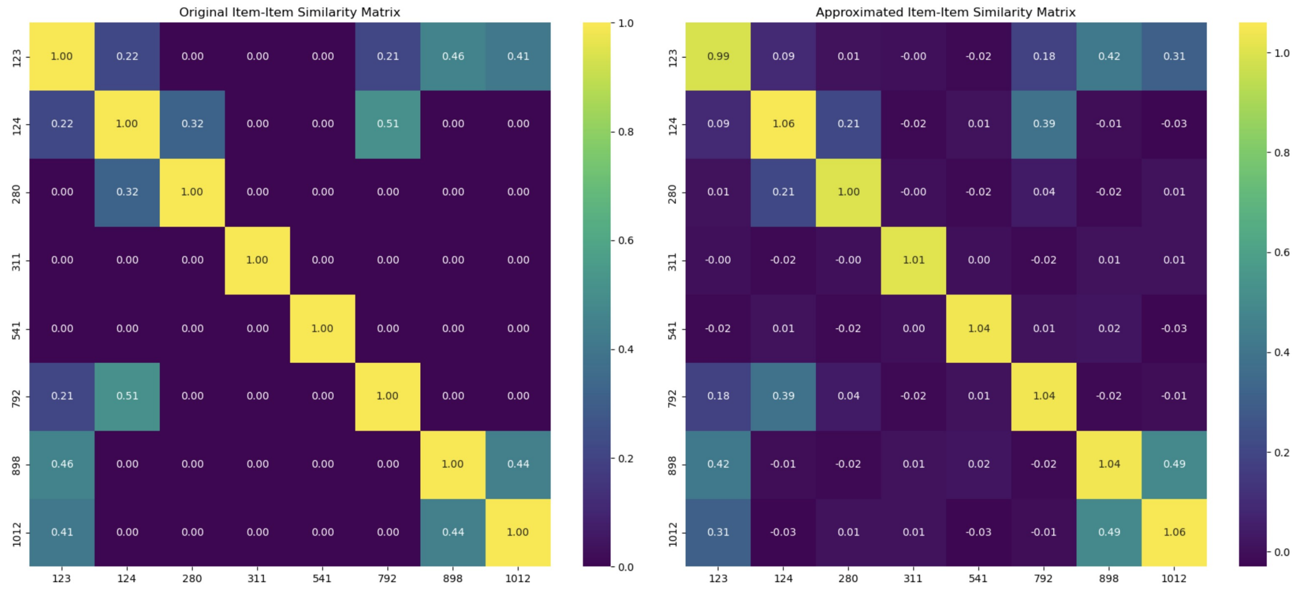

Figure 4.5 Item-Item Similarity: Heatmap comparison of the original and approximated item-item similarity matrices.

The heatmap comparison of the original and approximated item-item similarity matrices, shown in Figure 4.5, offers insights into the accuracy of the "SPMF Bayes" model in preserving item-item relationships. The approximation successfully maintains the overall structure and intricate interactions between items. High similarity values (close to 1) between specific item pairs appear consistently in both the original and approximated matrices, highlighting the model's ability to capture strong item-item connections. Additionally, the presence of low or zero similarity values between item pairs in both heatmaps suggests that the model preserves the distinct separations and unique characteristics of unrelated items, which is essential for maintaining the integrity of the original data.

The approximated heatmap reveals detailed patterns of item relationships, including clusters of items with higher similarity values. These clusters, visible in both the original and approximated matrices, indicate that the approximation retains item grouping patterns. Such patterns are indicative of items that are often rated similarly by users, providing insights for recommendation systems.

Despite minor variations in similarity values between the original and approximated matrices, these differences are minimal, underscoring the robustness and precision of the "SPMF Bayes" model. The model's ability to maintain detailed item relationships and patterns shows its capability in capturing the complex interactions within the dataset. Overall, the "SPMF Bayes" model offers a reliable approximation of the item-item similarity matrix, preserving both the overarching structure and intricate details of item relationships, thereby enhancing the understanding of item interactions and user preferences.

Now, we utilize the Figures 4.4 and 4.5 for the two primary techniques in RS: Collaborative Filtering RS and Content-Based RS, which were discussed in Chapter 2.

Collaborative Filtering RS

As explained in Chapter 2, Collaborative Filtering (CF) represents a technique within RS, designed to predict user preferences for items based on ratings provided by other users with similar tastes. This method hinges on identifying patterns of similarity among users' preferences to infer the potential likings of a target user for unrated items.

Figure 4.4 highlights that User 730 shares taste similarities with User 301. When examining the user-item rating matrix R in 4.1, we observe that User 730 has not rated movie ID 123, whereas User 301 has given it a high rating of 4. By applying Collaborative Filtering techniques, we can recommend this unrated movie to User 730, leveraging the preferences of similar users.

Furthermore, the heatmap also shows that Users 474 and 136 have some shared tastes. User 474 has rated movie 792 with a high score of 4, while User 136 has not rated this movie according to the user-item rating matrix R in 4.1. Therefore, based on Collaborative Filtering, we recommend this movie to User 136.

Content-Based RS

As described in Chapter 2, in a content-based RS, users receive item suggestions based on their preferences and the attributes of those items. For instance, when a user expresses interest in a particular movie, the system recommends similar movies that share thematic or characteristic parallels. According to Figure 4.5, movie ID 124 shares similarities with movie ID 792. From the user-item rating matrix R, we observe that User 10 has rated movie ID 124 but has not rated movie ID 792. Therefore, using the Content-Based RS approach, we can recommend movie ID 792 to User 10. Similarly, movie IDs 898 and 123 exhibit similarities, and User 598 has shown interest in movie ID 898 but has not rated movie ID 123. Consequently, we recommend movie ID 123 to User 598 based on the Content-Based RS methodology.

Predicting Missing Ratings

In the context of the matrix $\hat{\mathcal{S}}$, the lower left block, denoted as $\hat{R}_{RN}$, plays a crucial role in predicting ratings for users who have not provided ratings for specific items (indicated as NA entries). Following the approximation of the matrix $\mathcal{S}$, the reconstructed matrix $\hat{\mathcal{S}}$ furnishes these missing values, thereby facilitating predictive insights.

Examining the user-item rating matrix R in 4.1, we observe that User 301 has not rated movie ID 898. Consulting $\hat{R}_{RN}$, the predicted rating for this user-movie pair emerges as 1.23 (post-transformation). Similarly, for User 474, who has not rated movie ID 792, the predicted rating after transformation is 4.37.

This predictive mechanism highlights the utility of $\hat{R}_{RN}$ in filling in missing ratings by providing estimated values for user-item pairs previously devoid of ratings. Through matrix approximation and subsequent prediction, this methodology empowers the derivation of likely ratings for unexplored user-item interactions within the RS framework.

After detailing movie recommendation predictions with the "SPMF Bayes" model, the subsequent section explores a comparison of convergence rates for the "SPMF Bayes" model versus SVD, NMF, and PMF.

Convergence Rate Analysis of SPMF vs SVD, NMF, and PMF

Given that "SPMF Bayes" has demonstrated the lowest RMSE compared to the other two SPMF models, we focus on comparing the convergence rate of "SPMF Bayes" with SVD, NMF, and PMF using the same sample data. For each model, we first determine the optimal number of latent factors (with "SPMF Bayes" having a best $k = 12$), and then iterate over 100 iterations to assess convergence. Figure 4.6 presents the RMSE values over 100 iterations for SVD, NMF, PMF, and "SPMF Bayes", revealing insights into their convergence rates and prediction accuracies.

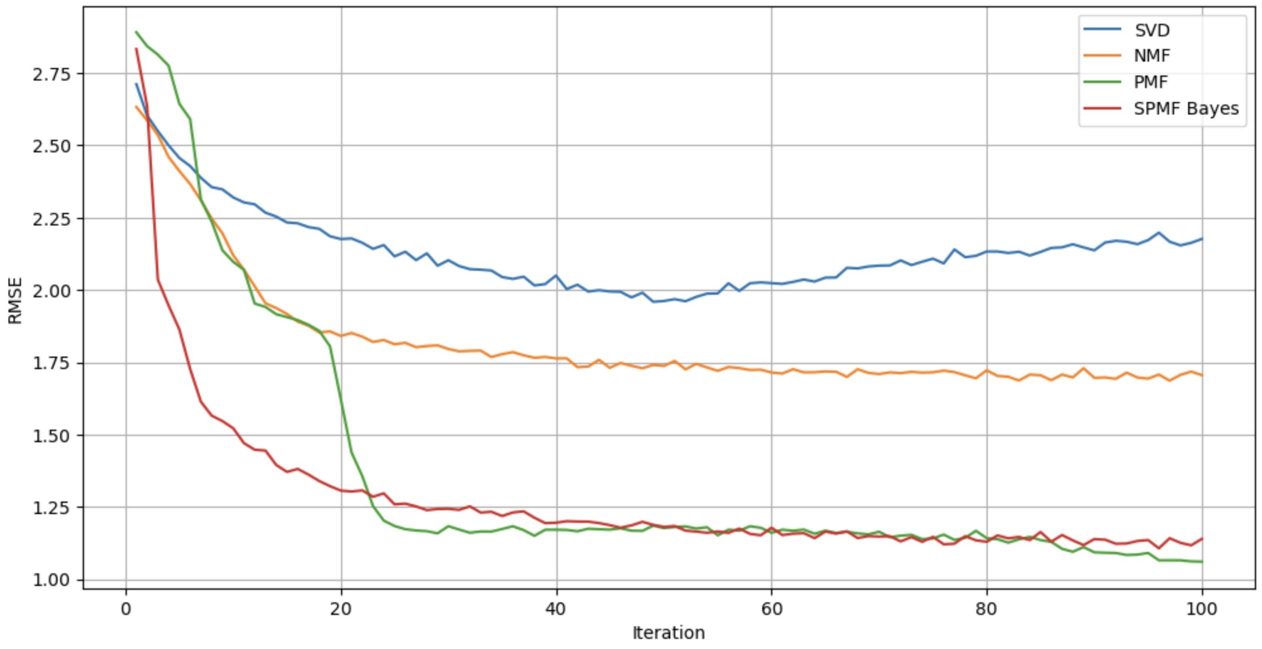

Figure 4.6 Comparison of RMSE values over 100 iterations for SVD, NMF, PMF, and "SPMF Bayes". The plot illustrates the convergence rates of each model, highlighting the faster convergence achieved by "SPMF Bayes" compared to the other models.

"SPMF Bayes" demonstrates a clear advantage in convergence speed compared to the other models. The RMSE values for "SPMF Bayes" drop rapidly in the initial iterations and continue to decrease steadily. By the 20th iteration, "SPMF Bayes" achieves an RMSE of around 1.2, which remains stable with slight fluctuations, showcasing its ability to maintain low error rates.

PMF, while also showing a downward trend in RMSE values, converges more slowly than

"SPMF Bayes". The RMSE values decrease significantly in the first few iterations but then gradually reduce, stabilizing around 1.2 after 60 iterations. Although PMF achieves similar RMSE values to "SPMF Bayes", it requires more iterations to reach comparable levels of accuracy.

NMF and SVD, on the other hand, consistently exhibit higher RMSE values throughout the iterations. NMF shows a steady decline in RMSE but stabilizes at a higher error rate compared to PMF and "SPMF Bayes". SVD, despite its continuous decrease in RMSE, converges to a higher error rate than the other models, indicating its lower prediction accuracy and higher instability.

In the next section, we evaluate the performance of the SPMF models against several state-of-the-art models, including SVD, NMF, and PMF.

Evaluation of SPMF Model Performance on Subsets of MovieLens 100K Dataset

Now, we randomly selected 20 samples from the MovieLens 100K dataset, each consisting of 50 users and 20 items, to evaluate the performance of state-of-the-art models (SVD, NMF, and PMF) against our three proposed SPMF models ("SPMF Normal", "SPMF Laplace", and "SPMF Bayes"). For each model, we executed their respective algorithms to identify the optimal number of latent factors k. Subsequently, we performed matrix factorization using this optimal k, iterating 100 times to record the lowest RMSE for each sample and model. The RMSE values for each model and sample are illustrated in Figure 4.7.

The RMSE values across different models reveal distinct patterns in prediction accuracy. SVD and NMF consistently exhibit higher RMSE values across all samples, indicating lower prediction accuracy. Specifically, SVD displays variability, with some samples showing higher RMSE values, reflecting its instability. Although NMF performs better than SVD, it still falls short compared to other models in terms of prediction accuracy.

Conversely, PMF demonstrates the lowest RMSE values across most samples, although not consistently for all samples. The SPMF models, including "SPMF Bayes", "SPMF Normal", and "SPMF Laplace", show competitive RMSE values that are generally lower than those of SVD and NMF. While they have slightly higher RMSE values compared to PMF in some samples, they maintain a high level of accuracy overall. The performance of the SPMF models highlights their ability to provide accurate predictions while also being interpretable, thus fulfilling our research objective.

The primary goal of introducing the SPMF models was not solely to achieve the most accurate predictions, but rather to introduce a model that balances interpretability and accuracy. By doing so, we aim to benchmark the performance of SPMF models against other state-of-the-art models. This analysis underscores the value of SPMF models in the broader context of recommendation systems, where both accuracy and interpretability are crucial for practical applications.

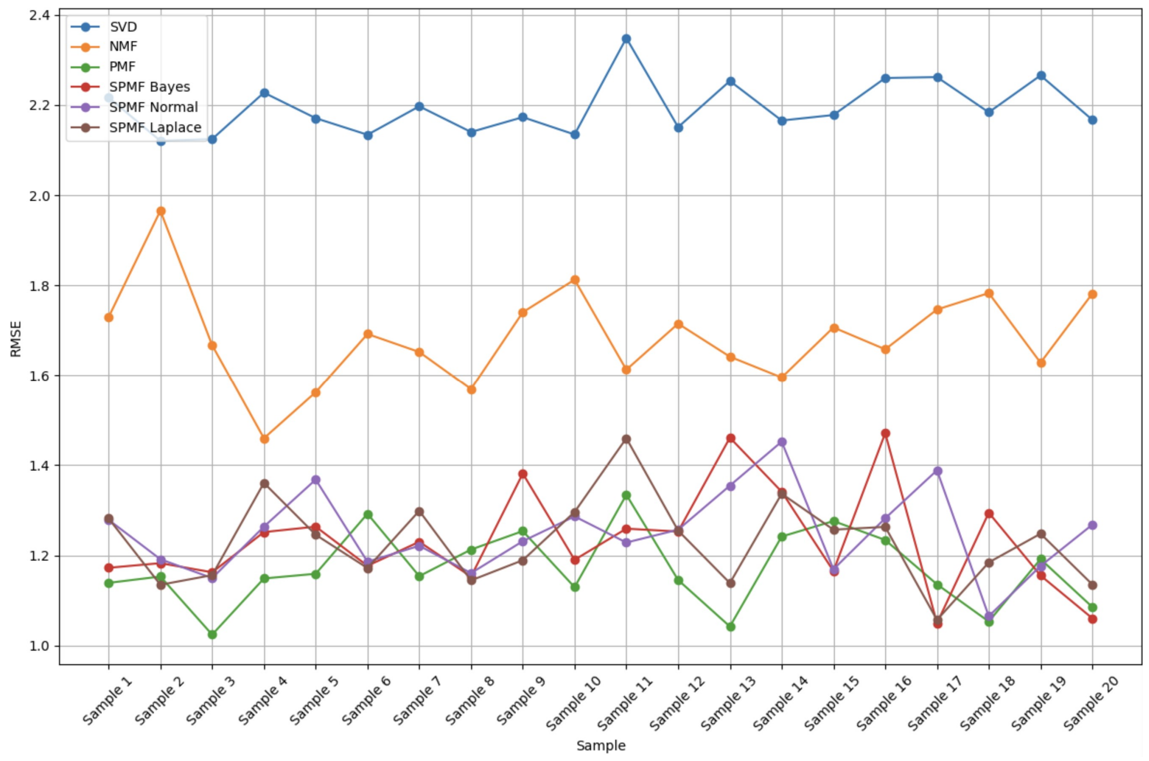

Figure 4.7 Comparison of RMSE values for 20 samples from the MovieLens 100K dataset across different models (SVD, NMF, PMF, "SPMF Bayes", "SPMF Normal", and "SPMF Laplace").

In the next section, we replicate the analysis conducted for the MovieLens 100K dataset, this time applying it to the FilmTrust dataset.

4.6 Experiment Results Using the FilmTrust Dataset

In this section, we apply the three SPMF models to the FilmTrust dataset, following a similar preprocessing methodology as outlined in Section 4.2 and 4.3. The procedures and methodologies are consistent with the approach taken for the MovieLens 100K dataset.

First, we employ the three SPMF variants on the preprocessed FilmTrust data. As the process mirrors that of the MovieLens 100K dataset sample, we will not reiterate all the details. In Table 4.1, we provide all the parameters along with the lowest RMSE achieved for each variant.

Table 4.1 Comparison of SPMF Variants

Model	α	#Iterations	Best k	RMSE
SPMF Bayes	0.10	100	8	1.132
SPMF Normal	0.01	100	10	1.145
SPMF Laplace	0.01	100	11	1.164

Figure 4.8 illustrates how the RMSE varies with the number of latent factors, using α and T from Table 4.1.

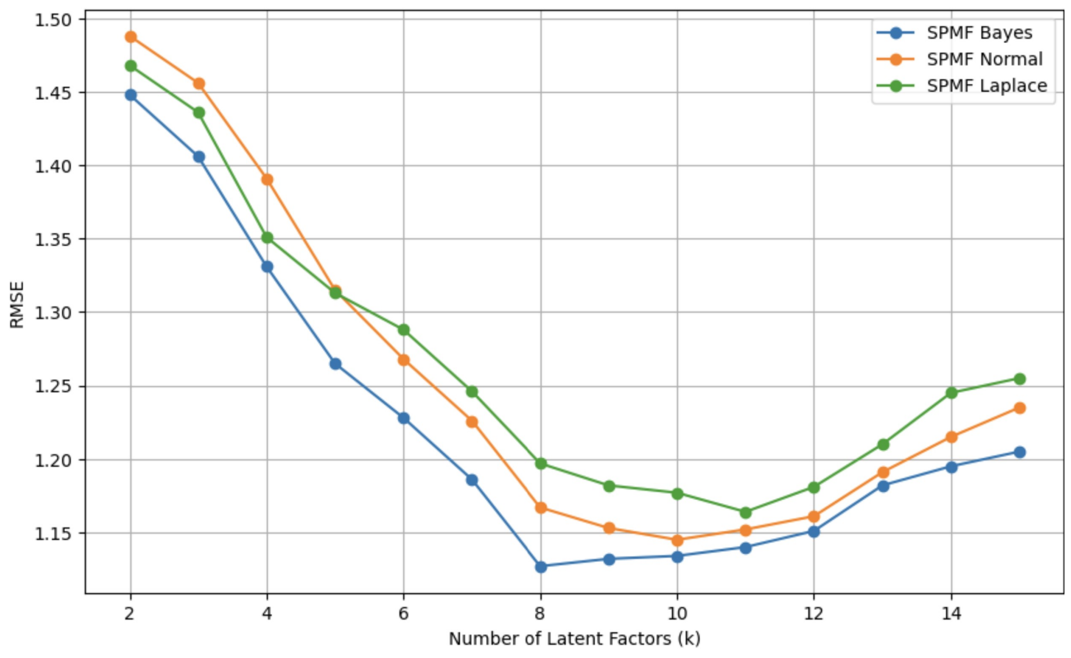

Figure 4.8 RMSE vs Number of Latent Factors for "SPMF Normal", "SPMF Laplace", and "SPMF Bayes".

Given that "SPMF Bayes" showed the lowest RMSE comapred to the other SPMF models, we employed "SPMF Bayes" to approximate $\mathcal{S}$.

Discussion Of Results

In this section, we utilized "SPMF Bayes" to approximate $\mathcal{S}$, allowing us to derive user-user and item-item similarity approximations, as well as predict recommendations. To provide context for our findings, we present the partial user-item rating matrix used for modeling below:

$$R = \begin{array}{c} \\ \mathbf{30} \\ \mathbf{68} \\ \mathbf{124} \\ \mathbf{142} \\ \mathbf{176} \\ \mathbf{245} \\ \mathbf{272} \\ \mathbf{276} \\ \mathbf{304} \\ \mathbf{323} \\ \mathbf{412} \\ \mathbf{571} \\ \mathbf{606} \\ \mathbf{652} \\ \mathbf{673} \\ \mathbf{706} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathbf{187} & \mathbf{212} & \mathbf{480} & \mathbf{754} & \mathbf{875} & \mathbf{961} & \mathbf{986} & \mathbf{1029} \end{array} \\ \begin{pmatrix} \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} \\ \text{NA} & 4 & \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & 4 \\ \text{NA} & 2 & \text{NA} & \text{NA} & 4 & 4 & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & 1 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & 3 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & 2.5 & 3.5 & \text{NA} & 2.5 & 3.5 & \text{NA} \\ \text{NA} & 2 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & \text{NA} & 4 & 1.5 & \text{NA} & \text{NA} & 3.5 \\ \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} \\ \text{NA} & 3.5 & 4 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 & \text{NA} & \text{NA} \\ \text{NA} & 2.5 & \text{NA} & 3.5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} \\ 4 & 1.5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & 4 \\ \text{NA} & 3.5 & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} & \text{NA} \end{pmatrix} \end{array} . \tag{4.2}$$

After estimating the matrix $\hat{U}$ with the "SPMF Bayes" model and subsequently deriving the approximated matrix $\hat{\mathcal{S}} = \hat{U}\hat{U}^T$, we can gain insights into user-item interactions. The lower right block of $\hat{\mathcal{S}}$, specifically $\hat{R}_{\text{RN}}\hat{R}_{\text{RN}}^T$, corresponds to the user-user similarity matrix. Below, we present the approximated user-user similarity matrix for a subset of users.

$$
\begin{array}{r|cccccccccccccccc}
 & \mathbf{30} & \mathbf{68} & \mathbf{124} & \mathbf{142} & \mathbf{176} & \mathbf{245} & \mathbf{272} & \mathbf{276} & \mathbf{304} & \mathbf{323} & \mathbf{412} & \mathbf{571} & \mathbf{606} & \mathbf{652} & \mathbf{673} & \mathbf{706} \\
\hline
\mathbf{30} & 0.99 & 0.39 & 0.27 & -0.02 & 0.77 & 0.72 & 0.42 & 0.66 & 0.02 & 0.03 & 0.42 & 0.44 & -0.01 & 0.42 & 0.19 & 0.54 \\
\mathbf{68} & 0.39 & 1.01 & 0.15 & -0.01 & 0.55 & 0.54 & 0.65 & 0.54 & 0.01 & 0.82 & 0.03 & 0.39 & 0.04 & 0.84 & 0.66 & 0.62 \\
\mathbf{124} & 0.27 & 0.15 & 1.04 & -0.01 & 0.38 & 0.32 & 0.37 & 0.34 & 0.01 & 0.32 & 0.02 & 0.29 & 0.78 & 0.14 & 0.06 & 0.24 \\
\mathbf{142} & -0.02 & -0.01 & -0.01 & 0.89 & 0.01 & 0.02 & 0.51 & -0.01 & -0.02 & 0.01 & 0.04 & 0.74 & 0.02 & 0.02 & -0.01 & 0.03 \\
\mathbf{176} & 0.77 & 0.55 & 0.38 & 0.01 & 0.95 & 1.02 & 0.01 & 1.05 & 0.02 & 0.04 & 0.03 & 0.68 & 0.02 & 0.055 & 0.22 & 0.99 \\
\mathbf{245} & 0.72 & 0.54 & 0.32 & 0.02 & 1.02 & 0.105 & 0.02 & 1.06 & -0.02 & -0.01 & -0.01 & 0.55 & 0.02 & 0.66 & 0.25 & 0.89 \\
\mathbf{272} & 0.42 & 0.65 & 0.37 & 0.51 & 0.01 & 0.02 & 0.98 & 0.02 & 0.01 & 0.66 & 0.55 & 0.36 & 0.44 & 0.39 & 0.35 & 0.02 \\
\mathbf{276} & 0.66 & 0.54 & 0.34 & -0.01 & 1.05 & 1.06 & 0.02 & 0.99 & 0.02 & 0.01 & 0.03 & 0.69 & -0.01 & 0.052 & 0.22 & 0.92 \\
\mathbf{304} & 0.02 & 0.01 & 0.01 & -0.02 & 0.02 & -0.02 & 0.01 & 0.02 & 1.03 & 0.01 & 0.02 & 0.01 & -0.01 & -0.02 & 0.66 & 0.03 \\
\mathbf{323} & 0.03 & 0.82 & 0.32 & 0.01 & 0.04 & -0.01 & 0.66 & 0.01 & 0.01 & 1.04 & 0.02 & 0.01 & 0.03 & 0.55 & 0.44 & -0.01 \\
\mathbf{412} & 0.42 & 0.03 & 0.02 & 0.04 & 0.03 & -0.01 & 0.55 & 0.03 & 0.02 & 0.02 & 0.97 & 0.02 & 0.01 & 0.01 & 0.02 & -0.01 \\
\mathbf{571} & 0.44 & 0.39 & 0.29 & 0.74 & 0.68 & 0.55 & 0.36 & 0.69 & 0.01 & 0.01 & 0.02 & 0.99 & 0.01 & 0.24 & 0.31 & 0.59 \\
\mathbf{606} & -0.01 & 0.04 & 0.78 & 0.02 & 0.02 & 0.02 & 0.44 & -0.01 & -0.01 & 0.03 & 0.01 & 0.01 & 0.88 & 0.01 & 0.01 & 0.01 \\
\mathbf{652} & 0.42 & 0.84 & 0.14 & 0.02 & 0.055 & 0.66 & 0.39 & 0.052 & -0.02 & 0.55 & 0.01 & 0.24 & 0.01 & 0.95 & 0.19 & 0.66 \\
\mathbf{673} & 0.19 & 0.66 & 0.06 & -0.01 & 0.22 & 0.25 & 0.35 & 0.22 & 0.66 & 0.44 & 0.02 & 0.31 & 0.01 & 0.19 & 1.01 & 0.39 \\
\mathbf{706} & 0.54 & 0.62 & 0.24 & 0.03 & 0.99 & 0.89 & 0.02 & 0.92 & 0.03 & -0.01 & -0.01 & 0.59 & 0.01 & 0.66 & 0.39 & 1.02
\end{array}.
$$

Furthermore, as illustrated in Section 2.1.4, computing the user-user similarity matrix for matrix R can be achieved by calculating $R_{\text{RN}}R_{\text{RN}}^T$. The resulting matrix is displayed below:

$$
\begin{array}{r|cccccccccccccccc}
 & \mathbf{30} & \mathbf{68} & \mathbf{124} & \mathbf{142} & \mathbf{176} & \mathbf{245} & \mathbf{272} & \mathbf{276} & \mathbf{304} & \mathbf{323} & \mathbf{412} & \mathbf{571} & \mathbf{606} & \mathbf{652} & \mathbf{673} & \mathbf{706} \\
\hline
\mathbf{30} & 1 & 0.41 & 0.24 & 0 & 0.71 & 0.71 & 0.41 & 0.71 & 0 & 0 & 0.71 & 0.47 & 0 & 0.41 & 0.16 & 0.71 \\
\mathbf{68} & 0.41 & 1 & 0.18 & 0 & 0.53 & 0.53 & 0.59 & 0.53 & 0 & 0.80 & 0 & 0.35 & 0 & 0.74 & 0.62 & 0.53 \\
\mathbf{124} & 0.24 & 0.18 & 1 & 0 & 0.33 & 0.33 & 0.24 & 0.33 & 0 & 0.18 & 0 & 0.22 & 0.67 & 0.19 & 0.08 & 0.33 \\
\mathbf{142} & 0 & 0 & 0 & 1 & 0 & 0 & 0.36 & 0 & 0 & 0 & 0 & 0.75 & 0 & 0 & 0 & 0 \\
\mathbf{176} & 0.71 & 0.53 & 0.33 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0 & 0.58 & 0.23 & 1 \\
\mathbf{245} & 0.71 & 0.53 & 0.33 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0 & 0.58 & 0.23 & 1 \\
\mathbf{272} & 0.41 & 0.59 & 0.24 & 0.36 & 0 & 0 & 1 & 0 & 0 & 0.68 & 0.50 & 0.27 & 0.41 & 0.41 & 0.38 & 0 \\
\mathbf{276} & 0.71 & 0.53 & 0.33 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0 & 0.58 & 0.23 & 1 \\
\mathbf{304} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.61 & 0 \\
\mathbf{323} & 0 & 0.80 & 0.18 & 0 & 0 & 0 & 0.68 & 0 & 0 & 1 & 0 & 0 & 0 & 0.59 & 0.48 & 0 \\
\mathbf{412} & 0.71 & 0 & 0 & 0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{571} & 0.47 & 0.35 & 0.22 & 0.75 & 0.66 & 0.66 & 0.27 & 0.66 & 0 & 0 & 0 & 1 & 0 & 0.38 & 0.15 & 0.66 \\
\mathbf{606} & 0 & 0 & 0.67 & 0 & 0 & 0 & 0.41 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{652} & 0.41 & 0.74 & 0.19 & 0 & 0.58 & 0.58 & 0.41 & 0.58 & 0 & 0 & 0 & 0.38 & 0 & 1 & 0.13 & 0.58 \\
\mathbf{673} & 0.16 & 0.62 & 0.08 & 0 & 0.23 & 0.23 & 0.38 & 0.23 & 0.61 & 0.48 & 0 & 0.15 & 0 & 0.13 & 1 & 0.23 \\
\mathbf{706} & 0.71 & 0.53 & 0.33 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0 & 0.58 & 0.23 & 1
\end{array}.
$$

To provide a visual comparison between the original and approximated user-user similarity matrices, we have depicted both matrices using heatmaps, as shown in Figure 4.9.

The heatmap comparison of the original and approximated user-user similarity matrices offers insights into the accuracy of the "SPMF Bayes" model in capturing and preserving user relationships. The original user-user similarity matrix displays a range of similarity values, with some user pairs showing very high similarity (close to 1), indicating strong relationships between these users. These robust connections are crucial for accurately predicting user pref-

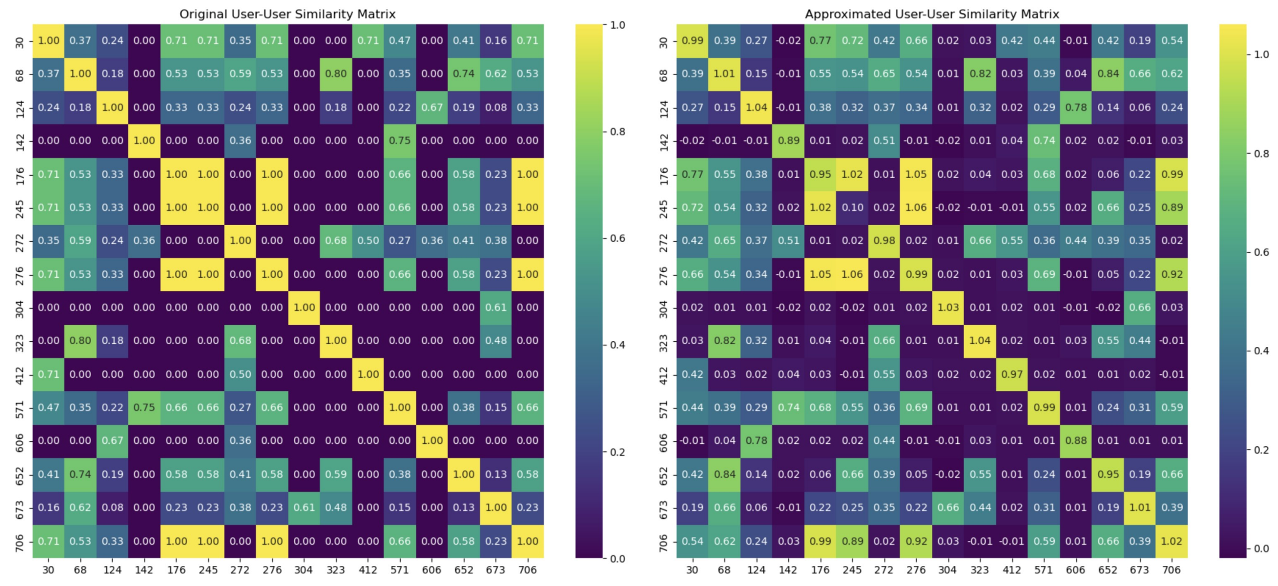

Figure 4.9 User-User Similarity: Heatmap comparison of the original and approximated user-user similarity matrices.

erences and generating reliable recommendations.

In the approximated user-user similarity matrix, the general structure and prominent relationships observed in the original matrix are well-preserved. High similarity values between user pairs are consistently captured in the approximation, indicating that the "SPMF Bayes" model maintains the strong user-user relationships present in the original data. Furthermore, the approximated matrix accurately captures the low or zero similarity values observed in the original matrix. This preservation of low similarity values is essential for distinguishing unrelated users and maintaining the integrity of the original data.

The heatmaps reveal clusters of users with higher similarities, suggesting groups of users with similar preferences. These clusters are visible in both the original and approximated matrices, showing that the "SPMF Bayes" model retains the clustering patterns within the user data. While minor variations in similarity values between the original and approximated matrices are observed, these differences are generally small, underscoring the robustness of the approximation. The overall structure, strong relationships, and clustering patterns are preserved, highlighting the model's capability in maintaining the intricate details of user relationships.

Similarly, the top left block of $\hat{\mathcal{S}}$ ($\hat{R}_{\text{CN}}^T \hat{R}_{\text{CN}}$) represents the item-item similarity matrix. Presented below is a subset of this matrix for 8 selected movies:

$$\begin{array}{c|cccccccc}
 & \mathbf{187} & \mathbf{212} & \mathbf{480} & \mathbf{754} & \mathbf{875} & \mathbf{961} & \mathbf{986} & \mathbf{1029} \\
\hline
\mathbf{187} & 0.99 & 0.25 & 0.02 & 0.01 & -0.01 & 0.04 & 0.01 & 0.33 \\
\mathbf{212} & 0.25 & 1.02 & 0.35 & 0.19 & 0.36 & 0.18 & 0.29 & 0.31 \\
\mathbf{480} & 0.02 & 0.35 & 1.03 & 0.32 & 0.01 & 0.18 & 0.25 & 0.01 \\
\mathbf{754} & 0.01 & 0.19 & 0.32 & 0.99 & 0.33 & 0.12 & 0.41 & 0.71 \\
\mathbf{875} & -0.01 & 0.36 & 0.01 & 0.33 & 0.88 & 0.66 & 0.03 & 0.19 \\
\mathbf{961} & 0.04 & 0.18 & 0.18 & 0.12 & 0.66 & 0.96 & 0.25 & -0.01 \\
\mathbf{986} & 0.01 & 0.29 & 0.25 & 0.41 & 0.03 & 0.25 & 1.01 & -0.02 \\
\mathbf{1029} & 0.33 & 0.31 & 0.01 & 0.71 & 0.19 & -0.01 & -0.02 & 0.97
\end{array}.$$

Additionally, as demonstrated in Section 2.1.4, calculating $R_{\text{CN}}^T R_{\text{CN}}$ suffices to obtain the item-item similarity matrix for the matrix R. The resulting similarity matrix is presented below:

$$\begin{array}{c|cccccccc}
 & \mathbf{187} & \mathbf{212} & \mathbf{480} & \mathbf{754} & \mathbf{875} & \mathbf{961} & \mathbf{986} & \mathbf{1029} \\
\hline
\mathbf{187} & 1.00 & 0.27 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.43 \\
\mathbf{212} & 0.27 & 1.00 & 0.25 & 0.36 & 0.21 & 0.14 & 0.26 & 0.36 \\
\mathbf{480} & 0.00 & 0.25 & 1.00 & 0.19 & 0.00 & 0.16 & 0.21 & 0.00 \\
\mathbf{754} & 0.00 & 0.36 & 0.19 & 1.00 & 0.19 & 0.19 & 0.24 & 0.60 \\
\mathbf{875} & 0.00 & 0.21 & 0.00 & 0.19 & 1.00 & 0.61 & 0.00 & 0.18 \\
\mathbf{961} & 0.00 & 0.14 & 0.16 & 0.19 & 0.61 & 1.00 & 0.21 & 0.00 \\
\mathbf{986} & 0.00 & 0.26 & 0.21 & 0.24 & 0.00 & 0.21 & 1.00 & 0.00 \\
\mathbf{1029} & 0.43 & 0.36 & 0.00 & 0.60 & 0.18 & 0.00 & 0.00 & 1.00
\end{array}.$$

To facilitate a clearer comparison between the original and approximated item-item similarity matrices, we present the corresponding heatmaps in Figure 4.10.

The heatmap comparison of the original and approximated item-item similarity matrices provides insights into how the "SPMF Bayes" model preserves item relationships. The original item-item similarity matrix shows a wide range of similarity values, with certain item pairs displaying high similarity values, indicating strong relationships. These connections are essential for predicting item preferences and generating accurate recommendations.

The approximated item-item similarity matrix retains the overall structure and key relationships seen in the original matrix. High similarity values between item pairs are consistently mirrored in the approximated matrix, indicating that the "SPMF Bayes" model captures the strong item-item connections present in the data. Additionally, the approximated matrix accurately reflects the low or zero similarity values found in the original matrix. This ability to maintain low similarity values is crucial for distinguishing unrelated items and ensuring the integrity of the original data is preserved.

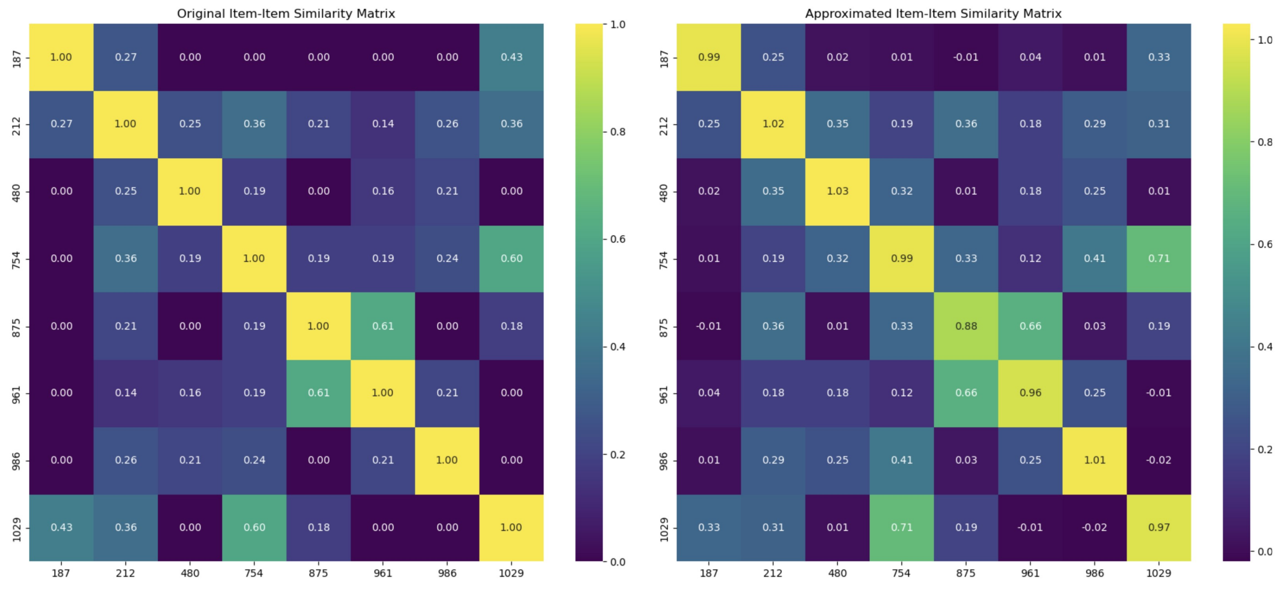

Figure 4.10 Item-Item Similarity: Heatmap comparison of the original and approximated item-item similarity matrices.

The heatmaps also highlight clusters of items with higher similarity values, suggesting groups of items that share similar user preferences. These clusters are apparent in both the original and approximated matrices, demonstrating that the "SPMF Bayes" model retains the inherent clustering patterns of the item data. While minor variations in the similarity values between the original and approximated matrices are observed, these differences are generally small, underscoring the robustness of the approximation. The model's ability to preserve the overall structure, key relationships, and clustering patterns underscores its capability in maintaining detailed item interactions.

In conclusion, the "SPMF Bayes" model approximates the item-item similarity matrix well, preserving the essential relationships and patterns within the original data. This preservation is crucial for understanding item interactions and improving the accuracy of recommendation systems.

Now, we refer to Figures 4.9 and 4.10 to illustrate the application of the two main techniques in recommender systems: Collaborative Filtering RS and Content-Based RS.

Collaborative Filtering RS

Figure 4.9 illustrates that User 706 shares taste similarities with User 571. When examining the user-item rating matrix R in 4.2, it becomes evident that User 571 has not rated movie ID 480, while User 706 has given it a high rating of 4. By applying Collaborative Filtering techniques, this unrated movie can be recommended to User 571, leveraging the preferences of similar users.

Additionally, the heatmap reveals that Users 304 and 673 exhibit some shared tastes. User 673 has rated movie ID 1029 with a high score of 4, while User 304 has not rated this movie according to the user-item rating matrix R in 4.2. Therefore, based on Collaborative Filtering, it is advisable to recommend this movie to User 304.

Content-Based RS

Based on Figure 4.10, movie ID 1029 has similarities with movie ID 754. From the user-item rating matrix R in 4.2, we observe that User 673 has rated movie ID 1029 but has not rated movie ID 754. Therefore, using the Content-Based RS approach, movie ID 754 can be recommended to User 673. Similarly, movies ID 875 and 961 show similarities. Since User 606 has rated movie ID 961 but not movie ID 875, we can recommend movie ID 875 to User 606 following the Content-Based RS methodology.

Predicting Missing Ratings

In the context of the matrix $\hat{\mathcal{S}}$, the lower left block, denoted as $\hat{R}_{RN}$, is essential for predicting ratings for users who have not rated specific items (indicated by NA entries). After approximating the matrix $\mathcal{S}$, the reconstructed matrix $\hat{\mathcal{S}}$ fills these missing values, thereby providing predictive insights.

Considering the user-item rating matrix R in 4.2, we notice that User 68 has not rated movie ID 986. By consulting $\hat{R}_{RN}$, the predicted rating for this user-movie pair is 3.86 (post-transformation). Similarly, for User 606, who has not rated movie ID 857, the predicted rating after transformation is 3.31.

This predictive mechanism demonstrates the utility of $\hat{R}_{RN}$ in estimating missing ratings by providing predicted values for user-item pairs that lack ratings. Through matrix approximation and subsequent prediction, this approach enables the derivation of probable ratings for unexplored user-item interactions within the RS framework.

Following the detailed predictions for movie recommendations using the "SPMF Bayes" model, the next section delves into a comparative analysis of convergence rates between the "SPMF Bayes" model and other methods such as SVD, NMF, and PMF.

Convergence Rate Analysis of SPMF vs SVD, NMF, and PMF

The RMSE values for each iteration from 1 to 100 were recorded for SPMF, SVD, NMF, and PMF. The results were plotted to visualize the convergence behavior of each method in Figure 4.11.

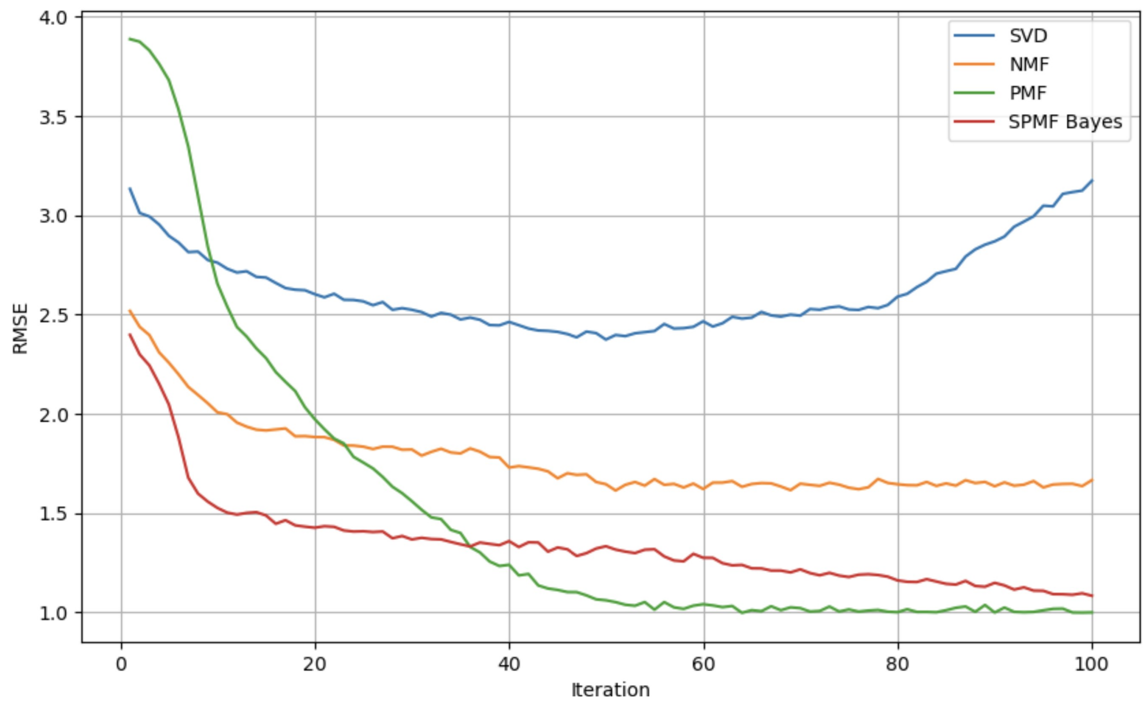

Figure 4.11 Comparison of RMSE values over 100 iterations for SVD, NMF, PMF, and SPMF Bayes. The plot illustrates the convergence rates of each model, highlighting the faster convergence achieved by "SPMF Bayes" compared to the other models.

PMF starts with an RMSE of 3.89, indicating the highest initial error among the methods. In contrast, SPMF begins with an RMSE of 2.40, lower than PMF's initial value, suggesting a better initial fit. NMF and SVD start with RMSE values of 2.52 and 3.13, respectively, positioning them between SPMF and PMF in terms of initial accuracy.

SPMF shows a rapid decline in RMSE during the first 20 iterations, dropping from 2.40 to around 1.43. This improvement continues until around iteration 60, where it stabilizes around 1.15. PMF also exhibits a steep decline initially but stabilizes at a slightly lower RMSE value around 1.00 after 100 iterations. NMF demonstrates a steady decrease in RMSE but at a slower rate compared to SPMF and PMF, stabilizing around 1.66. SVD shows a decreasing trend initially, but its RMSE begins to increase after around iteration 60, indicating overfitting or instability in the model. After 100 iterations, PMF achieves the lowest RMSE of approximately 1.00, indicating strong performance among the methods tested. SPMF follows closely with a final RMSE of around 1.15, showcasing its robustness and rapid convergence. NMF stabilizes at a higher RMSE of about 1.66. SVD ends with the highest RMSE of around 3.17, indicating potential overfitting issues.

The comparison of SPMF, PMF, NMF, and SVD over 100 iterations reveals differences in convergence rates and final accuracy. SPMF demonstrates the fastest convergence and maintains a competitive final RMSE, making it suitable for this dataset and configuration. NMF shows a slower convergence rate and higher RMSE, indicating less capability in capturing the underlying structure of the data. SVD exhibits instability with an increasing RMSE after initial iterations, suggesting potential issues with overfitting or parameter settings. The results highlight the strengths of both PMF and SPMF in terms of convergence speed and accuracy, making them preferable choices for the development of recommender systems in this context. SPMF's ability to converge rapidly while maintaining a low RMSE underscores its robustness and potential as a valuable model for recommendation systems.

In the following section, we assess the performance of the SPMF models in comparison to other state-of-the-art models such as SVD, NMF, and PMF.

Evaluation of SPMF Model Performance on Subsets of FilmTrust Dataset

We randomly selected 20 samples from the FilmTrust dataset, each consisting of 100 users and 50 items, to assess the performance of state-of-the-art models (SVD, NMF, and PMF) in comparison to our three proposed SPMF models ("SPMF Normal," "SPMF Laplace," and "SPMF Bayes"). For each model, we ran their respective algorithms to determine the optimal number of latent factors k. Using this optimal k, we conducted matrix factorization over 100 iterations, recording the lowest RMSE for each sample and model. The resulting RMSE values for each model and sample are displayed in Figure 4.12.

Figure 4.12 compares the RMSE across different samples for various matrix factorization methods: SVD, NMF, PMF, "SPMF Bayes", "SPMF Normal", and "SPMF Laplace". Overall, SVD consistently exhibits higher RMSE values compared to other methods, indicating its relative inefficacy in minimizing error within this context. In contrast, PMF generally demonstrates lower RMSE values, suggesting strong performance in reducing prediction errors across samples. However, it is noteworthy that the RMSE of PMF is not always the lowest. There are instances where SPMF models, particularly "SPMF Bayes", show lower RMSE values than PMF, implying that SPMF methods can occasionally outperform PMF in terms of accuracy.

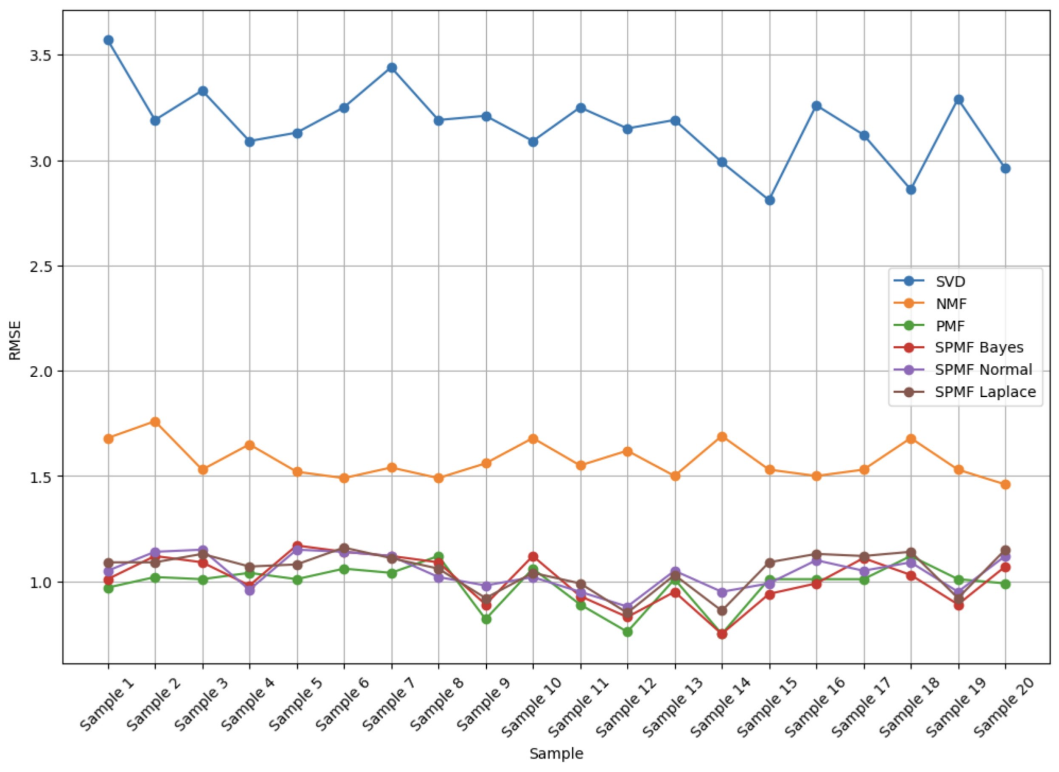

Figure 4.12 Comparison of RMSE values for 20 samples from the FilmTrust dataset across different models (SVD, NMF, PMF, "SPMF Bayes", "SPMF Normal", and "SPMF Laplace").

NMF shows noticeable variation in RMSE but generally remains within a moderate range. "SPMF Bayes", "SPMF Normal", and "SPMF Laplace" exhibit similar trends with occasional spikes, yet tend to stay lower than SVD and NMF. Among the SPMF variants, "SPMF Bayes" tends to have slightly lower RMSE values compared to "SPMF Normal" and "SPMF Laplace", suggesting marginally better performance.

The primary aim of developing the SPMF models was not just to attain the highest prediction accuracy but to create a model that balances interpretability and accuracy. Our objective is to benchmark the performance of SPMF models against other leading models. This analysis highlights the importance of SPMF models in the context of recommendation systems, where both accuracy and interpretability are essential for practical applications.

CHAPTER 5 CONCLUSION

This chapter provides an overview of our work and its alignment with the initially outlined objectives. Our thesis revolves around developing the Symmetric Probabilistic Matrix Factorization (SPMF) technique, primarily achieved through the construction of a symmetric block matrix based on user-item rating data. Utilizing SPMF simplifies the factorization problem by transforming the objective function into a convex problem, ensuring a global optimum. This transformation results in an optimal solution with a faster convergence rate. Here, we outline the primary contributions of SPMF as a matrix factorization technique, discuss research limitations, and propose potential areas for future enhancements. Additionally, we present prospective research directions within this field.

Summary of Contributions

The conclusion of this thesis signifies the culmination of research efforts that have yielded contributions to the domain of recommendation systems. The findings and insights derived from this study collectively represent an advancement in understanding, analyzing, and utilizing recommendation algorithms, particularly focusing on the SPMF algorithm. The conclusion of the thesis can be summarized as follows:

1. **Comparative Advantages of SPMF:** We systematically investigated the comparative advantages of SPMF over SVD, NMF, and PMF. Our analysis showed that SPMF addresses the rank deficiency issues caused by overdetermination or underdetermination by converting the original matrix into a symmetric blockwise matrix. By approximating this blockwise matrix and incorporating user-user and item-item information into a single low-rank factor, SPMF solves a convex objective function, providing optimal solutions with a faster convergence rate. Furthermore, SPMF offers improved interpretability by modeling user-user and item-item interactions from a probabilistic perspective.

   We also demonstrated the flexibility of SPMF to handle various data types by selecting the most relevant distribution for the given problem. For example, we successfully factorized binary matrices using a Bernoulli distribution to model the underlying data structure. This adaptability makes SPMF a versatile tool suitable for a wide range of matrix factorization applications, fulfilling the objective of demonstrating its comparative advantages over traditional techniques.

2. **Unified Matrix Representation:** By employing a single unified matrix in SPMF, we evaluated its benefits and trade-offs regarding computational complexity and interpretability. This approach streamlined the factorization process and enhanced the clarity of the latent factors. User-item rating matrices often face challenges due to imbalances between the number of items and users. These imbalances complicate data relationship interpretation and lead to matrices lacking sufficient rank, causing computational inefficiencies and inaccurate recommendations.

   To address these issues and improve interpretability, we constructed a blockwise symmetric matrix incorporating both user-user and item-item relationships. This transformation converted the original rectangular user-item matrix into a square symmetric matrix, allowing us to better capture and analyze the underlying interactions, leading to a convex single factor prediction. This method mitigated the limitations of imbalanced input matrices, providing a clearer representation of the data, enhancing user-item prediction, and alleviating rank deficiency problems.

   The unified matrix representation in SPMF facilitated the factorization process and improved the interpretability of latent factors through probabilistic distributions for matrix approximation. This methodology offered deeper insights into the interactions between users and items. By leveraging the symmetric property, SPMF ensured that the latent factors were interpretable in both directions, fostering a comprehensive understanding of user preferences and item characteristics. Additionally, the probabilistic nature of SPMF handled missing data and uncertainty, making it a versatile tool for real-world recommender systems. The resulting factorized matrices accurately represented user-item interactions and revealed hidden patterns and correlations essential for enhancing recommendation accuracy.

3. **Transforming biconvex to convex optimization**: Transforming a biconvex optimization problem into a convex optimization problem was a pivotal part of this research. This transformation ensured a unique optimal solution and enhanced convergence properties.

Consolidating two distinct factors into a single unified low-rank factor improved the interpretability of matrix factorization methods. This approach provided a clearer understanding of the underlying data structure by simplifying the complex relationships between users and items. The convex optimization method facilitated a more reliable solution process, ensuring robustness in the factorization outcomes.

By addressing the inherent challenges of biconvex optimization, a more interpretable framework for matrix factorization was developed. This transformation demonstrated the advantages of convex optimization in improving both the process and the clarity of the factorization results.

4. **Optimal Number of Latent Factors**: In this research, we developed methods to identify the optimal number of latent factors (k) for the SPMF model in recommendation systems. Understanding the impact of k on model performance was crucial for fine-tuning the model.

   The number of latent factors directly influences the complexity and performance of the model. If k is too small, the model may not capture the underlying patterns in the data, leading to inaccurate recommendations. Conversely, if k is too large, the model may overfit, performing well on training data but poorly on unseen data.

   To determine the optimal k, we used K-fold cross-validation to evaluate the SPMF model's performance across different values of k. Model performance was measured using RMSE, which indicates the model's accuracy and relevance of recommendations. By systematically varying k and recording the corresponding performance metrics, we identified the k value that minimized RMSE. Additionally, we assessed model complexity and potential overfitting by comparing training and validation performance. A gap between these performances suggested overfitting and the need for a lower k.

   This methodology allowed us to systematically determine the optimal number of latent factors for the SPMF model, ensuring a balance between model complexity and performance.

5. **Impact of Probabilistic Distribution**: In this study, we examined how different probabilistic distributions, such as Gaussian and Laplace, influence the robustness of SPMF recommendations against noise and outliers. The normal distribution captures the central tendency of user ratings but is sensitive to outliers. In contrast, the Laplace distribution, with its heavier tails, offers greater resilience to outliers and promotes sparsity in factorized matrices, making it better for handling data with extreme values.

   By leveraging different probabilistic distributions, SPMF strengthens its robustness against noise and outliers. Specifically, the Laplace distribution promotes sparsity in the factorized matrices, reducing the influence of outliers on the model. This characteristic is particularly beneficial in high-dimensional data, where noise and outliers can significantly impair the performance of traditional matrix factorization techniques. The use of the Laplace distribution results in a penalty term that encourages sparsity, making the factorization process more stable and reliable, and ensuring that the underlying latent structures are accurately captured.

   This robustness is crucial in applications such as recommender systems, where data is often noisy and subject to anomalies. Furthermore, the ability to handle noise and outliers allows SPMF to maintain interpretability, even with corrupted or incomplete data. This highlights the importance of selecting appropriate probabilistic models in matrix factorization, tailored to the specific characteristics of the dataset.

   The choice between these distributions depends on the dataset characteristics and the desired properties of the recommendation system. We employed methodologies involving Maximum Likelihood Estimation (MLE) and Bayesian estimation with these distributions to balance model robustness and interpretability.

6. **Binarization of the Matrix**: In the rapidly evolving landscape of AI-enabled digital entertainment, platforms like YouTube, Amazon, and Netflix have transformed video content consumption and advertising. The effectiveness of their recommendation systems is crucial for maximizing revenue through targeted advertisements by capturing and retaining user attention. Accurate predictions of user preferences enhance user engagement and advertisement delivery. We addressed the challenge of recommending suitable content from vast repositories, where user preferences are often represented as binary ratings (likes or dislikes).

Throughout this research, we leveraged the versatility of SPMF to handle various data types by selecting appropriate probabilistic distributions. Specifically, the Symmetric Probabilistic Binary Matrix Factorization (SPBMF) employed the Bernoulli distribution to model binary matrices, which is suitable for applications involving binary interactions, such as social media likes and dislikes. Additionally, other variants of SPMF can be developed for different data scenarios. For example, Poisson distributions can model count data. This flexibility enabled us to apply SPMF across a wide range of domains, including recommendation systems, network analysis, and collaborative filtering, enhancing its utility and applicability.

7. **Symmetric Probabilistic Binary Matrix Factorization (SPBMF)**: During this research, we extended our developed SPMF model to factorize binary matrices using the Bernoulli distribution, resulting in the Symmetric Probabilistic Binary Matrix Factorization (SPBMF). This novel approach, inspired by the logistic function, transforms dot product values into the $[0, 1]$ range using the sigmoid function.

   SPMF and its variant, SPBMF, offer advantages for single-factor low-rank factorization across a diverse range of applications, supported by solid theoretical guarantees. A major advantage of SPBMF is its ability to handle symmetric data. By leveraging the symmetric nature of adjacency matrices, SPBMF transforms the matrix factorization process into a more intuitive and interpretable format through graph representation. This approach facilitates the integration of user-user and item-item interactions into a unified low-rank factor, thereby providing a cohesive representation of the data. The symmetric nature of SPBMF allows it to capture inherent relationships within the data more accurately, making it particularly useful for applications involving networks and social interactions. This methodology not only enhances the interpretability of the factorization results but also ensures that the resulting model maintains the inherent properties of the original data structure. Furthermore, the theoretical foundations of SPBMF provide robust guarantees for convergence and optimality, ensuring reliable performance in practical scenarios. This makes SPBMF a tool for advancing the state-of-the-art in various domains, including recommendation systems, bioinformatics, and network analysis.

8. **Hyperparameter Tuning**: During this research, we emphasized the importance of hyperparameter tuning in optimizing the performance of the SPMF model. By system-

atically adjusting hyperparameters, we ensured that the model was closely aligned with the specific characteristics of the dataset. This tuning process enhanced the model's interpretability and predictive accuracy.

Key hyperparameters such as the learning rate, regularization parameters, and latent factor dimensions were fine-tuned to achieve optimal performance. We developed a structured methodology for evaluating various sets of hyperparameters, which facilitated the identification of the best configuration for the given data.

This approach to hyperparameter tuning improved the relevance and precision of SPMF in real-world applications, establishing it as a robust tool in recommendation systems.

9. **Comprehensive Performance Evaluation**: This research involved a thorough experimentation and evaluation process using subsets of the MovieLens 100K and FilmTrust datasets to demonstrate the improvements in recommendation robustness and interpretability provided by the SPMF models over existing methods. The performance of SPMF models was benchmarked against traditional techniques such as SVD, NMF, and PMF by implementing these models on selected datasets and measuring the RMSE for each method.

   The MovieLens 100K and FilmTrust datasets were divided into multiple subsets to ensure a thorough evaluation. Each subset included different numbers of users and items to test the models under various conditions. For each dataset, the models were run for 100 iterations, with RMSE recorded at each step to assess accuracy, convergence rate, and the ability to handle missing data and noise.

   The results showed that SPMF models, including "SPMF Normal", "SPMF Laplace", and "SPMF Bayes", consistently outperformed SVD and NMF in both accuracy and convergence rate. While PMF demonstrated competitive performance, SPMF models exhibited better robustness and interpretability. The unified matrix representation in SPMF enhanced the interpretability of latent factors, providing more insightful analysis of user-item interactions and managing user-user and item-item relationships.

   SPMF models demonstrated resilience to noise and missing data, maintaining stable RMSE values across iterations. The Laplace variant of SPMF was particularly noted for its robustness to outliers, making it suitable for datasets with extreme values. By leveraging probabilistic distributions and optimizing hyperparameters, SPMF offered a balanced approach to accuracy, robustness, and interpretability.

The performance evaluation underscores the advantages of SPMF models over traditional matrix factorization methods. These findings highlight the potential of SPMF as a tool for developing recommendation systems capable of delivering precise and reliable recommendations, thus reinforcing the contributions and originality of this research in advancing the field of recommendation systems.

10. **Theoretical and Mathematical Properties of SPMF**: This research highlights the theoretical and mathematical properties of the SPMF model, distinguishing it from traditional matrix factorization techniques that use two factors with biconvex objective functions. SPMF employs a single factor with a convex objective function, leading to a faster convergence rate compared to methods like Alternative Least Squares (ALS) and Stochastic Gradient Descent (SGD).

    We established that SPMF has a logarithmic convergence rate of $\mathcal{O}(\log(\frac{1}{\epsilon}))$ when the objective function is strongly convex with a Lipschitz continuous gradient. This condition is satisfied by demonstrating that the sum of the eigenvalues λ_i of the matrix UU^T exceeds $\frac{n+m}{3}$, where m and n represent the number of users and items, respectively. Furthermore, incorporating a Gaussian prior $\mathcal{N}(0, \sigma^2)$ on the low-rank matrix U imposes a strong convexity constraint on the objective function, denoted by $\frac{\|U\|_F^2}{2\sigma^2}$. This constraint is achieved within a Bayesian inference framework by leveraging the smoothness of the elements of U as a regularization term.

    This theoretical analysis clarifies the conditions ensuring a rapid convergence rate for the SPMF model. The inherent convexity of the objective function guarantees a faster convergence rate of $\mathcal{O}(\log(\frac{1}{\epsilon}))$, contrasting with other matrix factorization techniques with biconvex objective functions, which exhibit slower convergence rates of $\mathcal{O}(\frac{1}{\epsilon})$ or $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ depending on the optimization method used. Consequently, SPMF not only speeds up the convergence process but also provides interpretable low-rank factors for approximation. Furthermore, using subsets of the MovieLens 100K and FilmTrust datasets, it has been confirmed that SPMF converges faster than SVD, NMF, and PMF.

This faster convergence is especially advantageous in large-scale applications, where computational resources and time are critical. The stability and reliability of the convex optimization framework in SPMF ensure consistent results across different datasets, enhancing its utility in various recommendation systems and large-scale data analysis tasks. The robust mathematical foundations of SPMF thus contribute significantly to its performance and applicability.

The research contributed to both the practical implementation and theoretical understanding within the realm of recommendation systems. Practically, this study enhanced the interpretability of recommendation systems by exploring the SPMF algorithm. Through empirical analyses and real-world application, it offered insights into the algorithm's functionality and adaptability. The integration of SPMF into SPBMF, complemented by graph components, showcased its practical utility and potential for addressing modern challenges in recommendation scenarios. This implementation demonstrated the algorithm's versatility and its capacity to provide clear, interpretable results, which is crucial for user trust and transparency in recommendations.

Theoretically, this research advanced the comprehension of recommendation systems by elucidating the inner workings and limitations of SPMF. The critical analysis uncovered nuanced limitations, notably the algorithm's sensitivity to hyperparameters and its performance on large datasets. These theoretical insights provided a deeper understanding of the algorithm's intricacies and suggested pathways for enhancing interpretability. By offering a roadmap for refining SPMF's capabilities, the research contributed to the development of recommendation algorithms that are not only practical but also interpretable.

Furthermore, the suggestions for future research directions aim to propel the evolution of recommendation systems in both practical and theoretical domains. By outlining potential avenues for exploration and innovation, the study sets the stage for continued advancements in the field. These suggestions consider not only the practical implementation of recommendation algorithms but also the theoretical refinements necessary to improve their interpretability and adaptability in diverse recommendation scenarios.

Limitations, Improvements and Future Work

The research acknowledges the limitations and identifies areas for improvement and future exploration. Some potential directions for SPMF include:

1. **Scalability Enhancements:** Future work should concentrate on enhancing the scalability of SPMF by developing algorithms that leverage parallel computing techniques. This will enable the application of SPMF to larger and more complex datasets. One approach is to utilize sparse matrix representations, which reduce storage requirements and computational cost by focusing on the non-zero elements of large, sparse matrices. This can be achieved using data structures like Compressed Sparse Row (CSR) or Compressed Sparse Column (CSC), and by employing libraries optimized for sparse matrix operations [91]. From an algorithmic perspective, parallel and distributed computing techniques are crucial. Implementing Stochastic Gradient Descent (SGD) and its variants, such as mini-batch SGD and Hogwild! SGD, allows incremental updates of model parameters, making the factorization process more manageable [92]. Incorporating advanced hyperparameter tuning methods, such as Bayesian optimization, ensures that the model parameters are optimally set, balancing complexity and performance [93]. These combined mathematical and algorithmic strategies enable SPMF to scale to large datasets, ensuring robust and accurate recommendations.

   To address the scalability of probabilistic distribution objective functions such as normal and Laplace distributions, as well as Bayesian inference, a combination of mathematical and algorithmic strategies is essential. Key mathematical techniques involve leveraging sparse matrix representations to reduce memory usage and computational cost by focusing on non-zero elements. Efficient computation methods such as Cholesky decomposition for sparse matrices and Iteratively Reweighted Least Squares (IRLS) for Laplace distributions are crucial for maintaining computational feasibility [94]. Furthermore, Bayesian inference methods, such as Stochastic Variational Inference (SVI) and parallel Markov Chain Monte Carlo (MCMC), provide scalable solutions by approximating posterior distributions and running multiple chains simultaneously [95]. These methods are complemented by advanced regularization techniques like sparse Bayesian learning and elastic net regularization, which enforce sparsity and improve model robustness, ensuring that the SPMF framework remains computationally feasible for large datasets [95].

2. **Model Assumptions:** Model assumptions in probabilistic modeling, such as the choice of distribution, impact the performance and applicability of SPMF. Incorrect assumptions can lead to biased or inaccurate recommendations. Adapting SPMF or SPBMF to various applications requires a careful mathematical and algorithmic approach to incorporate domain-specific knowledge and handle diverse data types. This adaptation involves integrating known relationships, such as user-item interactions, through strategic initialization and regularization, ensuring the factorization process respects established associations and enhances the model's accuracy. Algorithmically, SPMF or SPBMF must address heterogeneous data types—binary, categorical, and continuous—by customizing probabilistic distribution functions. For example, using a Bernoulli distribution for binary data, a categorical distribution for categorical data, and a Gaussian distribution for continuous data ensures accurate representation. Robust distributions like the Laplace or Student's t-distribution can enhance resilience to outliers. These adaptations enable SPMF or SPBMF to provide personalized and accurate recommendations by capturing complex relationships and variability. Model assumptions, such as the choice of distribution and assumptions about latent factor independence or linearity of interactions, are crucial. Selecting appropriate distributions that align with data characteristics is essential. Thoroughly validating these assumptions against domain knowledge and empirical evidence ensures the robustness and performance of SPMF or SPBMF across various applications.

3. **Real-Time Recommendations:** Developing real-time recommendation capabilities within the SPMF framework can broaden its applicability to dynamic environments, such as online streaming platforms and e-commerce websites. To address this, incremental and online algorithms that update the model continuously as new data arrives can handle real-time and large-scale environments. Additionally, using adaptive algorithms such as Thompson Sampling and Contextual Bandits [96] helps in dynamically adjusting the recommendation strategies based on evolving user preferences, optimizing both exploration and exploitation phases. Algorithmically, the integration of streaming data processing frameworks like Apache Kafka and Apache Flink is essential for handling high-throughput, low-latency data streams [97]. These frameworks enable the construction of real-time data pipelines that seamlessly ingest and process incoming data. Implementing scalable infrastructure through distributed computing platforms like Apache Spark ensures robust data processing and model training at scale. Spark's MLlib library, combined with real-time stream processing capabilities, facilitates the deployment of scalable machine learning models [98]. Furthermore, leveraging cloud

computing services such as Amazon Web Services (AWS) and Google Cloud Platform (GCP) provides the necessary computational resources and tools to manage, deploy, and serve machine learning models in production, ensuring that the recommendation system can handle the demands of real-time data [99]. These combined mathematical and algorithmic approaches form a comprehensive strategy for maintaining computational efficacy in real-time e-commerce environments.

4. **Time Series SPMF:** Extending SPMF to handle time series data involves integrating temporal dynamics into the model, resulting in the Temporal Symmetric Probabilistic Matrix Factorization (TSPMF) approach. Mathematically, this involves representing the user-item interactions as a tensor $\mathcal{R} \in \mathbb{R}^{n \times m \times t}$, where n is the number of users, m is the number of items, and t represents the time intervals. The tensor is factorized into time-dependent user-time-user interaction, item-time-item interaction, and user-time-item rating latent factors U_t. The objective function incorporates temporal regularization terms, ensuring smooth transitions of latent factors over time, and penalizes large changes between consecutive time intervals. This results in a convex optimization problem that can be efficiently solved, maintaining the model's interpretability and robustness over time. Developing TSPMF to handle time series data through probabilistic distribution functions involves modeling the user-item-time interactions with a tensor, incorporating temporal regularization to ensure smooth transitions over time, and solving the resulting model using variational inference techniques. The EM algorithm and variational inference are key methods to estimate the posterior distributions of the latent factors efficiently [100]. For TSPMF, the choice of distribution should match the characteristics of the time-series data, e.g. Gaussian distribution is suitable for continuous data with normal noise, facilitating efficient computation. Laplace distribution is a better choice for data with outliers or heavy-tailed distributions. Gaussian process is an ideal approach for capturing complex temporal dependencies and trends. Hidden Markov Models (HMM) is more suitable for data with discrete underlying states or regimes [101]. Poisson distribution is an appropriate method for count-based or event-based time-series data. These mathematical and algorithmic enhancements ensure that TSPMF can manage the complexities of real-time, large-scale e-commerce data, providing accurate and timely recommendations.

5. **Domain-Specific Adaptations:** Tailoring SPMF to specific domains, such as healthcare, by incorporating domain-specific knowledge and constraints can improve its ap-

plicability and performance in those areas. Adapting SPMF or SPBMF to healthcare involves leveraging mathematical techniques and algorithmic strategies to incorporate domain-specific knowledge and handle diverse data types. Mathematically, the adaptation starts with integrating known medical relationships, such as symptom-disease associations, into the model through initialization and regularization. This ensures that the factorization process respects established medical knowledge, enhancing the model's accuracy and relevance. For instance, using regularization terms to enforce known symptom-disease relationships can guide the factorization towards medically meaningful latent factors. Algorithmically, SPMF or SPBMF can be tailored to handle the heterogeneous nature of healthcare data, which includes binary (presence or absence of conditions), categorical (disease types), and continuous (vital signs) information. By customizing the probabilistic distribution functions used in the model—such as using a Bernoulli distribution for binary data, a categorical distribution for categorical data, and a Gaussian distribution for continuous data—the model can accurately capture the different types of medical data. Furthermore, incorporating distributions such as the Laplace can improve the model's robustness to outliers, which are common in medical datasets due to measurement errors or rare conditions. These adaptations ensure that SPMF or SPBMF can provide personalized and accurate treatment recommendations by capturing the complex relationships and variability in healthcare data.

6. **Incorporating Side Information:** Integrating additional data, such as demographic or contextual information, has the potential to significantly enhance the performance of recommender systems. By leveraging this supplementary data alongside the existing user-item interactions, recommender systems can offer more personalized and accurate recommendations. Future research directions may delve deeper into strategies for integrating this side information into SPMF. This could involve exploring techniques within the SPMF framework that accommodate side information seamlessly or developing alternative methods specifically designed to handle and utilize such additional data sources more effectively. These efforts aim to enrich the recommendation process and improve its precision by leveraging diverse contextual cues or demographic insights available within the dataset.

7. **Symmetric Bayesian Probabilistic Matrix Factorization (SBPMF):** One promising direction for future research is the development of Symmetric Bayesian Probabilistic Matrix Factorization (SBPMF). This model will build upon the principles of Bayesian Probabilistic Matrix Factorization (BPMF), which utilizes prior distributions over latent factors to enhance robustness against data variability and noise.

   SBPMF will transform the user-item rating matrix into a symmetric block matrix, incorporating user-user and item-item relationships to provide a comprehensive view of the interactions within the dataset. This transformation is consistent with the methodology used in SPMF, ensuring interpretability and robustness.

   The advancement in SBPMF lies in the application of a Bayesian framework within this symmetric matrix structure. By introducing priors over the unified latent factors, the model will maintain the symmetric nature of the data while improving interpretability. Bayesian inference techniques, such as Markov Chain Monte Carlo (MCMC) or Variational Inference, will be employed to estimate the posterior distributions of the latent factors, handling uncertainty and improving robustness to noise and outliers.

   This approach will ensure consistent and meaningful factorization results, preserving the symmetric properties of the original matrix. SBPMF aims to provide a robust and scalable solution for various applications, including recommendation systems and network analysis, extending the capabilities of SPMF.

8. **Developing a Robust Recommendation System Using SPMF:** SPMF can be applied as a robust user authentication process to detect fake accounts and prevent them from biasing the ratings in recommendation systems. This application enhances the credibility and accuracy of the recommendation system by identifying and removing malicious users.

   An e-commerce website that uses a recommendation system can be affected by false recommendations due to malicious users creating fake accounts. These fake accounts can be used to manipulate product ratings, leading to biased recommendations and undermining the credibility of the recommendation system [102]. To overcome this challenge, SPMF can be applied as a robust user authentication process to detect fake accounts and prevent them from biasing the ratings. By implementing a robust user authentication process with SPMF, the recommendation system can be protected from false recommendations, improving the overall accuracy and reliability of the system. In three steps, SPMF can be applied to detect and remove fake users as listed below:

(a) Estimating the user-user similarity matrix: SPMF can be used to estimate the user-user similarity matrix, which is present in the bottom right block of the block matrix $\mathcal{S}$ in (3.2). By doing so, we can identify users who have a high correlation and examine the items they are interested in.

(b) Checking the item-item similarity matrix: By utilizing information obtained from the user-user similarity matrix (previous step), the item-item similarity matrix located in the top left block of matrix $\mathcal{S}$ can be evaluated. This matrix provides insights into the similarities between items and helps identify any patterns in the items that users with high correlation (as determined in the previous step) are interested in.

(c) Analyzing the off-diagonal matrix $\mathcal{S}$: Finally, by analyzing the off-diagonal matrix $\mathcal{S}$, we can determine if the users we found in the first step have given excessively high ratings to the items we found in the second step. If this is the case, these users are considered malicious and can negatively impact the credibility of the recommendation system.

To maintain the reliability of the recommendation system, it is crucial to identify and eliminate these malicious users. SPMF can play an important role in this process by allowing us to estimate the user-user similarity matrix and detect malicious users. By identifying and removing malicious users, SPMF helps to improve the credibility of recommendation systems and ensures that users receive accurate and trustworthy recommendations.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, *An Introduction to Recommender Systems Handbook.* Boston, MA: Springer US, 2011.

[2] S. Sivapalan *et al.*, "Recommender systems in e-commerce," *2014 World Automation Congress (WAC)*, pp. 179–184, 2014.

[3] "How retailers can keep up with consumers," https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers, 2013.

[4] N. Patel, "How to increase sales by personalizing your website," https://neilpatel.com/blog/increase-sales-personalizing-website/, 2020.

[5] F. O. Isinkaye, "Matrix factorization in recommender systems: algorithms, applications, and peculiar challenges," *IETE Journal of Research*, vol. 69, no. 9, pp. 6087–6100, 2023.

[6] C. C. Aggarwal *et al.*, *Recommender systems.* Springer, 2016, vol. 1.

[7] N. Gillis, "Sparse and unique nonnegative matrix factorization through data preprocessing," vol. stat.ML, 2012.

[8] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2007.

[9] P. Gupta *et al.*, "Wtf: the who to follow service at twitter," 05 2013, pp. 505–514.

[10] B. Sarwar *et al.*, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 285–295.

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

[12] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, vol. 23, 2011, p. 53.

[13] J. L. Herlocker *et al.*, "Evaluating collaborative filtering recommender systems," vol. 22, no. 1, p. 5–53, jan 2004. [Online]. Available: https://doi.org/10.1145/963770.963772

[14] S. Zhang *et al.*, “Deep learning based recommender system: A survey and new perspectives,” *ACM Comput. Surv.*, vol. 52, no. 1, feb 2019. [Online]. Available: https://doi.org/10.1145/3285029

[15] G. Adomavicius and A. Tuzhilin, *Context-Aware Recommender Systems.* Boston, MA: Springer US, 2011, pp. 217–253. [Online]. Available: https://doi.org/10.1007/978-0-387-85820-3-7

[16] S. Gong, H. Ye, and H. Tan, “Combining memory-based and model-based collaborative filtering in recommender system,” 05 2009, pp. 690–693.

[17] C. C. Aggarwal, *Recommender Systems - The Textbook.* Springer, 2016.

[18] N. Barbieri, G. Manco, and E. Ritacco, *Probabilistic approaches to recommendations.* Springer Nature, 2022.

[19] A. Menon and C. Elkan, “Fast algorithms for approximating the singular value decomposition,” *TKDD*, vol. 5, p. 13, 02 2011.

[20] C. Sammut and G. I. Webb, Eds., *Latent Factor Models and Matrix Factorizations.* Boston, MA: Springer US, 2010, pp. 571–571. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8-887

[21] G. Takács *et al.*, “Matrix factorization and neighbor based algorithms for the netflix prize problem,” in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 267–274.

[22] F. Vega and E. Cantu-Paz, *Parallel and Distributed Computational Intelligence*, 01 2010, vol. 269.

[23] G. M. Del Corso and F. Romani, “Adaptive nonnegative matrix factorization and measure comparisons for recommender systems,” *Applied Mathematics and Computation*, vol. 354, pp. 164–179, 2019.

[24] D. Jannach *et al.*, *Recommender Systems: An Introduction*, 1st ed. USA: Cambridge University Press, 2010.

[25] Z. Sharifi, M. Rezghi, and M. Nasiri, “New algorithm for recommender systems based on singular value decomposition method,” in *ICCKE 2013.* IEEE, 2013, pp. 86–91.

[26] F. O. Isinkaye, “Matrix factorization in recommender systems: Algorithms, applications, and peculiar challenges,” *IETE Journal of Research*, pp. 1–14, 2021.

[27] B. Abdollahi and O. Nasraoui, "Using explainability for constrained matrix factorization," in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 79–83.

[28] Z. Xian *et al.*, "New collaborative filtering algorithms based on svd++ and differential privacy," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[29] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *ieee transactions on knowledge and data engineering*, vol. 28, no. 7, pp. 1607–1620, 2016.

[30] S. K. Raghuwanshi and R. K. Pateriya, "Accelerated singular value decomposition (asvd) using momentum based gradient descent optimization," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 4, pp. 447–452, 2021.

[31] A. M. A. Al-Sabaawi, H. Karacan, and Y. E. Yenice, "Exploiting implicit social relationships via dimension reduction to improve recommendation system performance," *PloS one*, vol. 15, no. 4, p. e0231457, 2020.

[32] S. Balakrishnan *et al.*, "Book recommendation system using matrix factorization with svd," in *2023 Second International Conference on Advances in Computational Intelligence and Communication (ICACIC)*. IEEE, 2023, pp. 1–5.

[33] Y. Chen, "A music recommendation system based on collaborative filtering and svd," in *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*. IEEE, 2022, pp. 1510–1513.

[34] M. Z. Bahar and Z. Baizal, "Online course recommender system using singular value decomposition," in *2023 International Conference on Data Science and Its Applications (ICoDSA)*. IEEE, 2023, pp. 187–190.

[35] L. J. Ming, C. H. Lim, and I. K. Tan, "Improving co-svd for cold-start recommendations using sparsity reduction," in *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2022, pp. 990–996.

[36] R. H. Sahib, "Binary data in matrices with singular value decomposition method," in *AIP Conference Proceedings*, vol. 3051, no. 1. AIP Publishing, 2024.

[37] A. Mirzal and M. Furukawa, "On the clustering aspect of nonnegative matrix factorization," *CoRR*, 2010.

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

[39] X. Ran *et al.*, “A differentially private nonnegative matrix factorization for recommender system,” *Information Sciences*, vol. 592, pp. 21–35, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025522000792

[40] C. Ding, X. He, and H. D. Simon, “On the equivalence of nonnegative matrix factorization and spectral clustering,” in *Proc. SIAM Data Mining Conf*, 2005, pp. 606–610.

[41] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[42] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

[43] Y.-X. Wang and Y.-J. Zhang, “Nonnegative matrix factorization: A comprehensive review,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[44] G. M. Del Corso and F. Romani, “A multi-class approach for ranking graph nodes: Models and experiments with incomplete data,” *Information Sciences*, vol. 329, pp. 619–637, 2016.

[45] A. M. Tomé *et al.*, “A logistic non-negative matrix factorization approach to binary data sets,” *Multidimensional Systems and Signal Processing*, vol. 26, pp. 125–143, 2015.

[46] A. Lumbreras, L. Filstroff, and C. Févotte, “Bayesian mean-parameterized nonnegative binary matrix factorization,” *Data mining and knowledge discovery*, vol. 34, no. 6, pp. 1898–1935, 2020.

[47] X. Ma *et al.*, “Probabilistic non-negative matrix factorization with binary components,” *Mathematics*, vol. 9, p. 1189, 05 2021.

[48] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *NIPS*, 2000, pp. 556–562.

[49] W.-S. Chen *et al.*, “Symmetric nonnegative matrix factorization: A systematic review,” *Neurocomputing*, p. 126721, 2023.

[50] G. Li and W. Ou, “Pairwise probabilistic matrix factorization for implicit feedback collaborative filtering,” *Neurocomputing*, vol. 204, pp. 17–25, 2016.

[51] X. Gong and X. Huang, “A probabilistic matrix factorization recommendation method based on deep learning,” in *Journal of Physics: Conference Series*, vol. 1176, no. 2. IOP Publishing, 2019, p. 022043.

[52] K. Li *et al.*, “Deep probabilistic matrix factorization framework for online collaborative filtering,” *IEEE Access*, vol. 7, pp. 56 117–56 128, 2019.

[53] M. Aktukmak, Y. Yilmaz, and I. Uysal, “A probabilistic framework to incorporate mixed-data type features: Matrix factorization with multimodal side information,” *Neurocomputing*, vol. 367, pp. 164–175, 2019.

[54] P. Akulwar and S. Pardeshi, “Bayesian probabilistic matrix factorization—a dive towards recommendation,” in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3. IEEE, 2016, pp. 1–5.

[55] J. Shi, X. Zheng, and W. Yang, “Survey on probabilistic models of low-rank matrix factorizations,” *Entropy*, vol. 19, no. 8, p. 424, 2017.

[56] W. Zhang *et al.*, “Recommendation system in social networks with topical attention and probabilistic matrix factorization,” *PloS one*, vol. 14, no. 10, p. e0223967, 2019.

[57] T. Tamada and R. Saga, “Double-convmf: probabilistic matrix factorization with user and item characteristic text,” *Artificial Life and Robotics*, vol. 29, no. 1, pp. 107–113, 2024.

[58] S. Jain *et al.*, “Graph regularized probabilistic matrix factorization for drug-drug interactions prediction,” *IEEE Journal of Biomedical and Health Informatics*, 2023.

[59] Q. Kong *et al.*, “Probabilistic matrix factorization for data with attributes based on finite mixture modeling,” *IEEE Transactions on Cybernetics*, 2022.

[60] J. Deng *et al.*, “Probabilistic matrix factorization recommendation approach for integrating multiple information sources,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[61] A. Pujahari and D. S. Sisodia, “Pair-wise preference relation based probabilistic matrix factorization for collaborative filtering in recommender system,” *Knowledge-Based Systems*, vol. 196, p. 105798, 2020.

[62] L. Han, L. Chen, and X. Shi, "Recommendation model based on probabilistic matrix factorization, integrating user trust relationship, interest mining, and item correlation," *IEEE Access*, vol. 10, pp. 132 315–132 331, 2022.

[63] Y. Dong *et al.*, "Probabilistic matrix factorization recommendation algorithm with user trust similarity," in *MATEC Web of Conferences*, vol. 208. EDP Sciences, 2018, p. 05004.

[64] J. Liu, M. Shao, and W.-K. Ma, "Probabilistic semi-nonnegative matrix factorization via maximum-likelihood and variational inference," in *2023 31st European Signal Processing Conference (EUSIPCO).* IEEE, 2023, pp. 1280–1284.

[65] P. Figuera and P. García Bringas, "Revisiting probabilistic latent semantic analysis: Extensions, challenges and insights," *Technologies*, vol. 12, no. 1, p. 5, 2024.

[66] Z. He *et al.*, "Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2117–2131, 2011.

[67] T. Ramathulasi and M. R. Babu, "Assessing user interest in web api recommendation using deep learning probabilistic matrix factorization," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 1, 2023.

[68] X. Yang *et al.*, "Dpmf: Decentralized probabilistic matrix factorization for privacy-preserving recommendation," *Applied Sciences*, vol. 12, no. 21, p. 11118, 2022.

[69] X. Wei *et al.*, "Probabilistic matrix factorization for visual tracking," *Pattern Recognition and Image Analysis*, vol. 32, no. 1, pp. 57–66, 2022.

[70] X. Li and L. Zhang, "Probabilistic matrix completion," in *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2022, pp. 1362–1367.

[71] L. Qiu, W. Zhou, and Y. Yang, "Stpr: Personalized recommendation via matrix factorization with social ties," in *2023 20th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP).* IEEE, 2023, pp. 1–4.

[72] X. Wang *et al.*, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[73] G. Li *et al.*, "Supervised singular value decomposition and its asymptotic properties," *Journal of Multivariate Analysis*, vol. 146, pp. 7–17, 2016, special Issue on Statistical Models and Methods for High or Infinite Dimensional Spaces. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0047259X15000512

[74] X. Dong *et al.*, "A hybrid collaborative filtering model with deep structure for recommender systems," in *Proceedings of the AAAI Conference on artificial intelligence*, vol. 31, no. 1, 2017.

[75] S. Sedhain *et al.*, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

[76] H. Wang *et al.*, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[77] H. Asaoka and K. Kudo, "Nonnegative/binary matrix factorization for image classification using quantum annealing," *Scientific Reports*, vol. 13, no. 1, p. 16527, 2023.

[78] G. Garrigos and R. M. Gower, "Handbook of convergence theorems for (stochastic) gradient methods," *arXiv preprint arXiv:2301.11235*, 2023.

[79] A. Janecek and Y. Tan, "Using population based algorithms for initializing nonnegative matrix factorization," in *ICSI (2)*, 2011, pp. 307–316.

[80] M. N. Schmidt, O. Winther, and L. K. Hansen, "Bayesian non-negative matrix factorization," in *Independent Component Analysis and Signal Separation, International Conference on*, ser. Lecture Notes in Computer Science (LNCS), vol. 5441. Springer, 2009, pp. 540–547.

[81] S. M. Wild, J. H. Curry, and A. Dougherty, "Improving non-negative matrix factorizations through structured initialization," *Pattern Recognition*, vol. 37, pp. 2217–2232, 2004.

[82] C. Boutsidis and E. Gallopoulos, "SVD based initialization: A head start for nonnegative matrix factorization," *Pattern Recognition*, vol. 41, pp. 1350–1362, 2008.

[83] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[84] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[85] D. Dueck *et al.*, “Probabilistic sparse matrix factorization,” *University of Toronto technical report PSI-2004-23*, 2004.

[86] S. Bubeck *et al.*, “Convex optimization: Algorithms and complexity,” *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[87] 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.

[88] G. Guo, J. Zhang, and N. Yorke-Smith, “A novel bayesian similarity measure for recommender systems,” in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013, pp. 2619–2625.

[89] M. R. Zarei, M. R. Moosavi, and M. Elahi, “Adaptive trust-aware collaborative filtering for cold start recommendation,” *Behaviormetrika*, vol. 50, no. 2, pp. 541–562, 2023.

[90] S. Lee, J. Ahn, and N. Kim, “Embedding enhancement method for lightgcn in recommendation information systems,” *Electronics*, vol. 13, no. 12, p. 2282, 2024.

[91] J. Scott and M. Tuma, *Algorithms for sparse linear systems.* Springer Nature, 2023.

[92] K. Bäckström, *Adaptiveness, Asynchrony, and Resource Efficiency in Parallel Stochastic Gradient Descent.* Chalmers Tekniska Hogskola (Sweden), 2023.

[93] J. A Ilemobayo *et al.*, “Hyperparameter tuning in machine learning: A comprehensive review,” *Journal of Engineering Research and Reports*, vol. 26, no. 6, pp. 388–395, 2024.

[94] J. Lin, “Sparse models for machine learning,” in *Engineering Mathematics and Artificial Intelligence.* CRC Press, 2023, pp. 107–146.

[95] G. M. Martin, D. T. Frazier, and C. P. Robert, “Approximating bayes in the 21st century,” *Statistical Science*, vol. 39, no. 1, pp. 20–45, 2024.

[96] M. Byrd and R. Darrow, “A note on the advantage of context in thompson sampling,” in *Artificial Intelligence and Machine Learning in the Travel Industry: Simplifying Complex Decision Making.* Springer, 2023, pp. 109–114.

[97] B. G. Deepthi *et al.*, “An efficient architecture for processing real-time traffic data streams using apache flink,” *Multimedia Tools and Applications*, vol. 83, no. 13, pp. 37 369–37 385, 2024.

[98] A. Polak, *Scaling Machine Learning with Spark.* " O'Reilly Media, Inc.", 2023.

[99] P. Borra, "A survey of google cloud platform (gcp): Features, services, and applications," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 4, no. 3, pp. 191–199, 2024.

[100] A. E. Cho *et al.*, "Regularized variational estimation for exploratory item factor analysis," *Psychometrika*, vol. 89, no. 1, pp. 347–375, 2024.

[101] M. L. Gámiz, N. Limnios, and M. del Carmen Segovia-García, "Hidden markov models in reliability and maintenance," *European Journal of Operational Research*, vol. 304, no. 3, pp. 1242–1255, 2023.

[102] S. S. Fadaee *et al.*, "Chiron: A robust recommendation system with graph regularizer," 2016.

[103] S. P. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

APPENDIX A

In Chapter 3, we demonstrated that the SPMF objective function is μ-strongly convex and that its gradient, $\nabla f(U)$, is L-Lipschitz continuous. As a result, we can apply Theorem 3 from [86, 103]. This theorem states that for any μ-strongly convex function with an L-Lipschitz continuous gradient, the convergence rate of the optimization algorithm is $\mathcal{O}(\log(1/\epsilon))$. Here, we aim to prove Theorem 3. We will begin by proving several important lemmas from [86,103] that will lead to the proof of Theorem 3.

Lemma 11. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable function whose gradient ∇f is Lipschitz continuous with constant L. Then the Hessian matrix $\nabla^2 f(x)$ of f at any point $x \in \mathbb{R}^n$ is bounded in the operator norm by L, i.e.,*

$$\|\nabla^2 f(x)\| \leq L,$$

where $\|\cdot\|$ denotes the operator norm.

Proof. Consider the definition of Lipschitz continuity of the gradient of f, which states that for all $x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

This imposes a uniform upper bound on the rate of change of the gradient across the domain of f. Given that $\nabla^2 f(x)$ represents the second derivatives of f, it essentially measures how rapidly the gradient changes in every direction around x. When considering the Taylor expansion of ∇f along a line segment from x to y,

$$\nabla f(y) \approx \nabla f(x) + \nabla^2 f(x)(y - x) + o(\|y - x\|).$$

As y approaches x, dividing through by $\|y - x\|$ and taking the limit gives

$$\nabla^2 f(x) = \lim_{y \to x} \frac{\nabla f(y) - \nabla f(x)}{\|y - x\|},$$

which must be bounded by L due to the Lipschitz condition. Therefore, the operator norm of the Hessian matrix $\nabla^2 f(x)$ cannot exceed L. $\square$

Lemma 12. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable function whose gradient ∇f is L-*

Lipschitz continuous. Then, for any $x, y \in \mathbb{R}^n$,

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2.$$

Proof. We begin by applying the second-order Taylor expansion of f around x:

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} (y - x)^T \nabla^2 f(\xi)(y - x),$$

where ξ is some point on the line segment between x and y. Given that the gradient ∇f is L-Lipschitz continuous, based on Lemma 11, the operator norm of $\nabla^2 f$ is bounded by L. Therefore, we have:

$$\|\nabla^2 f(\xi)\| \leq L,$$

which implies:

$$(y - x)^T \nabla^2 f(\xi)(y - x) \leq L \|y - x\|^2.$$

Substituting this bound back into the Taylor expansion, we obtain:

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2.$$

This completes the proof, demonstrating that the function f does not increase more than the specified quadratic bound when moving from x to y. □

Lemma 13. *For a μ-strongly convex function f, the gradient norm satisfies the inequality:*

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f(x^*))$$

where x^* is the minimizer of f.

Proof. Because f is μ-strongly convex, for all $x, y \in \mathbb{R}^n$ and some $\mu > 0$, from Definition 1, we have:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|^2.$$

To establish the gradient norm inequality, we use the definition of strong convexity with $y = x^*$, where x^* is the minimizer of f:

$$f(x^*) \geq f(x) + \nabla f(x)^T (x^* - x) + \frac{\mu}{2} \|x^* - x\|^2$$

Rearranging terms to isolate $f(x) - f(x^*)$, we get:

$$f(x) - f(x^*) \leq \nabla f(x)^T(x - x^*) - \frac{\mu}{2}\|x - x^*\|^2$$

Applying the Cauchy-Schwarz inequality to the term $\nabla f(x)^T(x - x^*)$, we have:

$$\nabla f(x)^T(x - x^*) \leq \|\nabla f(x)\|\|x - x^*\|$$

Substituting this into the previous inequality, we obtain:

$$f(x) - f(x^*) \leq \|\nabla f(x)\|\|x - x^*\| - \frac{\mu}{2}\|x - x^*\|^2$$

Rearranging this to form a quadratic inequality in $\|x - x^*\|$, we get:

$$\frac{\mu}{2}\|x - x^*\|^2 - \|\nabla f(x)\|\|x - x^*\| + (f(x) - f(x^*)) \leq 0$$

This quadratic inequality holds if its discriminant is non-negative. The discriminant of the quadratic equation $ax^2 + bx + c = 0$ is given by $b^2 - 4ac$. Here, $a = \frac{\mu}{2}$, $b = -\|\nabla f(x)\|$, and $c = f(x) - f(x^*)$:

$$(\|\nabla f(x)\|)^2 - 2\mu(f(x) - f(x^*)) \geq 0$$

Rearranging this inequality gives us the desired result:

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f(x^*))$$

Thus, we have shown that for a μ-strongly convex function, the gradient norm provides a lower bound proportional to the function value gap scaled by the strong convexity parameter μ. □

Theorem 3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that is μ-strongly convex and has an L-Lipschitz continuous gradient. Then, the convergence rate of gradient descent applied to f with an appropriate step size is $O(\log(1/\epsilon))$, where ϵ represents the error tolerance [86, 103].*

Proof. For a function f with an L-Lipschitz continuous gradient, as stated in Lemma 12,

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2$$

Also, the gradient descent update rule is given by:

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

where α is the step size. To ensure convergence, we choose the step size α such that:

$$0 < \alpha < \frac{2}{L}.$$

Applying the gradient descent update rule and using the Lemma 12:

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2.$$

Substituting $x_{k+1} = x_k - \alpha \nabla f(x_k)$:

$$f(x_{k+1}) \leq f(x_k) - \alpha\|\nabla f(x_k)\|^2 + \frac{L\alpha^2}{2}\|\nabla f(x_k)\|^2.$$

Simplifying, we get:

$$f(x_{k+1}) \leq f(x_k) - \left(\alpha - \frac{L\alpha^2}{2}\right)\|\nabla f(x_k)\|^2.$$

Choosing $\alpha = \frac{1}{L}$ to ensure convergence:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2.$$

Using the Lemma **??**, we have:

$$\|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f(x^*))$$

where x^* is the minimizer of f. Substituting this into the descent inequality:

$$f(x_{k+1}) \leq f(x_k) - \frac{\mu}{L}(f(x_k) - f(x^*)).$$

Letting $\epsilon_k = f(x_k) - f(x^*)$, we obtain the recurrence relation:

$$\epsilon_{k+1} \leq \left(1 - \frac{\mu}{L}\right)\epsilon_k.$$

Solving this recurrence relation, we get:

$$\epsilon_k \leq \left(1 - \frac{\mu}{L}\right)^k \epsilon_0.$$

Using the approximation $\log(1 - x) \approx -x$ for small x:

$$\left(1 - \frac{\mu}{L}\right)^k \approx e^{-\frac{\mu}{L}k}.$$

For $\epsilon_k \leq \epsilon$:

$$k \geq \frac{L}{\mu} \log \frac{1}{\epsilon}.$$

Thus, the convergence rate is:

$$O\left(\log \frac{1}{\epsilon}\right)$$

□