ANGSHUL MAJUMDAR

COLLABORATIVE FILTERING

Recommender Systems

CRC Press
Taylor & Francis Group

Collaborative Filtering

This book dives into the inner workings of recommender systems, those ubiquitous technologies that shape our online experiences. From Netflix show suggestions to personalized product recommendations on Amazon or the endless stream of curated YouTube videos, these systems power the choices we see every day.

Collaborative filtering reigns supreme as the dominant approach behind recommender systems. This book offers a comprehensive exploration of this topic, starting with memory-based techniques. These methods, known for their ease of understanding and implementation, provide a solid foundation for understanding collaborative filtering. As you progress, you'll delve into latent factor models, the abstract and mathematical engines driving modern recommender systems.

The journey continues with exploring the concepts of metadata and diversity. You'll discover how metadata, the additional information gathered by the system, can be harnessed to refine recommendations. Additionally, the book delves into techniques for promoting diversity, ensuring a well-balanced selection of recommendations. Finally, the book concludes with a discussion of cutting-edge deep learning models used in recommender systems.

This book caters to a dual audience. First, it serves as a primer for practicing IT professionals or data scientists eager to explore the realm of recommender systems. The book assumes a basic understanding of linear algebra and optimization but requires no prior knowledge of machine learning or programming. This makes it an accessible read for those seeking to enter this exciting field. Second, the book can be used as a textbook for a graduate-level course. To facilitate this, the final chapter provides instructors with a potential course plan.

Collaborative Filtering

Recommender Systems

Angshul Majumdar

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

Cover: Web Large Image (Public)

First edition published 2025
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 978-1-032-84082-6 (hbk)
ISBN: 978-1-032-84112-0 (pbk)
ISBN: 978-1-003-51126-7 (ebk)

DOI: 10.1201/9781003511267

Typeset in Times
by Apex CoVantage, LLC

Contents

Foreword ix
Preface xi
Author xiii

Chapter 1 Introduction and Organization 1

1.1 Introduction 1
1.1.1 The Role of Recommender Systems in the Digital Age 4
1.2 Contents of This Book 7
1.2.1 Chapter 2 7
1.2.2 Chapter 3 8
1.2.3 Chapter 4 8
1.2.4 Chapter 5 9
1.2.5 Chapter 6 10
1.2.6 Chapter 7 11

Chapter 2 Neighborhood-Based Models 13

2.1 Introduction 13
2.1.1 User-Based CF 15
2.1.2 Item-Based CF 15
2.2 User-Based Approach 16
2.3 Item-Based Approach 19
2.3.1 Advantage of the Item-Based Approach 21
2.3.2 Neighborhood Selection 23
2.3.3 Coverage 25

Chapter 3 Ratings 28

3.1 Introduction 28
3.2 Biases and Baseline Correction 30
3.2.1 User Bias: The Tinted Lens of Personal Preferences 30
3.2.2 Item Bias: The Halo Effect of Popularity and Disrepute 31
3.2.3 The Impact of Bias on Collaborative Filtering 31
3.2.4 Baseline Estimation 33
3.3 Significance Weighting 35
3.4 Optimally Learned Interpolation Weights 39

Chapter 4 Latent Factor Models 43

4.1 Introduction 43
4.2 Latent Factor Model 45
4.2.1 Matrix Factorization 46
4.2.2 Matrix Completion via Factorization 49
4.2.3 Multiplicative Updates 53
4.3 Nuclear Norm Minimization 54

Chapter 5 Using Metadata 59

5.1 Introduction 59
5.2 Matrix Factorization on Graphs 61
5.3 Nuclear Norm Minimization on Multiple Graphs 63
5.4 Label-Consistent Nuclear Norm Minimization 65
5.5 Label-Consistent Matrix Factorization 69

Chapter 6 Diversity in Recommender Systems 81

6.1 Introduction 81
6.1.1 The Monotony of Accuracy-Driven Recommendations 81
6.1.2 The Dilemma of Popularity Bias 81
6.1.3 The Business Case for Diversity 81
6.1.4 The Quest for Optimal Breadth and Diversity 82
6.1.5 Balancing Accuracy and Diversity 82
6.1.6 The Future of Recommender Systems: diversity as a Driver of Value 82
6.1.7 Balancing Accuracy and Diversity in Recommender Systems 82
6.1.8 Two-Stage Recommendation Strategies 82
6.1.9 Unified Models 83
6.1.10 Individual Diversity (ID) and Aggregate Diversity (AD) 83
6.1.11 The Interplay of ID and AD 83
6.1.12 Balancing ID and AD for Optimal RS Design 83
6.2 Prior Art 84
6.2.1 Blind Compressed Sensing 85
6.3 Matrix Factorization-Based Diversity Model 86
6.3.1 Derivation 90
6.4 Nuclear Norm-Based Diversity Model 91
6.4.1 Derivation 95

Chapter 7 Deep Latent Factor Models 98

7.1 Introduction 98
7.1.1 Advantages of Deep Latent Factor Models 98

7.2 Brief Introduction to Representation Learning 100
7.3 Deep Latent Factor Model 104
7.3.1 Mathematical Formulation 106
7.3.2 Solution 107
7.4 Graphical Deep Latent Factor Model 109
7.5 Diversity in Deep Latent Factor Model 112

Chapter 8 Conclusion and Note to Instructors 118

8.1 Introduction 118
8.2 Course Organization 120
8.3 Expectation from Pupils 121
8.4 Evaluation 122

Index 127

Foreword

It is my pleasure to introduce Dr. Angshul Majumdar's latest contribution to the field of recommender systems. I have known Dr. Majumdar for the past seven years and have witnessed his exceptional work as Research Advisor at TCS Research since 2017. Moreover, his students, whom we have had the pleasure of hiring, have consistently demonstrated their prowess as prolific researchers under his guidance.

If we explore existing literature on recommender systems, we can see a dearth of comprehensive textbooks in this domain. While there are a few publications available, many are either anthologies or dated works that lack coverage of contemporary advancements. Upon reviewing the contents of this book, I was impressed by its thoroughness and accessibility. Beginning with fundamental techniques and progressing to advanced topics such as ratings bias, diversity, and deep learning-based recommender systems, this text is suitable for both senior undergraduates and graduate students in computer science and related disciplines.

Dr. Majumdar's emphasis on pedagogy is evident throughout the book, particularly in the final chapter tailored for instructors. Drawing from his teaching experience, he provides invaluable insights into course prerequisites, teaching strategies, and exercises, making this book an indispensable resource for classroom instruction. In addition to being a textbook on this important topic, this book can become a good practitioner's guide.

With a prolific record of publication and a talent for elucidating complex concepts, Dr. Majumdar brings a unique perspective to the subject matter. Whether you are a student seeking a comprehensive guide to collaborative filtering or a curious reader interested in understanding the inner workings of recommender systems, this book promises to be an indispensable companion on your journey.

I am confident that readers will find Dr. Majumdar's expertise and passion for the subject reflected in these pages, making it a must-have addition to any library or curriculum.

Dr. Arpan Pal
Distinguished Chief Scientist TCS Research
Tata Consultancy Services, India

Preface

My journey with teaching collaborative filtering began in the fall of 2013. Since then, it has evolved into an annual staple in my curriculum, save for a brief sabbatical in 2019. At the outset, I faced a significant challenge: the lack of a suitable textbook for this burgeoning field. The only available resource was an anthology by renowned experts, valuable but ultimately inadequate as a cohesive learning tool due to its inherent disconnectedness.

My early teaching attempts were marked by trial and error. Initially, I overestimated the students' preparedness, rushing through basic models and focusing heavily on recent advancements. This resulted in a sense of bewilderment and confusion among them, a mistake I readily acknowledge and assume full responsibility for. Subsequently, I adopted a completely different approach, dedicating extensive time to establishing a solid foundation in the fundamentals. While this approach laid a strong groundwork, it left insufficient room for exploring contemporary developments.

Through a series of adjustments and refinements, I gradually arrived at the carefully curated material that forms the core of this book. It strikes a delicate balance between accessibility and depth, ensuring that students acquire a comprehensive understanding of both the fundamental principles and cutting-edge advancements in collaborative filtering.

Over the years, the landscape of recommender systems has witnessed the release of numerous books, each valuable in its own right. However, none of them fully resonated with my vision for a textbook suitable for a graduate-level or advanced undergraduate-level course. Existing volumes either lacked the necessary depth for graduate students or assumed knowledge of advanced topics that are typically absent in first-year graduate students or senior undergraduates. This is the primary motivation behind this book, which encompasses approximately 25 years of research, spanning from the late 1990s to 2022.

The book embarks on its journey with the fundamentals of memory-based neighborhood models, paving the way for a thorough exploration of latent factor models and matrix factorization. These foundational concepts serve as a springboard for venturing into more advanced topics, including the integration of metadata, diversity in recommender systems, and the application of deep learning. The concluding chapter caters specifically to instructors who plan to utilize this book as a primary resource for their courses.

The inclusion of these advanced topics is a testament to the remarkable work of my exceptional graduate students. I have been truly fortunate to have mentored talented individuals who have made significant contributions to the field and, consequently, to the shape of this book. Anupriya Gogna, my first PhD student to delve into this subject, collaborated with me from 2013 to 2017. Around the same time, Shisagnee Banerjee, a brilliant master's student, conducted insightful research on

cold-start problems in recommender systems, graduating in 2016. More recently, Aanchal Mongia, whose PhD dissertation focused on bioinformatics, applied her expertise to the algorithmic aspects of collaborative filtering. Interestingly, all three of these gifted individuals are currently contributing their talents to medical informatics at GE Healthcare.

Author

Angshul Majumdar is a professor at TCG CREST, Kolkata, India. Prior to that, he was a professor at Indraprastha Institute of Information Technology, Delhi, India. He has been associated with the institute since 2012. Angshul did his master's (2009) and PhD (2012) in electrical and computer engineering from the University of British Columbia, Vancouver, Canada.

Angshul's research interests lie in signal processing and machine learning with applications in smart grids and bioinformatics. Angshul has co-authored over 200 articles in journals and top-tier conferences. He has written two books and co-edited two more and holds seven US patents. He is an associate editor for *IEEE Open Journal of Signal Processing* and *Neurocomputing.* In the past, he served as an associate editor for *IEEE Transactions on Circuits and Systems for Video Technology.*

Angshul is the director of student services at IEEE Signal Processing Society. Prior to that, he was the chair for the education committee in the IEEE SPS membership board (2019). Angshul also served as the chair for the chapter's committee in the IEEE SPS membership board (2016–18) and the founding chair of IEEE SPS Delhi Chapter (2015–18). Angshul was the organizing chair of two IEEE SPS Winter Schools in 2014 and 2017. He served as the finance chair of IEEE ISBA 2017, the flagship conference of IEEE Biometrics Council.

1 Introduction and Organization

1.1 INTRODUCTION

In the ever-expanding realm of digital media, the sheer volume of content available to consume is staggering. To put it into perspective, consider the ever-growing universe of Netflix, where a single dedicated viewer would require a mind-boggling four years of nonstop watching to exhaust its offerings. And that's just the current library; by the time such a feat could be accomplished, Netflix would have undoubtedly amassed an even more extensive collection of shows and movies.

Amazon, another titan of digital entertainment, boasts an equally impressive inventory of products. Its retail platform alone harbors an astonishing 12 million items, while the Amazon marketplace, where third-party sellers come into play, expands that figure to an eye-watering 350 million. These figures represent a carefully curated selection of products, ensuring a certain level of quality and relevance to consumers.

But when we delve into the realm of user-driven content platforms like YouTube, the numbers truly defy comprehension. Every minute, a staggering 2,500 new videos grace the platform, translating to a mind-boggling 183 hours of fresh content uploaded every 60 seconds. This relentless influx of material, spanning diverse genres and interests, reflects the boundless creativity and self-expression that fuel the YouTube phenomenon.

These staggering figures underscore the transformative impact of digital media on our lives. We are no longer confined to the limitations of traditional media, where content creation and distribution were tightly controlled. Now, anyone with an internet connection and a creative spark can contribute to the ever-expanding digital landscape.

This democratization of content creation has fostered an explosion of diverse perspectives, niche interests, and innovative storytelling. From captivating documentaries to hilarious comedy sketches, from thought-provoking discussions to mesmerizing musical performances, the digital realm offers a smorgasbord of content to suit every taste and inclination. However, this overwhelming abundance also presents a challenge. With such a vast ocean of content to navigate, it's easy to feel overwhelmed and indecisive. The sheer volume of choices can lead to decision fatigue, making it difficult to discern what truly captures our attention and sparks our curiosity.

In this age of information overload, curation plays a critical role. Platforms that effectively guide and filter content, helping users discover gems amid the vast sea of options, become invaluable allies in this digital exploration. Whether it's recommending personalized playlists on Spotify, surfacing relevant news articles on

DOI: 10.1201/9781003511267-1

Google, or highlighting trending videos on YouTube, curation helps us navigate the digital landscape with greater efficiency and satisfaction.

As the digital realm continues to evolve at breakneck speed, the volume and diversity of content will undoubtedly continue to expand. The challenge lies in harnessing this abundance effectively, leveraging curation and personalized recommendations to make informed choices and uncover the hidden treasures that await within the vast digital expanse.

To further illustrate the sheer scale of digital content, let's delve into some additional statistics. In 2022, TikTok users watched an astounding 100 billion videos per day, equivalent to 1.25 billion hours of footage consumed daily. This staggering figure highlights the platform's captivating nature and the immense amount of time users dedicate to engaging with its diverse content.

Social media platforms like Facebook and Instagram also contribute significantly to the ever-growing digital content landscape. In 2023, Facebook users generated an estimated 4 petabytes of data per day, equivalent to approximately 4 million hours of video content. This immense volume of data reflects the platform's role as a hub for sharing personal stories, experiences, and creative expressions.

The realm of digital content extends far beyond entertainment and social media. Educational platforms like Coursera and edX offer a vast library of online courses, encompassing a wide range of subjects from business and technology to art and history. With over 40 million students enrolled in these platforms, the demand for high-quality digital education is evident.

The growth of digital content has also transformed the way we consume news and information. Online news aggregators like Google News and Apple News provide users with personalized news feeds, tailored to their interests and preferences. This shift toward digital news consumption has democratized access to information, allowing users to curate their news diets and break free from traditional media gatekeepers. In essence, the sheer volume and diversity of digital content available today is a testament to the transformative power of technology. The ability to create, share, and consume content on an unprecedented scale has redefined the way we engage with information, entertainment, and education. As the digital realm continues to evolve, the boundaries of content creation and consumption will undoubtedly expand further, shaping the future of our interconnected world.

Envision a scenario where the vast troves of digital content, currently dispersed across the internet, were consolidated into a single, colossal repository, akin to a physical library housing an infinite collection of books, movies, music, and more. While the physical constraints of a library limit its capacity, the digital realm offers limitless storage potential. In this hypothetical digital library, the act of browsing and consuming content would undergo a radical transformation. Instead of endlessly scrolling through search results or navigating through multiple platforms, users would seamlessly access a centralized hub, where every conceivable piece of content is readily available at their fingertips.

Intelligent search algorithms would play a pivotal role in guiding users through this vast repository, enabling them to effortlessly discover content tailored to their interests and preferences. Advanced recommendation systems would analyze user behavior, preferences, and past consumption patterns to curate personalized content

streams, ensuring that users are constantly presented with material that resonates with their tastes.

In the pre-internet era, the world of product and service discovery was a vastly different landscape compared to the hyperconnected, information-rich environment we inhabit today. Businesses relied on traditional advertising methods to reach their target audience, and consumers had limited access to information and reviews before making purchasing decisions.

Newspapers, television, radio, and billboards were the primary channels for businesses to promote their products and services. Newspapers offered a mix of local and national advertisements, allowing businesses to reach a broad audience. Television commercials provided a more dynamic and engaging way to showcase products, while radio ads offered a targeted approach, reaching specific demographics based on station programming. Billboards, with their strategic placement along busy roads and highways, served as constant reminders of brands and their offerings.

Advertisements in the pre-internet era were primarily designed for mass appeal, aiming to capture the attention of a wide range of potential customers. Personalized advertising, tailored to individual preferences and interests, was not yet a common practice. This meant that consumers were often bombarded with advertisements that might not have been relevant to their specific needs or desires.

Without the vast repository of information available online, consumers in the pre-internet era faced challenges in making informed purchasing decisions. They relied on word-of-mouth recommendations, limited product information provided in advertisements, and personal experiences to evaluate products and services. This lack of comprehensive information could lead to impulsive decisions and buyer's remorse.

Publications like *Consumer Reports* played a crucial role in providing consumers with unbiased product reviews and comparisons. These magazines offered detailed evaluations of various products, helping consumers make informed choices based on performance, features, and price.

The advent of the internet and digital technologies revolutionized the way businesses advertise and consumers discover products and services. Personalized advertising, targeted marketing campaigns, and the abundance of online reviews have transformed the landscape of product and service discovery. While traditional advertising channels still hold some relevance, the digital realm has become the dominant force in shaping consumer behavior and purchasing decisions.

In the absence of the internet and its vast array of resources, consumers in the pre-internet era had to rely on more limited and personalized approaches to discover products and services. Two primary methods guided their choices: popularity-based selection and personal recommendations.

Word-of-mouth and box office success were significant indicators of popularity in the pre-internet era. Consumers often turned to friends, family, and peers for recommendations, seeking insights into popular products and services. Box office earnings served as a reliable measure of a movie's popularity, influencing moviegoers' choices. Similarly, popular music charts and bestseller lists guided consumers seeking music and books.

Personal feedback from trusted sources played a crucial role in consumer decision-making. Recommendations from friends, family, and experts were highly valued, as they provided personalized insights and opinions. Newspaper reviews, while not as personalized, offered a broader perspective on product quality and performance, influencing consumers' choices.

1.1.1 The Role of Recommender Systems in the Digital Age

In today's information-saturated world, consumers face an overwhelming array of choices when it comes to products, services, and content. To navigate this vast sea of options, recommender systems have emerged as indispensable tools, providing personalized recommendations tailored to individual preferences and interests.

Recommender systems, powered by sophisticated algorithms and machine learning techniques, analyze user behavior, preferences, and past consumption patterns to identify patterns and predict future choices. These systems act as digital surrogates for the trusted friends, family, and experts who traditionally guided consumer decisions in the pre-internet era.

Consider the example of YouTube. When users access the platform without logging in, they are presented with a curated selection of videos based on popularity in their geographic region. This popularity-based approach ensures that users are exposed to trending content that is likely to resonate with a broad audience.

However, once a user logs in, YouTube's recommender system kicks into action, analyzing their past viewing history, search preferences, and interactions with other users to generate a personalized feed of videos. This personalized approach ensures that users are presented with content that aligns with their unique interests and tastes, increasing the likelihood of engagement and satisfaction.

The same principle applies to other business-to-consumer (B2C) websites like Amazon and Spotify. Amazon's recommender system suggests products based on a user's past purchases, browsing behavior, and similar items purchased by other users with similar preferences. Spotify's recommender system suggests music based on a user's listening history, genre preferences, and similar artists enjoyed by other users.

Recommender systems essentially emulate the role of trusted friends, family, and experts in the digital space. They provide personalized recommendations, based on a deep understanding of individual preferences, just as trusted advisers would offer informed suggestions based on personal knowledge and experience.

This role is particularly crucial in today's digital world, where consumers are bombarded with an overwhelming amount of information and choices. Recommender systems help to cut through the noise, presenting users with curated content and products that are most likely to be relevant and engaging.

1.1.1.1 The Dual Importance of Recommender Systems

Recommender systems have become indispensable tools in today's digital landscape, serving as trusted guides for consumers and driving revenue growth for businesses. Understanding the significance of these systems is crucial for both consumers and businesses alike.

Recommender systems play a pivotal role in empowering consumers to navigate the vast ocean of products and services available online. Without these systems, consumers would face an overwhelming task of sifting through an endless sea of options, often lacking the information or guidance necessary to make informed decisions.

Effective recommender systems act as personal assistants, analyzing user behavior, preferences, and past consumption patterns to curate a selection of products and services that align with individual tastes and interests. This personalized approach eliminates the need for consumers to spend countless hours searching for relevant items, saving time and enhancing their overall shopping experience.

For businesses, recommender systems serve as powerful tools for driving revenue growth and customer engagement. By accurately suggesting relevant products and services to potential customers, businesses increase the likelihood of conversions and boost sales. This targeted approach not only enhances customer satisfaction but also optimizes marketing efforts, ensuring that advertising resources are directed toward the most receptive audience.

Moreover, recommender systems provide businesses with valuable insights into customer behavior and preferences. By analyzing user interactions with suggested items, businesses can identify trends, refine their product offerings, and tailor their marketing strategies to better meet customer needs. These insights can lead to improved customer retention, increased brand loyalty, and ultimately, enhanced profitability.

1.1.1.2 The Intertwined Benefits

The benefits of recommender systems extend beyond the individual consumer and business. These systems foster a symbiotic relationship between consumers and businesses, creating a more efficient and satisfying marketplace for both parties.

For consumers, recommender systems eliminate the guesswork involved in product discovery, leading to more informed purchasing decisions and a higher likelihood of finding products that truly meet their needs. This, in turn, benefits businesses by increasing customer satisfaction, reducing product returns, and fostering long-term customer relationships.

Recommender systems are continuously evolving, incorporating new data sources, algorithms, and machine learning techniques to improve their accuracy and personalization. As these systems become more sophisticated, they will play an even more integral role in shaping consumer behavior and influencing purchasing decisions.

The future of recommender systems lies in their ability to not only predict preferences but also to understand the underlying reasons behind those preferences. By delving into the motivations and behaviors of consumers, recommender systems can provide even more personalized and relevant recommendations, further enhancing the user experience and driving engagement.

In the realm of recommender systems, two distinct approaches have emerged: content-based filtering and collaborative filtering. While both aim to provide personalized recommendations to users, they differ significantly in their underlying principles and the data they utilize.

Content-based filtering draws inspiration from classical information retrieval techniques. This approach relies on analyzing the attributes and features of products

or services to identify similarities between them. By understanding the characteristics of items that a user has previously enjoyed or purchased, the recommender system can suggest similar items based on these shared characteristics.

For instance, a content-based filtering system might recommend movies to a user based on genre, actors, director, or plot descriptions. Similarly, it could recommend music based on artist, genre, tempo, or lyrics.

Despite its intuitive appeal, content-based filtering faces inherent limitations that hinder its effectiveness in real-world applications. One primary challenge is the requirement for extensive knowledge about both products and users. The system needs a detailed understanding of the attributes and characteristics of each item, as well as comprehensive information about user preferences and past consumption patterns.

In practice, acquiring and maintaining such a vast amount of data can be difficult and costly. Moreover, content-based filtering often struggles with the complexities of human preferences. It may fail to capture the nuances of taste and the serendipitous discoveries that users often seek.

Collaborative filtering, the dominant approach in modern recommender systems, takes a different perspective. Instead of relying on item descriptions and user profiles, it leverages the collective wisdom of users to make recommendations.

At the heart of collaborative filtering lies the concept of user similarity. This approach assumes that users with similar preferences are likely to enjoy similar products or services. By analyzing the ratings, choices, and past interactions of users, the system can identify groups of users who exhibit similar tastes.

For instance, a collaborative filtering system might recommend music to a user based on the listening habits of similar users. It could suggest movies based on the collective preferences of users who have watched similar films.

Collaborative filtering offers several advantages over content-based filtering. It does not require extensive knowledge about products or users, as it relies on the collective behavior of the user community. This makes it more adaptable to new items and users, and it can effectively capture the nuances of human preferences.

Moreover, collaborative filtering can uncover serendipitous recommendations that content-based filtering might miss. By analyzing the choices of similar users, the system can suggest items that might not be explicitly similar to those a user has enjoyed in the past, but still align with their overall tastes.

1.1.1.3 The Interplay of Recommender Systems

While content-based filtering and collaborative filtering have distinct strengths and weaknesses, they can also be combined to create more powerful recommender systems. Hybrid systems that incorporate both approaches can leverage the strengths of each to provide more comprehensive and personalized recommendations.

Content-based filtering can be used to initially filter the vast universe of items to a manageable subset, while collaborative filtering can then fine-tune the recommendations based on the user's specific preferences and the behavior of similar users.

The evolution of recommender systems from content-based filtering to collaborative filtering reflects the growing recognition of the power of collective intelligence.

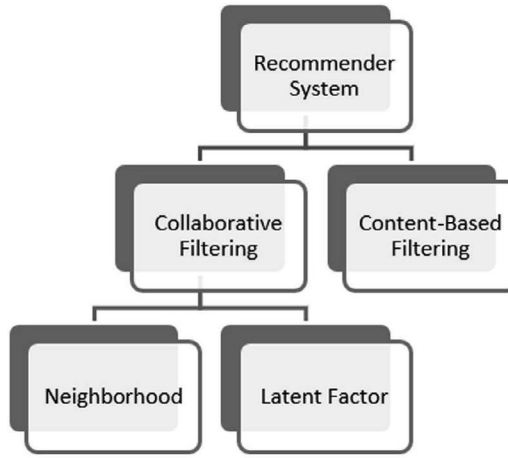

FIGURE 1.1 Taxonomy of Recommender Systems: Recommender systems can be broadly categorized into content-based and collaborative filtering. Collaborative filtering can be further subdivided into neighborhood-based methods and latent factor models.

By harnessing the wisdom of the user community, collaborative filtering has revolutionized the way we discover and consume products, services, and content.

As recommender systems continue to evolve, they will play an even more integral role in shaping our digital experiences, providing personalized guidance and serendipitous discoveries that enhance our lives (Figure 1.1).

1.2 CONTENTS OF THIS BOOK

1.2.1 Chapter 2

The next chapter delves into the realm of neighborhood-based models for recommender systems, a cornerstone of modern recommendation engines. These intuitive and straightforward models mimic real-world interactions where friends or acquaintances provide recommendations for movies, music, or shows. Their simplicity and ease of implementation have made them a mainstay in commercial recommender systems.

Neighborhood-based models can be categorized into two main approaches: user-based and item-based. User-based methods identify users with similar preferences and tastes to the target user and then recommend items that these similar users have rated highly. Conversely, item-based approaches focus on items rather than users, finding items that are similar to those the target user has enjoyed in the past.

Despite their simplicity, neighborhood-based models have proven to be remarkably effective in generating personalized recommendations. They are particularly well-suited for scenarios where user data is sparse or implicit, as they rely solely on the observed interactions between users and items.

While neighborhood-based models offer a straightforward and intuitive approach to recommendation systems, they also have limitations. Their performance can be hindered by cold-start problems, where new users or items lack sufficient data to establish meaningful relationships. Additionally, these models can be computationally expensive, especially for large datasets.

1.2.2 Chapter 3

In Chapter 3, we embark on a deeper exploration of ratings, delving into the inherent biases that can influence them. Bias, in this context, refers to the tendency for users or products to consistently receive positive or negative ratings, regardless of their actual quality. This phenomenon can significantly impact recommender systems, as ratings serve as the fundamental input for generating personalized recommendations.

User bias arises from various factors, including personal preferences, past experiences, and socio-demographic characteristics. For instance, a user might consistently rate movies highly in the action genre, while another user might favor romantic comedies. These inherent preferences can skew ratings, making it challenging for recommender systems to accurately assess the true quality of a movie.

Product bias, on the other hand, originates from factors inherent to the product itself, such as its genre, popularity, or brand reputation. For example, a product from a well-established brand might receive more favorable ratings simply due to brand association, even if its quality is comparable to that of a lesser-known brand.

The presence of bias in ratings can significantly hinder the effectiveness of recommender systems. Consider a scenario where a user consistently rates comedies highly, but the recommender system, influenced by user bias, primarily suggests action movies. This misalignment between the user's preferences and the system's recommendations can lead to dissatisfaction and a decline in user engagement.

Addressing bias in ratings is crucial for developing fair and effective recommender systems. By understanding the sources of bias and employing appropriate normalization techniques, we can ensure that recommender systems provide personalized recommendations that accurately reflect user preferences and product quality.

1.2.3 Chapter 4

Chapter 4 embarks on a journey into the realm of latent factor models for collaborative filtering, a powerful approach for generating personalized recommendations. Before delving into the intricacies of latent factor models, we'll first examine content-based filtering and its limitations, paving the way for the introduction of latent factor models.

Content-based filtering, a fundamental technique in recommender systems, relies on the characteristics of items to identify similar items for recommendation. For instance, if a user has enjoyed a movie with a specific genre, actor, or director, the system might recommend movies with similar attributes.

However, content-based filtering faces several challenges. It struggles with new items that lack sufficient descriptive information, a phenomenon known as the "cold-start"

problem. Additionally, it assumes that users' preferences are based solely on item attributes, overlooking the complex interplay of factors that influence user choices.

Latent factor models emerge as a sophisticated solution to the shortcomings of content-based filtering. These models assume that there exist underlying latent factors, unobserved features that capture the underlying preferences and characteristics of users and items.

To illustrate, consider a scenario where users rate movies. A latent factor model might infer that two movies, seemingly unrelated on the surface, are both associated with the latent factor of "action-packed thrillers." This latent connection enables the model to recommend movies that align with the user's preference for action-packed thrillers, even if they share no explicit attributes with previously rated movies.

Latent factor models exhibit remarkable mathematical prowess, often outperforming content-based and neighborhood-based models in terms of recommendation accuracy. However, their power comes at a cost: interpretability. The latent factors themselves are abstract and difficult to interpret, making it challenging to understand the underlying reasons behind specific recommendations.

Latent factor models represent a significant advancement in the field of recommender systems, offering a powerful and versatile approach for generating personalized recommendations. While their interpretability remains a challenge, their ability to uncover hidden patterns and connections between users and items makes them an invaluable tool for enhancing recommendation accuracy.

1.2.4 Chapter 5

In Chapter 5, we delve into the realm of metadata, exploring its potential to enhance the accuracy and performance of recommender systems. Metadata, encompassing product descriptions, user demographics, and other contextual information, provides a rich tapestry of insights that can refine recommendations and improve user experiences.

Metadata, often overlooked as mere supplementary data, holds immense value for recommender systems. Product descriptions, for instance, can reveal hidden attributes, genres, and themes that go beyond the explicit features of the product. Similarly, user demographics, such as age, location, and interests, can provide valuable clues about their preferences and potential biases.

By leveraging metadata effectively, recommender systems can transcend the limitations of traditional approaches that rely solely on item-to-item or user-to-user similarities. Metadata provides a deeper understanding of the context surrounding users and products, enabling the system to make more informed and personalized recommendations.

Graphical latent factor models (GLFMs) emerge as a powerful tool for incorporating metadata into recommender systems. These models introduce a graphical structure that captures the relationships between users, items, and metadata. By analyzing the graph, GLFMs can identify latent factors that not only reflect the intrinsic characteristics of users and items but also incorporate the influence of metadata.

The construction of the graph in GLFMs is crucial for extracting meaningful insights from metadata. One approach involves creating nodes for users, items, and metadata attributes and establishing edges based on their relationships. For instance,

an edge might connect a user to a movie they have watched or connect a movie to a genre to which it belongs.

GLFMs employ sophisticated algorithms to analyze the constructed graph, uncovering latent factors that capture the underlying relationships between users, items, and metadata. These latent factors, though abstract, represent the underlying preferences and characteristics that drive user behavior.

By incorporating metadata-driven latent factors into the recommendation process, GLFMs can significantly improve the accuracy and personalization of recommendations. The system can now consider not only the item itself but also the associated metadata, such as genre, product description, and user demographics, leading to more relevant and tailored recommendations.

Metadata, often overlooked as mere supplementary data, holds immense potential for enhancing the performance of recommender systems. By leveraging metadata effectively and employing graphical latent factor models, recommender systems can transcend traditional approaches, leading to more accurate, personalized, and context-aware recommendations.

1.2.5 Chapter 6

Chapter 6 delves into the critical concept of diversity in recommender systems, exploring the delicate balance between accuracy and diversification. While accuracy remains a cornerstone of recommendation algorithms, a sole focus on accuracy can lead to a phenomenon known as "filter bubbles," where users are confined to a narrow echo chamber of similar content and perspectives.

Accuracy-driven recommendation systems, often focused on maximizing prediction accuracy, can inadvertently restrict users' exposure to diverse viewpoints and experiences. This can lead to several negative consequences, including:

Limited Exploration: Users are less likely to encounter new and unfamiliar content, potentially hindering their personal growth and exposure to diverse perspectives.

Information Bias: Users are more susceptible to confirmation bias, receiving primarily information that reinforces their existing beliefs and opinions, potentially limiting their understanding of alternative viewpoints.

Homogeneity: Recommendation systems can contribute to a homogenization of content consumption, reducing the overall diversity of ideas and perspectives available to users.

Diversity in recommender systems encompasses two primary aspects:

Content Diversity: Recommending a variety of items from different genres, categories, or styles, broadening users' exposure to diverse content.

Perspective Diversity: Recommending items from different sources, creators, or viewpoints, providing users with a range of perspectives on various topics.

Incorporating diversity into recommendation algorithms offers several benefits:

Enhanced Exploration: Users are exposed to a wider range of content, encouraging them to explore new interests and expand their knowledge base.

Broader Perspective: Users encounter diverse viewpoints and perspectives, fostering a more comprehensive understanding of various topics and issues.

Reduced Bias: Recommendation systems become less susceptible to confirmation bias, providing users with a more balanced exposure to information.

The challenge lies in striking a balance between accuracy and diversity. Overemphasizing accuracy can lead to filter bubbles, while solely focusing on diversity can compromise the relevance and usefulness of recommendations.

Several techniques can be employed to achieve this balance:

Serendipity-Based Approaches: Introduce a small percentage of random or unexpected recommendations, encouraging users to explore beyond their usual preferences.

Context-Aware Diversification: Tailor recommendations to the specific context, such as time of day, location, or device, introducing elements of surprise and diversity.

Social Recommendations: Incorporate recommendations from friends, experts, or influential figures, providing users with diverse perspectives beyond their own.

Striking a balance between accuracy and diversity is crucial for developing fair, unbiased, and impactful recommender systems. By incorporating diversity into recommendation algorithms, we can foster a more inclusive and enriching user experience, exposing users to a broader range of content, perspectives, and ideas.

1.2.6 Chapter 7

Chapter 7 ventures into the realm of deep learning for recommender systems, exploring cutting-edge approaches that push the boundaries of traditional recommendation algorithms. These deep learning models harness the power of complex neural networks to uncover hidden patterns, extract meaningful insights, and deliver personalized recommendations with unprecedented accuracy and diversity.

The deep latent factor model (DLFM) emerges as a cornerstone of radical deep learning-based recommender systems. This model extends the traditional latent factor model by incorporating deep neural networks, enabling it to capture complex nonlinear relationships between users, items, and associated metadata.

The DLFM employs a multilayer neural network to learn latent factors from user–item interactions, metadata, and auxiliary information. These latent factors represent the underlying preferences and characteristics that drive user behavior, providing a deeper understanding of user preferences and interests.

Graphical latent factor models (GLFMs) introduce a graphical structure into deep latent factor models, capturing the intricate relationships between users, items, and metadata in a graph-based framework. This graphical representation allows the model to exploit the power of graph neural networks, enabling it to learn latent factors that not only reflect the intrinsic characteristics of users and items but also incorporate the influence of metadata and contextual information.

GLFMs employ sophisticated graph neural network algorithms to propagate information across the graph, effectively capturing the relationships between entities and extracting meaningful insights from the underlying graph structure. This approach leads to more accurate and personalized recommendations that are tailored to the specific context and preferences of each user.

Diversified deep learning models address the limitations of traditional recommendation algorithms that often prioritize accuracy over diversity. These models incorporate diversity metrics into the recommendation process, ensuring that users are exposed to a wider range of content and perspectives while maintaining a high level of accuracy.

One approach involves incorporating a diversity loss function into the model's training objective. This loss function penalizes the model for recommending items that are too similar to each other, encouraging it to explore a broader range of content and cater to users' diverse interests.

Another approach involves utilizing reinforcement learning techniques to train the recommendation model. In this reinforcement learning framework, the model receives rewards for recommending items that are both relevant to the user and contribute to overall diversity. This approach encourages the model to strike a balance between accuracy and diversity, providing users with a more enriching and multifaceted recommendation experience.

Deep learning-based recommender systems represent a significant step forward in the field of recommendation algorithms. These models, equipped with the power of deep neural networks and graph neural networks, can capture complex nonlinear relationships, extracting meaningful insights from graphical structures, and incorporating diversity metrics, leading to more accurate, personalized, and diverse recommendations that meet the evolving needs of modern users.

2 Neighborhood-Based Models

2.1 INTRODUCTION

Before the advent of streaming services like Netflix and Amazon Prime, the process of selecting a movie or book to consume was a more deliberate and often social affair. Unlike today, where algorithms and personalized recommendations curate our entertainment options, our choices were primarily guided by two key factors: expert reviews and word-of-mouth recommendations from friends and acquaintances.

The first method involved seeking guidance from established book and film critics, whose opinions held significant sway over the public's decision-making. Book reviewers, for instance, would provide insightful commentaries on newly released novels, carefully dissecting the plot, characters, and writing style without divulging crucial plot points that would spoil the reading experience. These reviews served as valuable tools for readers, helping them gauge whether a particular book aligned with their interests and literary preferences.

Similarly, when it came to choosing movies, recommendations from trusted friends and family members played a pivotal role. By understanding the tastes and preferences of our close circle, we could confidently rely on their opinions to steer us toward movies that were likely to resonate with us. This personalized approach to movie selection ensured that our cinematic experiences were more likely to be enjoyable and satisfying.

The concept of neighborhood-based models in the digital realm draws inspiration from this very notion of seeking recommendations from like-minded individuals. These models, employed by recommender systems, analyze user preferences and identify patterns among users with similar tastes. By understanding these commonalities, the system can effectively recommend movies, books, and other content that closely align with the user's interests, mimicking the way we would seek guidance from our trusted friends and book reviewers in the pre-streaming era.

In essence, neighborhood-based models aim to replicate the social dynamics of recommendation-seeking, leveraging the power of collective intelligence to enhance the user experience. By understanding the tastes and preferences of similar users, these models can provide personalized recommendations that are more likely to resonate with the individual's interests, fostering a sense of discovery and engagement within the digital realm.

Collaborative filtering (CF) is a technique that is used to make recommendations to users based on their past preferences and the preferences of similar users. CF is commonly used in recommender systems for e-commerce platforms, streaming services, and social media platforms.

DOI: 10.1201/9781003511267-2

Ratings-based CF is the most common type of CF. It relies on users providing ratings for items, such as products, movies, or music. These ratings are then used to calculate the similarity between users and items. Once the similarity between users and items is known, CF can be used to make recommendations to users.

EXAMPLE: AMAZON

Amazon is a popular e-commerce platform that uses ratings-based CF to make recommendations to users. When a user purchases an item on Amazon, they are asked to rate the item on a scale of 1 to 5. These ratings are then used to calculate the similarity between users and items. Once the similarity between users and items is known, Amazon can recommend items to users that are similar to items they have rated highly in the past.

EXAMPLE: NETFLIX

Netflix is a popular streaming service that uses ratings-based CF to make recommendations to users. When a user watches a movie or TV show on Netflix, they are asked to rate it on a scale of 1 to 5. These ratings are then used to calculate the similarity between users and items. Once the similarity between users and items is known, Netflix can recommend movies and TV shows to users that are similar to movies and TV shows they have rated highly in the past.

EXAMPLE: UBER

Uber is a popular ride-sharing service that uses ratings-based CF to make recommendations to users. When a user takes an Uber ride, they are asked to rate the ride and the driver on a scale of 1 to 5. These ratings are then used to calculate the similarity between users and drivers. Once the similarity between users and drivers is known, Uber can recommend drivers to users who are similar to drivers they have rated highly in the past.

At the heart of Netflix's recommendation system lies CF, a powerful technique that helps predict your preferences for movies and TV shows. CF algorithms analyze the vast trove of user ratings and viewing history to identify patterns and connections between users and items, essentially creating a network of shared tastes.

When you're browsing Netflix, the CF algorithm silently works behind the scenes, evaluating your past viewing habits and comparing them to those of other users. By analyzing ratings, watch history, and even factors like pause and rewind behavior, the algorithm builds a profile of your preferences.

To predict whether you'll enjoy a particular show, the CF algorithm doesn't just rely on your ratings. It considers the ratings and preferences of other users who share similar tastes with you. This approach, known as neighborhood-based CF, assumes that users who like similar things are likely to agree on whether they'll enjoy a particular show.

The CF algorithm essentially creates a virtual neighborhood of users with similar tastes and calculates the similarity between you and each of these neighbors. The higher the similarity, the more weight their ratings carry in predicting your rating.

Now, let's say you're considering watching a new show, "The Crown." The CF algorithm will look at the ratings and watch history of all your neighbors who have watched "The Crown." It will then calculate the average rating among these neighbors and use that as an estimate of your expected rating.

If the predicted rating is high, say above four out of five stars, the CF algorithm will confidently recommend "The Crown" to you. It believes that, based on your past preferences and the tastes of your similar neighbors, you're likely to enjoy this show.

However, if the predicted rating is low, say below three out of five stars, the CF algorithm will refrain from suggesting "The Crown." It concludes that, given your past viewing habits and the preferences of your neighbors, you're less likely to find this show appealing.

This collaborative approach, where CF takes into account the wisdom of the crowd, is what makes Netflix's recommendation system so effective. By leveraging the collective intelligence of its users, Netflix can tailor recommendations to your unique tastes and preferences, increasing the likelihood that you'll discover shows you'll love.

At the core of collaborative filtering lie two primary approaches: user-based CF and item-based CF. Each method takes a different approach to predicting your potential rating for a particular movie or show.

2.1.1 User-Based CF

In user-based CF, the algorithm identifies users with whom you have similar tastes, essentially creating a virtual neighborhood of like-minded individuals. By analyzing the ratings and viewing habits of these similar users, the algorithm can interpolate your expected rating for the movie or show in question.

The key advantage of user-based CF lies in its ability to capture the nuances of individual preferences. By considering the collective wisdom of users who share your tastes, the algorithm can make more personalized recommendations that align with your specific interests.

However, user-based CF can face challenges when dealing with new items or users for whom there is limited data. In such cases, the algorithm may struggle to find a sufficient number of similar users to make accurate predictions.

2.1.2 Item-Based CF

In contrast to user-based CF, item-based CF focuses on identifying items that are similar to the movie or show you're considering. By analyzing the ratings and watch history associated with these similar items, the algorithm can predict your potential rating.

Item-based CF excels at handling new items or users, as it doesn't rely on a large pool of similar users. Instead, it leverages the collective feedback for similar items to make informed recommendations.

However, item-based CF may overlook the unique preferences of individual users. By focusing solely on items, it may miss out on the subtle nuances of personal tastes that can influence a user's enjoyment of a particular movie or show.

2.2 USER-BASED APPROACH

Imagine a time before streaming services like Netflix revolutionized the way we consume entertainment. Before the days of personalized recommendations and instant access to a vast library of movies and shows, people relied heavily on word-of-mouth recommendations to guide their viewing choices.

In the pre-digital era, the decision to watch a particular movie often hinged on the opinions of trusted friends and family members. A casual conversation with colleagues at work, a chat with neighbors over the fence, or a lively discussion among friends at a social gathering could yield valuable insights into the worthiness of a new release.

This informal exchange of recommendations mirrored the core concept of user-based collaborative filtering (CF), a technique employed by modern recommendation systems to predict user preferences. Just as we would seek feedback from those we trust, user-based CF algorithms analyze the ratings and viewing habits of similar users to anticipate your potential enjoyment of a particular movie or show.

In the pre-digital era, the process of gathering recommendations was inherently social, fostering a sense of community and shared interests. Word-of-mouth recommendations carried a certain weight, as they reflected the authentic opinions of individuals we knew and respected.

Today, user-based CF algorithms replicate this social aspect of recommendation-seeking by leveraging the vast troves of data generated by online interactions. By analyzing ratings, reviews, and watch history, these algorithms create virtual neighborhoods of similar users, capturing the essence of those pre-digital conversations where we sought guidance from our trusted peers.

In essence, user-based CF emulates the very human instinct to seek recommendations from those we trust, transforming it into a powerful tool for personalization in the digital age. Just as we would seek feedback from our friends about the latest movie, user-based CF algorithms analyze the collective wisdom of similar users, guiding us toward movies and shows that are likely to resonate with our unique tastes.

	M1	M2	M3	M4	M5	M6
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

Consider a toy example. There are seven users, U1 to U7, and there are six movies, M1 to M6. The users have rated the movies on a scale of 1 to 5. Of course, not all the users would have watched all the movies. That is why some of the entries in

the user–item matrix are blanks. The CF wants to predict whether the M2 should be suggested to U4.

In user-based CF, identifying similar users is crucial for making accurate predictions about a user's potential rating for a particular movie or show. To identify these similar users, we need to establish a measure of "similarity" that quantifies how closely aligned two users' preferences are.

One common method for calculating similarity is the inverse absolute distance. This method measures the distance between two users' rating vectors, assigning lower values to users with closer ratings and higher values to those with more divergent ratings. The inverse of this distance is then used as the similarity measure, reflecting the idea that closer users are more similar, and thus their ratings should be given more weight in the prediction process.

The resulting similarity score will range from 0 to 1, with 0 indicating no similarity and 1 indicating perfect similarity. Users with higher similarity scores are considered more similar and their ratings will be given more weight in predicting the target user's rating.

Once we know the similarity measure to use, we will compute the distance between U4 and the rest of the users.

Between U4 and	Distance	Similarity
U1	1	1
U2	6	1/6
U3	3	1/3
U5	1	1
U6	1	1
U7	4	1/4

Of the six neighbors of U4, only U1, U3, U5 and U7 have rated M2; therefore, we will only consider these four neighbors. The final question now is, if we have the selected users (U1, U3, U5, and U7) and their ratings (3, 5, 1, and 2), how do we predict U4's rating on M2? The simplest choice is interpolation. To do so, we would need the interpolating weights.

Between U4 and	Distance	Similarity	Rating on M2	NN weight
U1	1	1	3	1
U2	6	1/6		
U3	3	1/3	5	1
U5	1	1	1	1
U6	1	1		
U7	4	1/4	2	1

There can be several choices for the weights. The simplest one would be nearest neighbor (NN) interpolation. In this, we will consider all the four users to be equally important. Therefore, all the interpolation weights will be 1. With this, the predicted rating of U4 on M2 will be:

$$\frac{3\times1+5\times1+1\times1+2\times1}{1+1+1+1}=\frac{11}{4}\approx3$$

We need to normalize the ratings by the sum of the weights (1+1+1+1=4) so that the predicting rating remains within the range of rating (in our case that is 5).

Between U4 and	Distance	Similarity	Rating on M2	Linear weight
U1	1	1	3	1
U2	6	1/6		
U3	3	1/3	5	1/3
U5	1	1	1	1
U6	1	1		
U7	4	1/4	2	1/4

The nearest neighbor may be the simplest, but is it the best? The similarity between U4 and U1, U5 is 1, which means U1 and U5 are very similar to U4. On the other hand, the similarity between U4 and U3 is 1/3 and between U4 and U7 is 1/4. Does it not make more sense to give more weightage to more similar users compared to the less similar ones? This can be captured using linear interpolation weights. In this, the weights are simply the similarities—the more similar two users are, the more the weights.

Using these linear interpolation weights, the predicted rating would come out to be:

$$\frac{3\times1+5\times\frac{1}{3}+1\times1+2\times\frac{1}{4}}{1+\frac{1}{3}+1+\frac{1}{4}}=2.38\approx2$$

Here we have used two types of interpolation—nearest neighbor and linear. One is free to choose more sophisticated interpolation techniques, but that is not usually done in practice.

In this scenario, NN interpolation predicts a rating of 3 and linear interpolation a rating of 2. Both are low. Therefore, in all likelihood, the CF would not recommend M2 to U4.

When using a user-based approach, one must remember to only consider neighbors that have rated the particular item we are interested in. For example, in this case, the item of interest was M2. Users 2 and 6 have not rated M2; therefore, we could not use any information from U2 and U6 while interpolating. Note that the ratings of U2 and U6 on M2 are unknown and not 0.

If we want to write down the algorithmic steps of user-based CF, it would be as follows:

1. Compute the similarity between the active users (in our example U4) and all other users.
2. Choose only those users who have rated the item of interest (in our case M2).
3. Choose interpolation weights (1 if nearest neighbor; similarity if linear).
4. Use the interpolation weights on the available ratings of the item of interest to find the weighted average. The weighted average is the predicted rating.

The choice of similarity measure in user-based CF is crucial for the effectiveness of the algorithm. While inverse absolute distance is a simple and intuitive measure, other more sophisticated measures can often yield better results.

Cosine similarity and Pearson correlation are two commonly used alternatives to inverse absolute distance. Cosine similarity measures the angle between two rating vectors, with a smaller angle indicating greater similarity. Pearson correlation, on the other hand, measures the linear relationship between two rating vectors, taking into account the magnitude and direction of the ratings.

In general, cosine similarity and Pearson correlation are considered more robust and accurate than inverse absolute distance, especially when dealing with rating data that may contain outliers or noise. Additionally, inverse Euclidean distance, which is based on the Euclidean distance between rating vectors, can also be an effective similarity measure in certain scenarios.

The choice of similarity measure should be considered carefully based on the characteristics of the data and the specific goals of the CF application. Empirical testing is often necessary to determine which measure provides the best results for a particular task.

2.3 ITEM-BASED APPROACH

The user-based approach to collaborative filtering (CF) is intuitively appealing as it mirrors our real-life scenario of seeking recommendations from friends and acquaintances with similar tastes. However, when exploring the world of CF, it's important to recognize that the item-based approach offers a complementary perspective, providing recommendations based on the similarity of items rather than users.

Just as we might ask a friend, "If I liked movie X, what other movies would you recommend?," the item-based approach analyzes the ratings and preferences associated with similar items to predict a user's potential enjoyment of a particular movie or show.

In the user-based approach, we focused on the rows, identifying similar users to predict User 1's rating for Movie C. However, we can also transpose the matrix and analyze the columns, effectively switching our focus to items rather than users.

Let us consider the same toy example as before.

	M1	M2	M3	M4	M5	M6
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

Our problem remains the same, that is, we want to find out what is U4's predicted rating on M2. But instead of finding similar users as before, we will find similar items. We need to find out the similarity between M2 and the other movies. As before, for our ease, we will be using inverse absolute distance as the similarity metric.

Between M2 and	Distance	Similarity
M1	3	1/3
M3	6	1/6
M4	5	1/5
M5	2	1/2
M6	4	1/4

Once we have the similarity between the item of interest (M2) and others, we will only look at the ratings of the active user (U4) on the different items. Note that U4 has only rated items M1, M4, M5, and M6 with ratings 2, 4, 3, and 4 respectively.

Between M2 and	Distance	Similarity	Ratings by U4	NN weight
M1	3	1/3	2	1
M3	6	1/6		
M4	5	1/5	4	1
M5	5	1/2	3	1
M6	4	1/4	4	1

As we did in the user-based approach, we can use NN weights for simplicity. With that, the predicted rating of U4 on M2 is computed to be

$$\frac{2\times1+4\times1+3\times1+4\times1}{1+1+1+1}=\frac{13}{4}\approx 3$$

Although simple, as argued before, NN may not be the best choice for accuracy. It gives equal weights to all the neighbors of M2. Ideally, one should give more importance to similar items. This is achieved by linear interpolation weights, which are simply the similarity values.

Between M2 and	Distance	Similarity	Ratings by U4	NN weight
M1	3	1/3	2	1/3
M3	6	1/6		
M4	5	1/5	4	1/5
M5	5	1/2	3	1/2
M6	4	1/4	4	1/4

Using linear interpolation, the predicted rating turns out to be

$$\frac{2\times\frac{1}{3}+4\times\frac{1}{5}+3\times\frac{1}{2}+4\times\frac{1}{4}}{\frac{1}{3}+\frac{1}{5}+\frac{1}{5}+\frac{1}{4}}=3.09\approx 3$$

Using both NN and linear interpolation, we see that the rating is predicted to be 3. On a 5-point scale, this is not a very high value. Therefore, in all likelihood, the CF will skip recommending M2 to U4.

As before, we must mention that we have inverse absolute distance since it is simple to compute. In practice, one would use cosine similarity or Pearson correlation.

In terms of algorithmic steps, the item-based approach can be succinctly expressed as:

1. Compute the similarity between the item of interest (M2) and all other items.
2. Consider only those items that the active user (U4) has rated.
3. Choose a type of interpolation weight.
4. The predicted rating is the weighted average of the active user's ratings on other items along with the similarity between the item of interest and other items.

2.3.1 Advantage of the Item-Based Approach

In the practical realm of recommendation systems, item-based collaborative filtering (CF) holds a distinct computational advantage over user-based CF. This advantage stems from the dynamic nature of user behavior and the resulting impact on similarity calculations.

In a typical recommendation system, users continuously interact with the platform, adding new ratings to existing items and occasionally exploring new ones. This constant influx of ratings poses a computational challenge for user-based CF, as it necessitates recalculating user similarities every time a new rating is added.

Consider a scenario where a large e-commerce platform with millions of users and products employs user-based CF. With every new purchase and subsequent rating, the system must recompute the similarities between the active user and all other users in the database. This continuous recalculation becomes computationally burdensome, especially in enterprise-level systems with massive user bases and product catalogs.

In contrast, item-based CF circumvents this computational hurdle by precomputing item similarities offline. This offline computation, typically performed daily or at regular intervals, analyzes the rating matrix to determine the similarity between all pairs of items. Once these item similarities are established, they remain static and can be efficiently utilized for future recommendations.

When a new rating is added, item-based CF simply multiplies the new rating with the existing item similarities to estimate the missing rating. This process is computationally efficient and avoids the need for real-time recalculation of user similarities.

To illustrate the computational advantage, consider the following example:

- **User-Based CF:** Suppose a user rates a new product. The system needs to recalculate the similarities between the active user and all other users (millions of computations).
- **Item-Based CF:** The system multiplies the new rating with the existing item similarities (a handful of computations).

The difference in computational complexity is evident, especially as the number of users and items grows. User-based CF's reliance on real-time user similarity calculations can lead to performance bottlenecks and hinder the scalability of the recommendation system.

Moreover, item-based CF offers several additional advantages:

- **New Item Recommendation:** Item-based CF excels at recommending new items to users. By leveraging the similarities between items, the system can identify items with similar characteristics to those the user has rated highly in the past.
- **Scalability:** Item-based CF is more scalable than user-based CF, as it doesn't require real-time recalculations of user similarities. This scalability is crucial for enterprise-level recommendation systems with massive user bases and product catalogs.

In summary, item-based CF offers a distinct computational advantage over user-based CF in practical recommendation scenarios. Its ability to precompute item similarities, handle cold-start problems, recommend new items, and scale effectively makes it a valuable tool for large-scale recommendation systems.

2.3.2 Neighborhood Selection

In the previous examples, we've explored both user-based and item-based collaborative filtering (CF) approaches, assuming that we'll consider all available neighbors for our calculations. However, in a practical scenario, this isn't always the most effective approach.

Consider the real-world analogy of seeking movie recommendations. We typically seek recommendations from friends and acquaintances whose tastes align with our own. We wouldn't approach random strangers on the street for movie suggestions.

Similarly, in CF, we want to focus on users who are highly similar to the active user. This means excluding users whose similarity scores are close to zero, as they would be akin to digital strangers. Including such neighbors would introduce noise into the CF system, potentially diminishing its accuracy.

There can be two obvious strategies for choosing neighbors. The first one is to select only those neighbors whose similarity is beyond a certain threshold. To illustrate, let us consider the previous example.

	M1	M2	M3	M4	M5	M6
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

Say, we want to predict U5's rating on M4. If we are using the item-based approach, the first step is to find the similarities between all M4 and the rest.

Between M4 and	Distance	Similarity
M1	4	1/4
M2	5	1/5
M3	5	1/5
M5	5	1/5
M6	2	1/2

We only want to consider neighbors above a certain threshold, say we fix it at 1/4 for our case. Then, we will only have two neighbors to consider—M1 and M6.

Between M4 and	Distance	Similarity	Rating by U5
M1	4	1/4	2
M6	2	1/2	3

If we are using linear interpolation, after neighborhood selection by thresholding, the predicted rating of U5 on M4 will come out to be

$$\frac{2\times\frac{1}{4}+3\times\frac{1}{2}}{\frac{1}{4}+\frac{1}{2}}=\frac{2}{\frac{3}{4}}\approx 3$$

When we use thresholding, the number of neighbors will be different for each active user. There is another way to select the neighbors using the top k-nearest neighbor for each active user. Let us understand using the previous example.

	M1	M2	M3	M4	M5	M6
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

We want to find out U3's predicted rating on M6 with the user-based approach and top-k nearest neighbor selection. Our first step is to find out the similarities between U3 and the rest.

Between U3 and	Distance	Similarity
U1	4	1/4
U2	3	1/3
U4	3	1/3
U5	5	1/5
U6	4	1/4
U7	5	1/5

Say, we want to select top two neighbors. Then we will consider only U2 and U4. But if we want to consider top three neighbors, we are in a dilemma; since both U1 and U6 have equal similarities with U3! This is bound to happen for one active user or the other in real-life scenarios where millions of users and items are there. In such a dilemma, we will consider more than k nearest neighbors (4 instead of 3). We will discuss later why in such a situation, considering less than k (in our case, 2 instead of 3) is not a good idea. Therefore, our selected neighbors are U1, U2, U4, and U6.

Between U3 and	Distance	Similarity	Rating on M6
U1	4	1/4	
U2	3	1/3	1
U4	3	1/3	4
U6	4	1/4	

Using linear interpolation, U3's predicted rating on M6 will be

$$\frac{1\times\frac{1}{3}+4\times\frac{1}{3}}{\frac{1}{3}+\frac{1}{3}}=\frac{5}{2}\approx 3$$

When we have to incorporate neighborhood selection in user-based CF, the algorithm will be as follows:

1. Compute the similarity between the active users and the rest of the users.
2. Using either thresholding or top-k nearest neighbor, select the neighbors of the active user.
3. Of these neighbors, only consider those users who have rated the item of interest.
4. Choose the type of interpolation and the corresponding weights.
5. Compute the weighted average of the available ratings from selected users to predict the required rating.

For item-based neighborhood selection, the algorithm is as follows:

1. Compute the similarity between the item of interest and the rest of the items.
2. Using either thresholding or top-k nearest neighbor, select the neighbors of the item of interest.
2. Consider only those items that the active user has rated.
3. Choose the type of interpolation and the corresponding weights.
4. Compute the weighted average of the available ratings from selected users to predict the required rating.

2.3.3 Coverage

In the pursuit of accurate recommendations, collaborative filtering (CF) algorithms strive to identify the most relevant neighbors for making predictions. This process of neighborhood selection is a delicate balancing act, as there are inherent trade-offs to consider.

As the threshold for selecting neighbors increases, meaning we consider only the top-k most similar neighbors, the likelihood of introducing noise into the calculations decreases. This is because we are excluding users or items with lower similarity scores, which are more likely to represent irrelevant or misleading preferences.

Consequently, with a stricter threshold, the recommendations are more likely to reflect the true preferences of the active user, leading to improved accuracy. However, there is a downside to this approach. By limiting the pool of neighbors, we increase the risk of encountering a situation where none of the selected neighbors have ratings for the item of interest.

This situation, known as the cold-start problem, is particularly challenging in CF algorithms. When there are no relevant neighbors with ratings for the item, the algorithm struggles to make an accurate prediction. This can lead to gaps in recommendations, especially for new items or users with limited data.

To mitigate the cold-start problem, CF algorithms often employ a hybrid approach, combining neighborhood-based techniques with other methods, such as content-based filtering. Content-based filtering analyzes the characteristics of the item itself, such as genre, synopsis, or user reviews, to make recommendations when there is insufficient data from neighbors.

The choice of threshold for neighbor selection is crucial for balancing the trade-offs between accuracy and coverage. A higher threshold might improve accuracy but reduce coverage, while a lower threshold might increase coverage but introduce noise and reduce accuracy.

In practice, the optimal threshold is often determined empirically, through experimentation and evaluation of different threshold values using various metrics, such as precision, recall, and F1 score. The specific threshold that works best depends on the characteristics of the data, the specific application, and the desired balance between accuracy and coverage.

In conclusion, neighborhood selection in CF algorithms is a balancing act between accuracy and coverage. A higher threshold for selecting neighbors can improve accuracy but reduce coverage, while a lower threshold can increase coverage but introduce noise and reduce accuracy. The optimal threshold is determined empirically, considering the characteristics of the data, the specific application, and the desired balance between accuracy and coverage.

	M1	**M2**	**M3**	**M4**	**M5**	**M6**
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

In this example, say we want to predict U1's rating on M4 using the item-based approach. Therefore, we compute the similarities between M4 and the rest.

Between M4 and	**Distance**	**Similarity**
M1	4	1/4
M2	5	1/5
M3	5	1/5
M5	5	1/5
M6	2	1/2

Now say we decided to choose the top two neighbors or select a similarity threshold of 1/4. In either case, the selected neighbors would be M1 and M6.

In the next step, we need to see what U1 has rated on M1 and M6. Unfortunately, U1 hasn't rated either. Therefore, in this case, we will not be able to compute the rating!

Neighborhood-based models are also called memory-based techniques. This is because they have to remember all the ratings. One can see that these techniques are easy to interpret and analyze and hence are widely used in commercial systems.

BIBLIOGRAPHY

Breese, J.S., Heckerman, D. and Kadie, C., 1998, July. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth ACM conference on uncertainty in artificial intelligence* (pp. 43–52). ACM.

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th ACM international conference on World Wide Web* (pp. 285–295). ACM.

3 Ratings

3.1 INTRODUCTION

In the realm of user interactions and preferences, ratings play a pivotal role in gauging sentiments and driving recommendations. They serve as a bridge between users and systems, providing valuable insights into what users like, dislike, or simply find engaging. Ratings manifest in two primary forms: explicit ratings and implicit ratings.

Explicit ratings are those that users directly provide through conscious actions or statements. They represent a straightforward expression of preference, often conveyed through numerical scales, star ratings, or simple thumbs up/down gestures. Examples of explicit ratings abound in our digital experiences. On e-commerce platforms like Amazon, users rate products on a five-star scale, offering a direct assessment of product quality and satisfaction. Similarly, in ride-sharing services like Uber, passengers provide star ratings to evaluate their drivers' performance. And in entertainment platforms like Netflix or YouTube, users express their preferences through thumbs up/down ratings, shaping their personalized recommendations.

Explicit ratings offer several advantages, including their simplicity and direct connection to user preferences. They are readily collected, easily interpreted, and directly reflect the user's stated opinion. This makes explicit ratings particularly valuable for recommendation systems, where they provide a clear signal for predicting user preferences and recommending relevant items.

While explicit ratings provide a direct window into user preferences, they often fall short in capturing the full spectrum of user interactions and engagement. This is where implicit ratings come into play. Implicit ratings are inferred from user behavior and interactions, offering a more nuanced and comprehensive understanding of user preferences. They are not explicitly provided by users but rather derived from their actions, such as clicks, views, purchases, browsing history, time spent on a particular item, and even social media engagement.

Implicit ratings provide a richer tapestry of user preferences, capturing not only stated preferences but also observed behaviors. They reveal patterns and trends that may not be explicitly expressed, allowing for a deeper understanding of user motivations and interests. For instance, in Netflix, if a user watches a full movie without rating it, it can be inferred that they enjoyed the movie despite not providing an explicit rating. Similarly, on Amazon, if a user spends a considerable amount of time browsing a particular product page, it suggests a level of interest that may not be reflected in an explicit rating.

While both explicit and implicit ratings provide valuable insights into user preferences, they differ in their ease of acquisition. Explicit ratings often require active participation from users, and as a result, they tend to be sparse. Users are not always inclined to rate every product or service they encounter, leading to a limited pool of explicit data.

 DOI: 10.1201/9781003511267-3

Implicit ratings, on the other hand, are easier to collect and more abundant. They are passively gathered by observing user behavior, leaving a digital trail of interactions that can be analyzed to infer preferences. This makes implicit ratings a more comprehensive source of data, providing a broader picture of user preferences.

The choice between explicit and implicit ratings often depends on the specific application and the availability of data. Explicit ratings offer simplicity, direct connection to user preferences, and ease of interpretation, making them valuable for recommendation systems. Implicit ratings, on the other hand, provide a more comprehensive and unbiased view of user preferences, capturing natural behavior and overcoming the limitations of explicit ratings.

In practice, many systems employ a combination of both explicit and implicit ratings to leverage the strengths of each approach. By combining the directness of explicit ratings with the richness of implicit ratings, systems can gain a deeper understanding of user preferences and provide more personalized recommendations.

Ratings, whether explicit or implicit, serve as a symphony of user preferences, revealing the delicate balance between stated opinions and observed behaviors. They provide a valuable lens through which we can understand user interests, predict their future choices, and enhance their digital experiences. As we continue to navigate the ever-evolving digital landscape, ratings will remain an essential tool for gauging user preferences and shaping personalized interactions.

While implicit ratings offer the convenience of abundant data collection, they come with a significant drawback: their lack of specificity, particularly in capturing negative sentiment. This limitation poses a challenge for collaborative filtering (CF) systems, which rely on accurate user preferences to generate meaningful recommendations.

Explicit ratings, on the other hand, provide a clearer picture of both positive and negative sentiments. When a user actively provides a rating, whether it's a thumbs up/down, a star rating, or a numerical score, they are explicitly stating their preference or dislike. This direct expression of sentiment allows CF systems to make more informed decisions about user preferences and recommend items that align with their stated tastes.

Consider the example of Amazon product ratings. If a user gives a product a one-star rating, it's a clear indication that they dislike the product. This negative sentiment is crucial for CF systems to avoid recommending similar products to that user. Similarly, a five-star rating signifies strong approval, providing valuable information for recommending similar products that the user is likely to enjoy.

In contrast, implicit ratings, such as clicks or browsing history, often lack this level of specificity. While a click on a movie advertisement may suggest interest, it doesn't necessarily mean the user dislikes the movie if they don't click. The omission of a click could be due to various reasons, such as lack of interest, distraction, or simply not having the time to engage.

This ambiguity in implicit ratings makes it difficult to accurately gauge negative sentiment. CF systems that rely heavily on implicit ratings may struggle to identify items that users dislike, potentially leading to recommendations that miss the mark.

Given the limitations of implicit ratings in capturing negative sentiment, CF systems have largely opted for explicit ratings as their primary source of user preferences.

Explicit ratings provide a clearer signal of user likes and dislikes, enabling CF systems to make more informed recommendations.

While implicit ratings offer a richer tapestry of user behavior, their lack of specificity in capturing negative sentiment outweighs their potential benefits in CF applications. Until implicit ratings can be refined to accurately capture negative sentiment, explicit ratings will remain the preferred choice for CF systems seeking to provide personalized recommendations that align with user preferences.

3.2 BIASES AND BASELINE CORRECTION

	M1	M2	M3	M4	M5	M6
U1	3	1	2	2	3	1
U2	5	3	3	4	4	3
U3	4	2	3	3	4	2
U4	5	1	1	1	5	5
U5	4	1	3	4	5	2

Consider this example. There are five users (U1 to U5) and six movies (M1 to M6). The rating is on a 5-point scale, where 1 means very bad and 5 means very good. Let us try to analyze the pattern for each user.

Note that U1 rates all the movies between 1 and 3; U2 rates between 3 and 5; U3 rates between 2 and 4; U4 uses the entire range of ratings. It appears that U1 is a very critical viewer; he/she rates everything on the lower side of the scale. Even if U1 likes the movie, he/she is not ready to rate it above 3. At the other end of the scale are the ratings of U2; the user rates everything between 3 and 5. It seems that user U2 is of an exuberant nature. U3 rates everything moderately. If he/she likes a movie, it is rated 4 and if not, 2; this user's mediocre rating is 3. User U4 rates everything on a binary scale; he/she either likes a movie and rates it 5 or doesn't like the movie and rates it 1. User U5 uses the entire range and rates movies between 1 and 5.

In the realm of collaborative filtering (CF), user ratings play a pivotal role in shaping recommendation algorithms. However, beneath the surface of these ratings lies a subtle yet significant factor that can skew the recommendations: bias. Bias can manifest in two distinct forms: user bias and item bias.

3.2.1 User Bias: The Tinted Lens of Personal Preferences

User bias arises from the inherent subjectivity of user preferences. Each user brings their unique set of tastes, experiences, and expectations to the table, influencing how they perceive and rate items. This inherent subjectivity leads to consistent patterns in user ratings, where certain users tend to rate items consistently higher or lower than their global averages.

For instance, a user who enjoys action-packed movies may consistently rate action films higher than their global averages, while a user who prefers romantic comedies

may consistently rate them lower. These consistent patterns in user ratings reflect their personal biases, shaping the recommendations they receive.

3.2.2 Item Bias: The Halo Effect of Popularity and Disrepute

Item bias, on the other hand, stems from the inherent characteristics of items themselves. Certain items possess a reputation that precedes them, influencing how users perceive and rate them. For example, critically acclaimed movies like "Lord of the Rings" or "Titanic" often receive high ratings, regardless of individual user preferences. This phenomenon is known as positive item bias, where items are consistently rated higher than the global average.

Conversely, items that are widely considered to be poorly made or unpopular, such as the movie "Dragonball: Evolution," may consistently receive low ratings. This is known as negative item bias, where items are consistently rated lower than the global average.

3.2.3 The Impact of Bias on Collaborative Filtering

User bias and item bias can significantly impact the effectiveness of CF algorithms. If CF algorithms fail to account for these biases, they may recommend items that are not aligned with a user's true preferences. For instance, a CF algorithm that ignores user bias may recommend action films to a user who prefers romantic comedies, leading to dissatisfaction and a decline in user engagement.

A rating consists of two parts. One part is called the baseline. This is independent of the interaction between the user and the item. It is purely defined by the user's and item's biases. The other part is the results of the interaction between the user and the item. This includes the part we learned in the previous chapter.

We will introduce mathematical formalism. User i's rating on item j, is expressed as $r_{i,j}$. Based on what we have discussed so far,

$$r_{i,j} = \text{baseline} + \text{interaction} \tag{1}$$

Biases are basically deviations from the global mean. Therefore, we can further break down the baseline as

$$r_{i,j} = \mu + b_i + b_j + \text{interaction} \tag{2}$$

Here μ is the global mean of the entire dataset, that is, simply put it is the total of all ratings in the dataset divided by the number of available ratings. This is easy to compute. The user's bias is b_i and the item's bias is b_j. If b_i is negative, it means that the user has a negative bias and rates all movies on the lower side of the scale; if it is positive, it means the user has a positive bias and rates everything on the higher side. Similarly, if b_j is negative, it means that the item has a negative bias while a positive value of b_j indicates positive bias.

Unlike the global mean, the user and item biases are not very easy or straightforward to compute. Once the biases are estimated for all users and items, baseline

correction can be done to remove these from the ratings, so that the remainder of the rating can be treated solely as interaction.

Let us take a toy example of three users and three items.

	M1	M2	M3
U1	3	1	2
U2	5	3	3
U3	4	1	3

$$\text{The global mean is } \mu = \frac{3+1+2+5+3+3+4+1+3}{9} \approx 2.8$$

Let us assume that the M1 has a positive item bias, M2 has a negative item bias, and M3 is neutral, that is, it does not have any bias. We also assume that U1 has a negative bias, U2 has a positive bias, and U3 is neutral, that is, it has a bias of 0. Say we have estimated the biases (we will learn it in a minute), and they come out to be

User		Item	
b_1	–.7	b_1	–.5
b_2	.3	b_2	.5
b_3	0	b_3	0

Once we estimate the biases, we do baseline correction, that is, we remove the baseline from the ratings—this will get us the z-scores. This will be $z_{i,j} = r_{i,j} - \left(\mu + b_i + b_j\right)$

$$z_{1,1} = 3 - \left(2.8 - .7 - .5\ \right) = 1.4$$
$$z_{1,2} = 1 - \left(2.8 - .7 + .5\ \right) = -1.6$$
$$z_{1,3} = 2 - \left(2.8 - .7 + 0\ \right) = -.1$$
$$z_{2,1} = 5 - \left(2.8 + .3 - .5\ \right) = 2.4$$
$$z_{2,2} = 3 - \left(2.8 + .3 + .5\ \right) = -.6$$
$$z_{2,3} = 3 - \left(2.8 + .3 + 0\ \right) = -.1$$
$$z_{3,1} = 4 - \left(2.8 + 0 - .5\ \right) = 1.7$$
$$z_{3,2} = 1 - \left(2.8 + 0 + .5\ \right) = -2.3$$
$$z_{2,3} = 3 - \left(2.8 + 0 + 0\ \right) = .2$$

The $z_{i,j}$ values are the ratings after baseline correction. Note that they can have positive or negative values. While predicting ratings, instead of using the raw ratings, one is supposed to use the baselines instead since they are accounted for biases.

3.2.4 Baseline Estimation

We have discussed what biases are and how to do baseline correction once we know the biases. Now, we will learn how the biases are estimated from the data. For each user–item pair, we know that

$$r_{i,j} = \mu + b_i + b_j + \text{interaction}$$

Our objective is to estimate the user and item biases. For that, we frame the following optimization problem—

$$\min_{b_i, b_j} \sum_i \sum_j \left\| r_{i,j} - \left(\mu + b_i + b_j\right) \right\|_2^2 + \lambda_1 \sum_i \left\| b_i \right\|_2^2 + \lambda_2 \sum_j \left\| b_j \right\|_2^2 \tag{3}$$

The summation over i and j indicates that we will use all the available data to estimate the user and item biases. There also two regularization terms— $\sum_i \left\| b_i \right\|_2^2$ and $\lambda_2 \sum_j \left\| b_j \right\|_2^2$. Let us try to understand what would happen without these two terms. The expression would be: $\min_{b_i, b_j} \sum_i \sum_j \left\| r_{i,j} - \left(\mu + b_i + b_j\right) \right\|_2^2$. When one solves for b_i's and b_j's from this optimization problem, those biases would completely explain the rating, that is, there would be no component for the interaction in equation (1)/(2). This is the reason one needs the regularization terms. The Euclidean norm regularization ensures that the magnitudes of the values (b_i' and b_j) are small. This, in turn, guarantees that the bias component remains small and there is some component left for interaction.

The values of λ_1 and λ_2 determine the trade-off between the baseline component of equation (1) and the interaction component. The higher their values, the smaller the magnitudes of baseline component, and consequently, the interaction component will be higher. Unfortunately, there is no principled way to estimate the values of the regularization parameters; one has to fix them via trial and error or cross validation.

The problem (3) is convex. However, it cannot be solved in a closed form. The easiest approach to solve it is by alternately updating b_i's and b_j's using gradient descent. The iterative updates for solving for the biases is given by

$$\begin{aligned}
e &= r_{i,j} - \hat{r}_{i,j} \\
b_i^k &\leftarrow b_i^{k-1} + \gamma\left(e - \lambda_1 b_i^{k-1}\right) \\
b_j^k &\leftarrow b_j^{k-1} + \gamma\left(e - \lambda_2 b_j^{k-1}\right) \\
\hat{r}_{i,j} &= \mu + b_i^k + b_j^k
\end{aligned}$$

Here "k" indicates the iteration number. Obviously, such an iterative solution necessitates an initialization. As the problem is convex, any initialization would work in theory.

Although the exact solution to (3) is by far the best, in practice the following simpler (closed form) formula suffices.

$$b_i = \frac{\sum_{i \in R(j)} (r_{i,j} - \mu)}{\lambda_1 + |R(j)|} \quad b_j = \frac{\sum_{j \in R(i)} (r_{i,j} - \mu - b_i)}{\lambda_2 + |R(i)|} \tag{4}$$

Here, $|R(j)|$ indicates the number of ratings provided on the j^{th} item and $|R(i)|$ indicates the number of ratings provided by the i^{th} user. As before, λ_1 and λ_2 are for regularization. Without the regularization parameters, the estimated user and item biases could completely explain the ratings, without any room for the interaction component.

To understand baseline estimation, let us take a concrete example.

	M1	M2	M3	M4	M5
U1		3	5		2
U2	4		4	3	
U3	2			4	3
U4	3	1	3		
U5		2		4	2

With this user–item matrix, we will estimate the user and item biases. We will assume that $\lambda_1 = 2$ and $\lambda_2 = 2$.

The first step is to find out the global mean.

$\mu = 45$ (sum of all ratings)/15 (total number of ratings) = 3

Using (4) we can find out the user and item biases.

Item biases

$$b_1 = \frac{(4-3)+(2-3)+(3-3)}{2+3} = 0$$
$$b_2 = \frac{(3-3)+(1-3)+(2-3)}{2+3} = \frac{-3}{5}$$
$$b_3 = \frac{(5-3)+(4-3)+(3-3)}{2+3} = \frac{3}{5}$$
$$b_4 = \frac{(3-3)+(4-3)+(4-3)}{2+3} = \frac{2}{5}$$
$$b_5 = \frac{(2-3)+(3-3)+(2-3)}{2+3} = \frac{-2}{5}$$

User biases

$$b_1 = \frac{(3-3+\frac{3}{5})+(5-3-\frac{3}{5})+(2-3+\frac{2}{5})}{2+3} = \frac{\frac{7}{5}}{5}$$
$$b_2 = \frac{(4-3-0)+(4-3-\frac{3}{5})+(3-3-\frac{2}{5})}{2+3} = \frac{1}{5}$$
$$b_3 = \frac{(2-3-0)+(4-3-\frac{2}{5})+(3-3+\frac{2}{5})}{2+3} = 0$$
$$b_4 = \frac{(3-3-0)+(1-3+\frac{3}{5})+(3-3-\frac{3}{5})}{2+3} = \frac{-2}{5}$$
$$b_5 = \frac{(2-3+\frac{3}{5})+(4-3-\frac{2}{5})+(2-3+\frac{2}{5})}{2+3} = \frac{-\frac{2}{5}}{5}$$

With these, we can try to estimate the baseline component of the missing ratings. Note that these will not be the full ratings as we are not considering the interaction component in the computations.

	M1	M2	M3	M4	M5
U1	3+0+7/25	3	5	3+2/5+7/25	2
U2	4	3–3/5+1/5	4	3	3–2/5+1/5
U3	2	3–3/5+0	3+3/5+0	4	3
U4	3	1	3	3+2/5–2/5	3–2/5–2/5
U5	3+0–2/25	2	3+3/5–2/25	4	2

In practical problems, one would ideally estimate the biases. Remove them from the ratings and get the z-scores. Instead of the raw ratings, we will use the z-scores for predicting the missing values with either item-based or user-based approach. Once the z-scores are predicted, we will get the corresponding ratings by adding the baselines to it.

3.3 SIGNIFICANCE WEIGHTING

	M1	M2	M3	M4	M5
U1		3	4		
U2	2	2		5	3
U3	1	5	1		2

Let us consider this toy example. Say, we want to predict U2's rating on M3. Using a user-based approach, our first task is to find out the similarities between U2 and U1 and between U2 and U3. Using inverse absolute distance as the similarity measure, it comes out to be

$$S_{2,1} = \frac{1}{3-2} = 1;\ S_{2,3} = \frac{1}{(2-1)+(5-2)+(3-2)} = \frac{1}{5}$$

The similarity between U2 and U1 is high, compared to similarity between U2 and U3. Therefore, based on what we have learned before, U1 should be the nearest neighbor.

When we navigate the vast expanse of online marketplaces like Amazon, our decisions are often guided by the wisdom of the crowd. We scrutinize product reviews, seeking insights from fellow shoppers who have ventured before us. In this realm of collective knowledge, do we place greater trust in a product with a handful of five-star ratings or one with a multitude of four-star ratings? The answer, almost unanimously, is the latter. This instinctive preference stems from our inherent belief in the power of numbers—the more data points, the more reliable the assessment.

This same principle extends to the realm of collaborative filtering (CF), a recommendation algorithm that relies on user–item interactions to predict user preferences. In CF, we seek to identify users with similar tastes and leverage their preferences to recommend items that the active user might enjoy. However, a crucial aspect often overlooked is the significance of co-rated items—the items that two users have both rated.

Consider an example where two users, U2 and U1, share a high similarity score based on their ratings of a single item. Meanwhile, user U2 shares a lower similarity score with another user, U3, but they have three co-rated items. While the similarity measure indicates a strong connection between U2 and U1, it fails to capture the deeper understanding gained from their shared ratings of three items. This is where significance weighting comes into play.

Significance weighting is a technique that assigns varying weights to co-rated items, highlighting the importance of these interactions in shaping user preferences. By assigning greater weight to items that have been rated by multiple users, CF algorithms can gain a more nuanced understanding of user similarities and provide more accurate recommendations.

The rationale behind significance weighting is rooted in the notion that co-rated items provide a richer tapestry of information about user preferences. When users rate multiple items together, it suggests a deeper level of engagement and a more refined understanding of their preferences. By amplifying the signal from these co-rated items, CF algorithms can better discern genuine user similarities and avoid misleading inferences based on a limited number of ratings.

Incorporating significance weighting into CF algorithms offers several benefits:

1. **Improved Recommendation Accuracy:** By emphasizing the importance of co-rated items, CF algorithms can identify more relevant users and provide recommendations that better align with the active user's true preferences.
2. **Mitigating Data Sparsity:** In situations where user–item interactions are limited, significance weighting can amplify the signal from the available data, leading to more robust recommendations.
3. **Capturing Nuances in User Preferences:** Significance weighting allows CF algorithms to delve deeper into user similarities, capturing subtle nuances that may be overlooked by traditional similarity measures.

In conclusion, significance weighting serves as a powerful tool for enhancing the effectiveness of CF algorithms. By acknowledging the importance of co-rated items and assigning them varying weights, CF algorithms can gain a more comprehensive understanding of user preferences and provide recommendations that are both relevant and nuanced.

The formula for predicted rating in the user-based approach can be expressed as

$$r_{a,ii} = \frac{\sum_{i \in N_a} s_{i,a} r_{i,ii}}{\sum_{i \in N_a} s_{i,a}} \tag{5}$$

Here, "*a*" denotes the active user and "*ii*" the item of interest. N_a denotes the neighborhood of the active user and "*i*" denotes the neighbor.

In significance weighting, the similarity is further weighted by a factor of *n/N*, where *N* is fixed a priori and *n* is the number of co-rated items between the active user and its neighbor. The objective is to penalize neighbors who have few co-rated items. When *n* is smaller than *N*, it would reduce the importance of the similarity for that neighbor. When *n* is greater than or equal to *N*, full weightage is given to that neighbor. The definition of significance weight is

$$sig_{i,a} = \begin{cases} n/N & n < N \\ 1 & otherwise \end{cases} \tag{6}$$

The essence is to give more importance to neighbors who have a lot of co-rated items and penalize those who don't. With significance weighting, the formula for user-based approach becomes

$$r_{a,ii} = \frac{\sum_{i \in N_a} s_{i,a} r_{i,ii} sig_{i,a}}{\sum_{i \in N_a} s_{i,a} sig_{i,a}} \tag{7}$$

Let us do one example of rating prediction with significance weights.

	M1	M2	M3	M4	M5	M6	M7
U1	1	0			0	1	
U2		1	0	1	1		0
U3	0		0		0	1	
U4			1	0		0	1

Say, we want to find out U2's rating on M6. Here we have assumed that the ratings are binary. The first step is to find out the similarities. For that, we will use the inverse Hadamard distance.

Since it is a toy example, for significance weights we will have N = 3.

Between U2 and	Similarity	Significance weight	Rating on M6
U1	1/2	2/3	1
U3	1	2/3	1
U4	1/3	1	0

Using the formula for significance weighting, the predicted rating is

$$r_{2,6} = \frac{\frac{1}{2}\times\frac{2}{3}\times 1 + 1\times\frac{2}{3}\times 1 + \frac{1}{3}\times 1\times 0}{\frac{1}{2}\times\frac{2}{3} + 1\times\frac{2}{3} + \frac{1}{3}\times 1} = \frac{\frac{1}{3}+\frac{2}{3}}{\frac{1}{3}+\frac{2}{3}+\frac{1}{3}} = \frac{3}{4} \approx 1$$

Here, we have shown significance weighting in the context of user-based approach. The same can be applied in the item-based approach as well. The formula for predicted rating using the item-based approach is

$$r_{a,ii} = \frac{\sum_{j\in N_{ii}} s_{j,ii} r_{a,ii}}{\sum_{j\in N_{ii}} s_{j,ii}} \tag{8}$$

Here, N_{ii} denotes the neighborhood of the item of interest, j denotes the j^{th} item, $s_{j,ii}$ is the similarity between the j^{th} neighbor and the item of interest.

In the item-based approach, with significance weighting, the formula will be

$$r_{a,ii} = \frac{\sum_{j\in N_{ii}} s_{j,ii} r_{a,ii} sig_{j,ii}}{\sum_{j\in N_{ii}} s_{j,ii} sig_{j,ii}} \tag{9}$$

The significance weights will be defined as

$$sig_{j,ii} = \begin{cases} n/N & n < N \\ 1 & otherwise \end{cases} \tag{10}$$

Where N is a fixed number and n is the number of users who have rated both items j and ii.

Let us do an example with the item-based approach.

	M1	M2	M3	M4
U1		5	2	4
U2	1		5	
U3		3		3
U4	2		2	1
U5	4	3		
U6			3	5

Our goal is to predict M1's rating on U1 using the item-based approach. The first step is to define the similarities. Here, we will use the inverse absolute distance. For significance weighting, we will keep N = 2.

Between M1 and	Similarity	Significance weight	Rating by U1
M2	$\frac{1}{1}=1$	$\frac{1}{2}$	5
M3	$\frac{1}{4}$	1	2
M4	$\frac{1}{1}=1$	$\frac{1}{2}$	4

Applying the formula (10), we can predict the rating as

$$r_{1,1}=\frac{1\times\frac{1}{2}\times 5+\frac{1}{4}\times 1\times 2+1\times\frac{1}{2}\times 4}{1\times\frac{1}{2}+\frac{1}{4}\times 1+1\times\frac{1}{2}}=\frac{\frac{5}{2}+\frac{1}{2}+2}{\frac{1}{2}+\frac{1}{4}+\frac{1}{2}}=\frac{5}{\frac{5}{4}}=4$$

3.4 OPTIMALLY LEARNED INTERPOLATION WEIGHTS

So far, we have been using the normalized similarities as the interpolation weights. It was a sensible choice. A user or item that is more similar to the active user or item of interest deserves more weightage. Let us take a look at formula (5): $r_{a,ii}=\frac{\sum_{i\in N_a} s_{i,a} r_{i,ii}}{\sum_{i\in N_a} s_{i,a}}$.

Here, $w_{a,i}=\frac{s_{i,a}}{\sum_{i\in N_a} s_{i,a}}$ is the interpolation weight, defined as the normalized similarity.

Similarly in formula (8): $r_{a,ii}=\frac{\sum_{j\in N_{ii}} s_{j,ii} r_{a,ii}}{\sum_{j\in N_{ii}} s_{j,ii}}$, $w_{j,a}=\frac{s_{j,ii}}{\sum_{j\in N_{ii}} s_{j,ii}}$ is the interpolation weight.

There is nothing unique about these formulae. As long as the interpolation weights are defined such that more similar users or items get more weightage, the definition will be valid. As a counterexample, one can define an interpolation weight, which is the normalized squared similarities. For the user-based approach, this would be

$$w'_{a,i}=\frac{s_{i,a}^2}{\sum_{i\in N_a} s_{i,a}^2} \tag{11}$$

This too is valid interpolation weight, since it satisfied the condition mentioned before. In fact, such a weight has been used before in a paper, where it was claimed that such a definition improved the accuracy of recommendations in several cases.

The similarity measures defined so far are user-defined. A better approach would be to have a data-driven definition for the interpolation weights. In this section, we will learn how that can be achieved.

The relationship between the neighbor's rating and the predicted rating in the user-based approach is given in (5); here, we generalize it:

$$r_{a,ii} = \sum_{i \in N_a} r_{i,ii} \times w_{a,i} \tag{12}$$

Here, $w_{a,i}$ is the interpolation weight corresponding to the i^{th} user's rating on the item of interest. Equation (12) can be expressed in vector notation as

$$r_{a,ii} = \sum_{i \in N_a} r_{i,ii} \times w_{a,i} = r_{ii}^T w_a \tag{13}$$

Here, $r_{ii} = \begin{bmatrix} r_{1,ii} \\ r_{2,ii} \\ \ldots \\ r_{|N_a|,ii} \end{bmatrix}$ and $w_a = \begin{bmatrix} w_{a,1} \\ r_{a,2} \\ \ldots \\ r_{a,|N_a|} \end{bmatrix}$; here, $|N_a|$ is the number of neighbors of the active user. The vector r_{ii} is known, but w_a is unknown. The problem is to estimate the unknown from the data. The unknown can be estimated from the known ratings of the active user "*a*." In machine learning terminology, this is the "training" data.

Say, the active user has rated items belonging to the set *R(a)*. Therefore, $|R(a)|$ is the number of ratings available from the active user "*a*." The set *R(a)* constitutes our training data. Therefore, we can write

$$r_{a,j} = r_j^T w_a \text{ where } j \in R(a) \tag{14}$$

In matrix vector notation, we can express (14) as follows:

$$\begin{bmatrix} r_{a,1} \\ r_{a,2} \\ \ldots \\ r_{a,|R(a)|} \end{bmatrix} = \begin{bmatrix} r_1^T \\ r_2^T \\ \ldots \\ r_{|R(a)|}^T \end{bmatrix} \left[w^a \right] \tag{15}$$

In a succinct fashion (15) is expressed as

$$r_a = R_a w_a \tag{16}$$

Here, $r_a = \begin{bmatrix} r_{a,1} \\ r_{a,2} \\ \ldots \\ r_{a,|R(a)|} \end{bmatrix}$ is the vector of ratings provided by the active user; $R_a = \begin{bmatrix} r_1^T \\ r_2^T \\ \ldots \\ r_{|R(a)|}^T \end{bmatrix}$ is the matrix of ratings provided by the neighbors of the active user and w_a is the interpolation weights that needs estimation. Note that not all the users would have provided ratings to the same set of items; therefore, in (15) the missing ratings from

the neighbors should be filled with zeroes. The solution to w_a can be posed as a linear least squares problem.

$$\min_{w_a} \|r_a - R_a w_a\|_2^2 \tag{17}$$

Most often, since data is sparse, solving (17) is not a good idea; it is better to regularize the solution and solve for the following instead,

$$\min_{w_a} \|r_a - R_a w_a\|_2^2 + \lambda \|w_a\|_2^2 \tag{18}$$

To solve (18), we will first expand the term that needs to be minimized. This turns out to be

$$\begin{aligned} &\|r_a - R_a w_a\|_2^2 + \lambda \|w_a\|_2^2 \\ &= (r_a - R_a w_a)^T (r_a - R_a w_a) + \lambda w_a^T w_a \end{aligned}$$

Taking a derivative with respect to the vector w_a, we get

$$\begin{aligned} &\nabla_{w_a} (r_a - R_a w_a)^T (r_a - R_a w_a) + \lambda w_a^T w_a \\ &= 2R_a^T R_a w_a - 2R_a^T r_a + 2\lambda w_a \end{aligned}$$

For the minimization, the derivative will equal zero. With that condition, we can find a solution to w_a.

$$\begin{aligned} &2R_a^T R_a w_a - 2R_a^T r_a + 2\lambda w_a = 0 \\ &\Rightarrow (R_a^T R_a + \lambda I) w_a = R_a^T r_a \leftarrow \text{ Normal Equations} \end{aligned}$$

From the Normal Equations, one can use any solver to estimate the interpolation weights. In theory, one can get the analytic solution via the pseudoinverse: $w_a = (R_a^T R_a + \lambda I)^{-1} R_a^T r_a$. However, in practice, it is not advisable to use this formula. One will be better off using a standard solver like conjugate gradient.

Once the interpolation weights are estimated, the rating for the active user on the item of interest can be found using (13).

Let us apply what we have learned to a concrete toy example where the ratings are between 1 and 5. That will help in better understanding.

	M1	M2	M3	M4	M5
U1		5		3	2
U2	1		3		5
U3	2	2		4	
U4			4		3

Say, we want to predict U2's rating on M4. In the first step, we have to find out the neighbors of U2. We will be using cosine similarity.

Similarity between U1 and U2 is .7
Similarity between U3 and U2 is .9
Similarity between U4 and U2 is .1
The next step is to choose the neighbors. Let us take the two nearest neighbors; these neighbors will be U1 and U3.
We will now derive the interpolation weights. For this, we will use the existing ratings of U2 as the training set.
From U2's rating on M1, we have $1 = 0 \times w_{1,2} + 2 \times w_{3,2}$
From U2's rating on M3, we have $3 = 0 \times w_{1,2} + 0 \times w_{3,2}$
From U2's rating on M5, we have $5 = 2 \times w_{1,2} + 0 \times w_{3,2}$
Solving this system of equations without regularization, we get $w_{1,2} = \frac{5}{2}, w_{3,2} = \frac{1}{2}$
With the estimated interpolation weights, the predicted rating will be $3 \times \frac{5}{2} + 4 \times \frac{1}{2} = 9.5$

One notices that in this formulation, the estimated rating can be out of bounds, that is, it is beyond the scale of the maximum rating of 5. This is because we have not regularized the solution. With proper regularization, that is, with high enough value of λ, this situation can be avoided.

BIBLIOGRAPHY

Bell, R.M. and Koren, Y., 2007, August. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 7–14). ACM.

Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J., 1999, August. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 230–237). ACM.

Koren, Y., Rendle, S. and Bell, R., 2021. Advances in collaborative filtering. In *Recommender systems handbook* (pp. 91–142). Springer.

4 Latent Factor Models

4.1 INTRODUCTION

Content-based filtering (CBF) is a well-established technique in information retrieval that has been widely used for recommending items based on their inherent characteristics and a user's past preferences. To understand the essence of CBF, let's consider the example of book recommendations.

When selecting a book, readers often consider various factors, such as the author, genre, plot, and writing style. In some cases, the author's name alone may be sufficient to make a decision. For instance, a reader who enjoys Dan Brown's thrillers might seek out his other novels without the need for further consideration. This scenario represents a straightforward application of CBF, where the author's identity serves as the primary factor influencing the recommendation.

However, user preferences often extend beyond such simple attributes. A reader who appreciates science fiction novels by Arthur C. Clarke and Isaac Asimov might also find H.G. Wells's works appealing. This suggests that the genre (science fiction) plays a significant role in their preferences, adding another layer of complexity to the recommendation process.

As the complexity of items increases, so does the challenge of identifying the key factors that influence user choices. In the case of movies, for example, the production process involves numerous individuals, each contributing to the film's overall quality and appeal. A movie enthusiast might be drawn to films by a particular director, actor, or screenwriter, making it difficult to pinpoint a single factor that drives their preferences.

The difficulty in explicitly defining these factors poses a significant challenge for CBF systems. While CBF can effectively handle situations where the relevant factors are clear-cut, it struggles when dealing with complex items like movies or fashion, where the underlying influences are multifaceted and intertwined.

Let's consider a more generalized scenario. Suppose you're an avid science fiction fan who has thoroughly enjoyed works by renowned authors like Arthur C. Clarke and Isaac Asimov. When faced with a selection of science fiction novels by less established authors, would a CBF system recommend these books to you?

In this instance, the system would need to consider additional factors beyond just the genre (science fiction) to make informed recommendations. This is because the reputation of an author can significantly influence the perceived quality and appeal of a book. A well-known author with a proven track record of success may command more trust and interest from readers, even if their genre preferences align.

To address this challenge, CBF systems can incorporate the factor of publisher reputation. A book published by a reputable publishing house like HarperCollins, known for its editorial standards and commitment to quality, might gain more

DOI: 10.1201/9781003511267-4

credibility compared to a book from an unknown publisher. This association with a renowned publisher can indirectly convey a level of quality and appeal to the reader.

In this expanded scenario, the CBF system would consider three key factors: author, genre, and publisher. By evaluating these factors in combination, the system can make more informed recommendations, even when dealing with less established authors. The system can infer that a science fiction novel by a new author published by HarperCollins is likely to align with the preferences of a science fiction enthusiast, even if the author's name alone doesn't carry the same weight.

This refined approach to CBF highlights the importance of considering multiple factors when recommending items, especially when dealing with complex domains like books or movies. By incorporating reputation and other relevant attributes, CBF systems can provide more personalized and effective recommendations that cater to the diverse tastes and preferences of users.

Content-based filtering has long been a mainstay in the realm of recommender systems, offering a personalized approach to suggesting items based on their inherent characteristics and a user's past preferences. However, its effectiveness is often limited by the inherent complexity of user preferences and the difficulty in identifying the key factors that influence their decisions.

Consider the realm of movies, a domain where CBF faces significant challenges. Making a movie is a collaborative endeavor involving hundreds of individuals, each contributing to the film's overall aesthetic, narrative, and emotional impact. While CBF can effectively recommend movies based on genre, actors, or director, it struggles to capture the nuances of individual preferences that stem from the interplay of various factors like cinematography, musical score, and acting performances.

To illustrate this challenge, imagine asking a movie aficionado to explain their decision-making process when selecting a film. They might mention their fondness for a particular genre, actor, or director, but pinpointing the exact factors that drive their preferences becomes a complex task. The choice often arises from a subconscious blend of preferences, making it difficult to codify these factors into a set of rules that CBF can effectively utilize.

Similarly, fashion, another domain where CBF encounters limitations, presents a complex interplay of personal style, cultural trends, social cues, and even emotional states. Fashion choices are not merely driven by the aesthetics or functionality of an item; they reflect an individual's identity, aspirations, and societal context. Capturing these intricate factors and translating them into a CBF algorithm is a formidable undertaking.

The inherent complexity of user preferences in these domains poses a significant hurdle for CBF. A CBF system might recommend a movie based on its genre and positive reviews, only to find that the user dislikes the film due to its specific directorial style or acting choices. Similarly, a CBF system might suggest a fashion item based on its style and popularity, but the user might reject it due to its taste or the social context in which they intend to wear it.

These limitations highlight the need for alternative recommendation approaches, such as collaborative filtering (CF), which leverages the collective

wisdom of users to identify items that are similar to those they have enjoyed in the past. While CF also faces challenges, particularly with cold-start problems and the assumption of similar preferences among users, it offers a more robust approach to recommendation when dealing with complex domains like movies and fashion.

In the case of movies, CF systems can analyze the preferences of a large group of users and identify patterns in their choices, enabling them to recommend movies that have been well-received by users with similar tastes. This approach goes beyond the limitations of CBF by capturing the collective preferences of users who have experienced the nuances of various films.

Similarly, for fashion, CF systems can identify patterns in users' purchase history and style preferences, recommending items that align with their tastes and current trends. This approach allows the system to adapt to the dynamic nature of fashion and capture the evolving preferences of its users.

In summary, content-based filtering faces inherent challenges in domains like movies and fashion due to the complexity of user preferences and the difficulty in identifying the key factors that influence their decisions. While CBF has demonstrated its effectiveness in simpler domains, alternative approaches like collaborative filtering offer more robust solutions for recommendation in these complex environments.

4.2 LATENT FACTOR MODEL

The inadequacies of content-based filtering led to the development of the latent factor model (LFM). Here, the assumption is that certain factors are responsible for our choices of items; however, as we know now from the failures of content-based filtering, these factors are impossible to know—hence, they are assumed to be hidden/latent. Our choices toward these factors determine whether we will like an item or not. Similarly, each item will possess these factors to a differing degree. When there is a match between the users and items latent factors (just like content-based filtering), the user likes the item and correspondingly the rating comes out to be high.

Formally, we introduce the user i's latent factors as u_i and item v's latent factors as v_j. Therefore, the rating of user i on item j is expressed as

$$x_{i,j} = u_i^T v_j \tag{1}$$

This expression holds for every user–item pair. Assuming there are m users and n items, we can rewrite (1) as,

$$X_{m \times n} = \begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_m^T \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \tag{2}$$

We can have a user's latent factor matrix defined as $\begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_m^T \end{bmatrix} = U$ and an item's latent factor matrix defined as $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = V$. With this (2) can be written more succinctly as

$$X = UV \tag{3}$$

Here, X denotes the complete ratings matrix. This is now known in recommender systems. What we have is a partially observed version of X. We denote it as

$$Y = R \cdot X \tag{4}$$

Here, Y is the observed matrix and R is a binary mask; it has zeroes where the ratings are not observed and 1 where the rating is known.

Incorporating the observation equation into the latent factor model leads to

$$Y = R \cdot X = R \cdot (UV) \tag{5}$$

Given Y and R, the problem is to estimate U and V. If the user and item latent factors are known, one can estimate the complete rating matrix using (3). In the rest of this section, we learn about the solution of (5). However, before we dive into its solution, we need to learn about the solution to matrix factorization.

4.2.1 Matrix Factorization

Matrix factorization is a powerful technique in the realm of machine learning that has found widespread application across various domains, ranging from computer vision and information retrieval to bioinformatics and sensor localization. At its core, matrix factorization aims to decompose a given matrix into a product of two or more lower-rank matrices. This process unveils hidden patterns and relationships within the data, providing valuable insights for various tasks.

Consider a matrix representing user–item interactions in a recommender system. Each row corresponds to a user, each column to an item, and the matrix entries represent the interactions or ratings between users and items. Applying matrix factorization to this user–item interaction matrix reveals latent features, or underlying characteristics, that govern user preferences and item attributes. By analyzing these latent features, the recommender system can effectively predict user preferences and recommend items that are likely to be of interest to a particular user.

In computer vision, matrix factorization plays a crucial role in image and video processing tasks, such as image compression, face recognition, and motion tracking. By decomposing images or video frames into lower-rank

matrices, matrix factorization can extract salient features and patterns, enabling efficient image encoding, accurate face identification, and robust motion tracking algorithms.

Information retrieval systems also leverage matrix factorization to enhance document search and relevance ranking. By decomposing a document-term matrix, which represents the relationship between documents and keywords, matrix factorization can identify latent topics or themes within the collection of documents. This enables the retrieval system to understand the semantics of documents and provide more relevant search results for user queries.

Bioinformatics researchers employ matrix factorization to analyze gene expression data and uncover hidden patterns in gene interactions. By decomposing gene expression matrices, matrix factorization can identify groups of genes that exhibit correlated expression patterns, suggesting potential functional relationships or regulatory mechanisms.

Sensor localization systems utilize matrix factorization to estimate the positions of multiple sensors based on their distance measurements. By decomposing the distance measurement matrix, matrix factorization can infer the relative positions of the sensors, enabling accurate localization in indoor or outdoor environments.

In summary, matrix factorization serves as a versatile tool in machine learning, enabling the analysis and interpretation of complex data across diverse domains. Its ability to uncover latent patterns and relationships within the data makes it an invaluable technique for tasks ranging from recommender systems and computer vision to information retrieval and bioinformatics. As machine learning continues to evolve, matrix factorization is poised to play an increasingly prominent role in extracting knowledge and insights from vast amounts of data.

Let X be the low-rank matrix (of rank f) that is to be factorized into U and V. The problem is expressed as

$$X_{m\times n} = U_{m\times f} V_{f\times n} \tag{6}$$

In matrix factorization, the problem is to estimate U and V given X and f.

The most straightforward way to express the estimation problem is to pose it as

$$\min_{U,V} \left\| X - UV \right\|_F^2 \tag{7}$$

Here $\left\| \bullet \right\|_F$ is the Frobenius norm of the matrix, which is defined as the Euclidean norm of all the elements in it.

Note that in (7), both U and V are variables; therefore, it is not a regular linear least squares problem as we encountered before. Rather, it is a bilinear problem. By that, what we mean is the problem is linear in V if U is assumed to be constant and it is linear in U if V is assumed to be constant.

The minimization problem (7) is non-convex. Therefore, there is no guarantee of reaching a global minimum. Furthermore, one needs to initialize the problem judiciously in order to reach a meaningful solution. The easiest way to solve (7) is via

alternating least squares (ALS), that is, update U assuming V to be fixed and update V assuming U to be fixed. Iterate over these two updates till the values of U and V do not change significantly with iterations.

The algorithm for matrix factorization via ALS is as follows:

Initialize: U_0
In iteration "k"

$$V_k \leftarrow \min_V \left\| X - U_{k-1} V \right\|_F^2$$

$$U_k \leftarrow \min_U \left\| X - U V_k \right\|_F^2$$

End

Here, we have assumed that U_0 is initialized. Alternately, one can initialize V_0 and start with the update of U in the kth iteration instead.

The updates for U_k and V_k are straightforward linear least squares problems. Although the variables are matrices, their solutions will be similar to the ones we did for the learned interpolation weights in the previous chapter. Let us take the example of updating V_k:

$$\min_V \left\| X - U_{k-1} V \right\|_F^2 \tag{8}$$

Note that X and V are matrices formed by stacking "n" column vectors, that is, $X = \left[x^1 \mid x^2 \mid \ldots \mid x^n \right]$ and $V = \left[v^1 \mid v^2 \mid \ldots \mid v^n \right]$. Solving (8) is the same as solving the following:

$$\min_{v^i} \left\| x^i - U_{k-1} v^i \right\|_2^2, \text{ i=1...n} \tag{9}$$

The solution to (9) can be obtained from the normal equations.

$$U_{k-1}^T U_{k-1} v^i = U_{k-1}^T x^i, \text{ i=1...n} \tag{10}$$

We can stack the individual vectors as columns of matrices V and X and get

$$\begin{aligned} & U_{k-1}^T U_{k-1} \left[v^1 \mid v^2 \mid \ldots \mid v^n \right] = U_{k-1}^T \left[x^1 \mid x^2 \mid \ldots \mid x^n \right] \\ & or,\ U_{k-1}^T U_{k-1} V = U_{k-1}^T X \end{aligned} \tag{11}$$

This gives us the update for V_k. It is possible to write the closed form solution as well:

$$V_k = \left(U_{k-1}^T U_{k-1} \right)^{-1} U_{k-1}^T X \tag{12}$$

However, it is not advisable to use this formula in practice owing to numerical problems that may arise because of its ill-conditioned nature.

The solution for U_k is almost the same. It is a linear least squares problem.

$$\min_U \left\| X - UV_k \right\|_F^2 \tag{13}$$

We can convert (13) to our more familiar form by taking the transpose of the entire expression.

$$\min_U \left\| X - UV_k \right\|_F^2 \equiv \min_U \left\| X^T - V_k^T U^T \right\|_F^2 \tag{14}$$

As in the case of V, the normal equations will be

$$V_k V_k^T U^T = V_k X^T \tag{15}$$

Solving (15) will get us U^T. From it, we can easily obtain U_k by transposing.

This brings us to the conclusion of the ALS algorithm. We will continue the updates for U and V till their values converge, that is, when $\left\| V_k - V_{k-1} \right\|_F^2$ and $\left\| U_k - U_{k-1} \right\|_F^2$ are small.

In most cases, we would want to regularize matrix factorization. Therefore, instead of solving (7), one would solve the following instead.

$$\min_{U,V} \left\| X - UV \right\|_F^2 + \lambda \left(\left\| U \right\|_F^2 + \left\| V \right\|_F^2 \right) \tag{16}$$

The derivation remains the same as before; therefore, we are directly writing the normal equations for updating V and U.

$$\left(U_{k-1}^T U_{k-1} + \lambda I \right) V_k = U_{k-1}^T X \tag{17a}$$

$$\left(V_k V_k^T + \lambda I \right) U^T = V_k X^T \tag{17b}$$

Depending on the problem, there may be other regularizers as well. We have shown the solution for the Frobenius norm regularization since it is the most common.

4.2.2 Matrix Completion via Factorization

Matrix factorization (MF) is a powerful technique for dimensionality reduction and latent feature extraction, but its direct application to collaborative filtering (CF) is hindered by the sparsity of the rating matrix. In CF, the matrix representing user–item interactions is typically highly sparse, with a large majority of entries representing missing ratings. This sparsity poses a challenge for traditional MF algorithms, which assume that the matrix is fully observed.

To address this challenge, a two-step approach is commonly employed:

1. **Matrix Imputation:** The first step involves imputing the missing ratings to obtain a complete matrix. Various techniques can be used for imputation,

such as mean imputation, k-nearest neighbors (kNN) imputation, and singular value decomposition (SVD) imputation. Each method aims to estimate the missing ratings based on the observed ratings and the relationships between users and items.

2. **Low-Rank Matrix Approximation:** Once the matrix is complete, MF can be applied to decompose the imputed matrix into a product of two lower-rank matrices. This decomposition aims to capture the latent features or underlying patterns that govern user preferences and item attributes. The resulting low-rank matrices represent a more compact representation of the original matrix while preserving the essential information for recommendation purposes.

The two steps, matrix imputation and low-rank matrix approximation, are often performed iteratively. The imputed matrix obtained from the first step is used as input for the MF algorithm in the second step. The resulting low-rank matrix is then used to refine the imputed matrix, and the process is repeated until convergence is reached.

This iterative approach allows the CF system to gradually improve its understanding of the underlying relationships between users and items, despite the sparsity of the rating matrix. By combining imputation techniques with MF, CF systems can effectively extract latent features and make personalized recommendations even in the presence of a large number of missing ratings.

Our final goal is to solve the problem

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 \tag{18}$$

The symbols have already been explained. Here, we will concentrate on deriving the algorithm. The derivation will be based on the majorization-minimization (MM) approach. The general outline of this approach is as follows:

Let $J(x)$ be the function to be minimized. Start with an initial point (at $k=0$) x_k (Figure 4.1 a). A smooth function $G_k(x)$ is constructed through x_k, which has a higher value than $J(x)$ for all values of x apart from x_k, at which the values are the same. This is the majorization step. The function $G_k(x)$ is constructed such that it is smooth and easy to minimize. At each step, minimize $G_k(x)$ to obtain the next iterate x_{k+1} (Figure 4.1 b). A new $G_{k+1}(x)$ is constructed through $x_{k+1,}$ which is now minimized to obtain the next iterate x_{k+2} (Figure 4.3 c). As can be seen, the solution at every iteration gets closer to the actual solution.

First, let us consider the minimization of the least squares problem:

$$J(x) = \left\| y - Ax \right\|_2^2 \tag{19}$$

For this minimization, $G_k(x)$ is chosen to be

$$G_k(x) = \left\| y - Ax \right\|_2^2 + (x - x(k))^T (aI - A^T A)(x - x(k)) \tag{20}$$

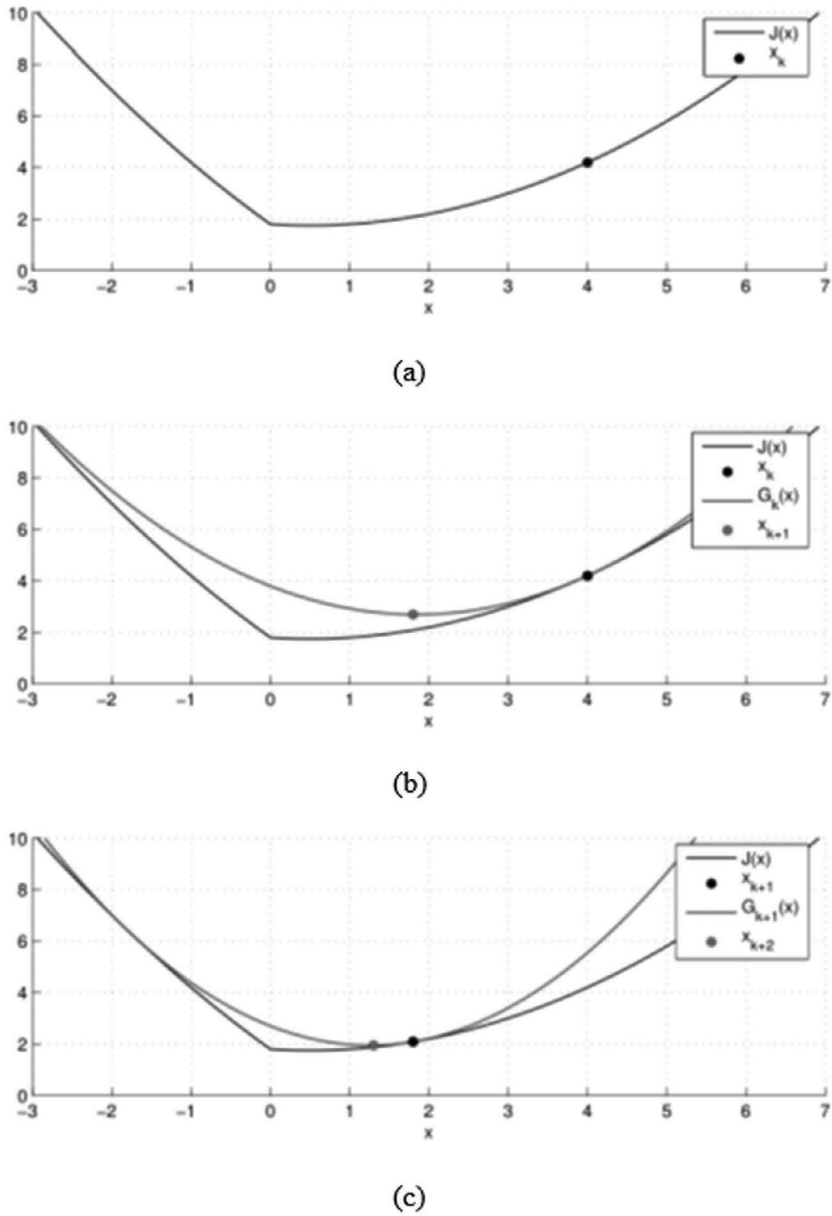

FIGURE 4.1 Majorization-Minimization: (a) shows the cost function to be minimized; the curve as one can see is non-differentiable at the point of interest; (b) shows the construction of the majorizer at the current solution; and (c) shows the update of the solution after minimizing the majorizer.

where a is the maximum eigenvalue of the matrix A^TA and I is the identity.

$$\begin{aligned} G_k(x) &= \|y - Ax\|_2^2 + (x - x(k))^T (aI - A^T A)(x - x(k)) \\ &= y^T y - 2y^T Ax + x^T A^T Ax + (x - x(k))^T (aI - A^T A)(x - x(k)) \\ &= y^T y + x(k)^T (aI - A^T A)x(k) - 2(y^T A + x(k)^T (aI - A^T A))x(k) + ax^T x \\ &= a(-2b^T x - x^T x) + c \end{aligned}$$

where $b = x(k) + \frac{1}{a} A^T (y - Ax(k))$, $c = y^T y + x(k)^T (aI - A^T A)x(k)$

Using the identity $\|b - x\|_2^2 = b^T b - 2b^T x + x^T x$, one can write

$$\begin{aligned} G_k(x) &= a\|b - x\|_2^2 - ab^T b + c \\ &= a\|b - x\|_2^2 + K \end{aligned}$$

where K consists of terms independent of x.

Therefore, minimizing (20) is the same as minimizing the following:

$$G_k^{'}(x) = \|b - x\|_2^2 \tag{21}$$

where $b = x(k) + \frac{1}{a} A^T (y - Ax(k))$.

This update is known as the Landweber iteration.

For our problem, $R = A$, $Y = y$, and $UV = x$. Substituting these, the iterative solution to (18) turns out to be

$$B = (UV)_k + \frac{1}{a} A^T (Y - R \cdot (UV)_k) \tag{22}$$

Since R is a restriction operator, it is possible to simplify (22).

$$B = (UV)_k + Y - R \cdot (UV)_k \tag{23}$$

In words, what it means is that one has to mask the entries in UV from the previous iteration and subtract it from the given observation Y before adding the resultant to UV from the previous iteration.

With the Landweber iterations, we replace the original problem that we need to solve (18) with the $\min_{U,V} \|B - UV\|_F^2$. This is simply matrix factorization, and we have already learned how to solve it.

If we put everything together, the algorithm for solving the latent factor model will be the following:

Initialize: X_0
In iteration "k"
Compute via Landweber iterations $B = X_{k-1} + Y - R \cdot X_{k-1}$
Solve for U and V via matrix factorization, that is, $\min_{U,V} \|B - UV\|_F^2$
  Initialize: U_0
  In iteration "p"
  Update V from the normal equations: $U_{p-1}^T U_{p-1} V = U_{p-1}^T X$
  Update U from the normal equations: $V_p V_p^T U^T = V_p X^T$
  Continue until convergence
Update: $X_k = UV$
Continue until convergence

The algorithm consists of two loops. The outer loop is for the Landweber iterations, and the inner loop is for matrix factorization.

Here, we have discussed the solution for the unregularized optimization. In practice, one might be interested in solving the regularized version like the following:

$$\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2 \right) \tag{24}$$

In this case, there will be no changes to the Landweber update. The only change will be in the matrix factorization. Instead of the unregularized matrix factorization we did, we will need to use regularized matrix factorization. We have already learned the updates for that.

4.2.3 Multiplicative Updates

While alternating least squares (ALS) is a commonly used approach for matrix factorization due to its simplicity and ease of implementation, it is not the mathematically optimal solution. A more efficient and effective alternative is to employ multiplicative updates, which directly optimize the objective function of matrix factorization without the need for alternating updates between user and item features.

Multiplicative updates offer several advantages over ALS:

1. **Faster Convergence:** Multiplicative updates typically converge to the optimal solution faster than ALS. This is because multiplicative updates directly minimize the objective function, while ALS involves an iterative process of updating user and item features.
2. **Scalability:** Multiplicative updates are more scalable to large datasets compared to ALS. This is because multiplicative updates can be efficiently parallelized, allowing for distributed computation on multiple machines.

3. **Avoidance of Local Minima:** Multiplicative updates are less prone to getting stuck in local minima compared to ALS. This is because multiplicative updates directly optimize the objective function, while ALS is susceptible to converging to suboptimal solutions.

In practice, multiplicative updates are often implemented using SGD (stochastic gradient descent) or Adam, which are optimization algorithms that iteratively update the model parameters in the direction that minimizes the objective function. These algorithms have proven to be highly effective in solving matrix factorization problems, particularly for large-scale datasets. While ALS is a widely used approach for matrix factorization, multiplicative updates offer a more efficient, scalable, and robust alternative. By directly optimizing the objective function and avoiding local minima, multiplicative updates can achieve better performance and converge to the optimal solution faster, making them a preferred choice for large-scale matrix factorization problems.

We are not going to give the derivation here but will state the algorithm for solving $\min_{U,V} \|X - UV\|_F^2$.

Initialize: V
In every iteration

$$U_{[i,j]} \leftarrow U_{[i,j]} \frac{\left(XV^T\right)_{[i,j]}}{\left(UVV^T\right)_{[i,j]}}$$

$$V_{[i,j]} \leftarrow V_{[i,j]} \frac{\left(U^T X\right)_{[i,j]}}{\left(U^T UV\right)_{[i,j]}}$$

End

In the algorithm, $U_{[i,j]}$ and $V_{[i,j]}$ to the entry in the i^{th} row and j^{th} column of the corresponding matrices.

The multiplicative updates are a better choice for the latent factor model. For that, after the Landweber iteration, instead of the ALS algorithm for updating U and V, this one can be used.

4.3 NUCLEAR NORM MINIMIZATION

The latent factor model's objective function, the function that the model aims to minimize, is non-convex. This means that the objective function may have multiple local minima, which are points where the gradient of the function is zero but the function itself is not at its lowest possible value.

The non-convexity of the objective function poses a challenge for optimization algorithms, as they can get stuck in local minima rather than converging to the global minimum, which represents the optimal solution. This sensitivity to initialization means that the final output of the latent factor model can significantly depend on the initial values assigned to the model parameters.

Let's take the following toy example for recommender systems.

	M1	**M2**	**M3**	**M4**
U1	5		3	
U2		2	3	
U3	3			4
U4		5		1

It is a matrix completion problem. We have users along the rows and items along the columns. The ratings are the entries of the matrix. The complete rating matrix is not known. The problem is to complete this matrix.

The latent factor model does that. Note that the matrix is of low rank. This is evident from the LFM since the number of factors is much smaller than the dimensionalities of the matrix. Completing a low rank matrix has a rich theory and algorithms behind it. Ideally, one would like to solve

$$\min_X rank(X) \text{ s.t. } \left\|Y - R\cdot(X)\right\|_F^2 \leq \varepsilon \tag{25}$$

However, rank minimization is a combinatorial optimization problem with doubly exponential complexity. This is why it is not feasible to solve (25) directly. Instead, mathematicians have shown that one can get a guaranteed minimum rank solution by minimizing the nuclear norm.

$$\min_X \left\|X\right\|_* \text{ s.t. } \left\|Y - R\cdot(X)\right\|_F^2 \leq \varepsilon \tag{26}$$

Here, $\left\|\bullet\right\|_*$ is the nuclear norm defined as the sum of its singular values. The nuclear norm is the closest convex surrogate to the rank of a matrix. The problem is convex and can be solved using semi-definite programming. The formulation (26) is one of constrained optimization. In practice, one solves the unconstrained version instead.

$$\min_X \left\|Y - R\cdot(X)\right\|_F^2 + \lambda\left\|X\right\|_* \tag{27}$$

Here, we will learn the solution to (27). As we did for the latent factor model, the first step would be to apply Landweber iterations to the data fidelity term of (27). With this, the minimization problem after every Landweber iteration translates to

$$\min_X \left\|B - X\right\|_2^2 + \lambda\left\|X\right\|_* \tag{27}$$

The following property of singular value decomposition holds in general

$$\|A_1 - A_2\|_F^2 \geq \|s_1 - s_2\|_F^2$$

where A_1 and A_2 denote two matrices and s_1 and s_2 are their singular value vectors, respectively.

Using this property, minimizing (27) is the same as minimizing the following:

$$\min_{s_x} \|s_b - s_x\|_2^2 + \lambda \|s_x\|_1 \tag{28}$$

where s_b and s_x are the singular values of matrices corresponding to B and X, respectively. Note that we have used the inequality only for the data fidelity term. The l_1-norm of the singular values follows from the definition of nuclear norm.

It is possible to write (28) in a decoupled fashion

$$\min_{s_x^{(i)}} \sum_i \left(s_b^{(i)} - s_x^{(i)}\right)^2 + \frac{\lambda}{a}\left|s_x^{(i)}\right|, \forall \mathrm{i} \tag{29}$$

It is possible to minimize (29) by minimizing each of the terms

$$2(s_b^{(i)} - s_x^{(i)}) + \lambda signum(s_x^{(i)})$$

And setting the derivative to zero and rearranging, we get

$$s_x^{(i)} = s_b^{(i)} + \frac{\lambda}{2} signum(s_x^{(i)}) \tag{30}$$

This is minimized by soft thresholding

$$s_x^{(i)} = signum\left(s_b^{(i)}\right) \max\left(0, |\, s_b^{(i)}\, | - \frac{\lambda}{2}\right)$$

We have computing for each singular value separately. Written for the entire vector of singular values,

$$s_x = signum(s_b) \max\left(0, |\, s_b\, | - \frac{\lambda}{2}\right) \tag{31}$$

From the thresholded singular values, the matrix is updated as $X_{k+1} = U\Sigma_x V^T$. Here, Σ_x is formed by having s_x along its diagonals and zeroes elsewhere.

The derivation leads to the following singular value thresholding algorithm:

Initialize: X_0
$B = X_{k-1} + Y - R \cdot X_{k-1}$
Compute its SVD: $B = U\Sigma_b V^T$
Soft threshold the singular values: $s_x = signum(s_b) \max\left(0, |\, s_b\, | - \frac{\lambda}{2}\right)$
Update: $X_{k+1} = U\Sigma_x V^T$

Note that, since nuclear norm minimization is convex, one can start with any random initialization.

Collaborative filtering (CF) is a powerful technique for recommender systems, enabling them to make personalized recommendations based on the preferences of a large group of users. However, CF faces a fundamental challenge: matrix completion. The rating matrix, representing user–item interactions, is often partially observed due to missing ratings. This sparsity poses a significant hurdle for CF algorithms, as they need to reconstruct the complete matrix to effectively capture user preferences and item attributes.

One approach to address matrix completion is matrix factorization (MF). MF aims to decompose the partially observed rating matrix into a product of two lower-rank matrices, representing user features and item features. This decomposition captures the latent patterns or underlying relationships between users and items, allowing for the estimation of missing ratings and improved recommendation accuracy.

An alternative approach to matrix completion is nuclear norm minimization (NNM). NNM directly optimizes a convex objective function that penalizes the rank of the matrix, effectively promoting a low-rank approximation. This approach is theoretically optimal, ensuring convergence to the globally optimal solution, unlike MF, which can get stuck in local minima.

Both MF and NNM offer distinct advantages and disadvantages:

Matrix Factorization:

- Advantages:
  - Efficient updates using least squares
  - Captures latent features and patterns
- Disadvantages:
  - Non-convex, susceptible to local minima
  - Sensitive to initialization
  - Requires specifying the number of latent factors

Nuclear Norm Minimization:

- Advantages:
  - Theoretically optimal, guaranteed convergence
  - Convex objective function, robust to initialization
  - Any rank can be chosen
- Disadvantages:
  - Computationally expensive SVD in each iteration
  - Less efficient than MF

In terms of accuracy, both MF and NNM have demonstrated similar performance when reasonable initialization is used for MF. The choice between the two methods often depends on specific requirements and computational resources. MF offers efficiency and the ability to capture latent features, while NNM provides theoretical guarantees and robustness to initialization.

Matrix completion is a fundamental challenge in collaborative filtering. Both matrix factorization and nuclear norm minimization offer effective solutions, each with its strengths and weaknesses. The choice between the two methods depends on specific application requirements, computational resources, and the desired balance between efficiency, theoretical guarantees, and robustness to initialization.

BIBLIOGRAPHY

Koren, Y., Bell, R. and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), pp. 30–37.

Lee, D. and Seung, H.S., 2000. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems* (p. 13). https://proceedings.neurips.cc/paper_files/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf

Majumdar, A. and Ward, R.K., 2011. Some empirical advances in matrix completion. *Signal Processing*, 91(5), pp. 1334–1338.

5 Using Metadata

5.1 INTRODUCTION

Collaborative filtering has emerged as a cornerstone of modern recommender systems, effectively predicting user preferences and delivering personalized recommendations. Traditionally, collaborative filtering has been treated as an interpolation problem, where the goal is to predict missing ratings within a user–item interaction matrix. However, in real-world applications, recommender systems possess access to a wealth of additional information, extending beyond mere ratings. This metadata, associated with both users and items, holds immense potential for enhancing the accuracy and effectiveness of recommendations.

User metadata provides valuable insights into individual preferences and characteristics, encompassing demographic details such as age, gender, location, and language preferences. Additionally, user behavior data, including browsing history, purchase history, and search patterns, further enrich the understanding of user interests and tendencies. By incorporating user metadata into collaborative filtering algorithms, recommender systems can tailor recommendations to align with specific user demographics and preferences, leading to more personalized and relevant suggestions.

Item metadata, associated with the items being recommended, provides a comprehensive description of their attributes and characteristics. This includes information such as genre, category, author, release date, and product specifications. By leveraging item metadata, recommender systems can identify semantic similarities between items and recommend items that share similar attributes to those the user has previously interacted with. This enables the system to go beyond mere ratings and recommend items based on their inherent characteristics, aligning with the user's underlying preferences.

Integrating user metadata and item metadata into collaborative filtering algorithms enhances the recommendation process by providing a richer understanding of both users and items. User metadata helps refine user profiles, while item metadata provides a deeper understanding of the items being recommended. This synergistic integration enables the system to capture complex relationships between users and items, leading to more accurate and personalized recommendations.

The sparsity of rating data, where many user–item interactions are missing, poses a significant challenge for collaborative filtering algorithms. Metadata, however, can be employed to alleviate data sparsity by providing additional information about users and items. This supplementary information can be used to infer user preferences and item similarities, even in the absence of explicit ratings. By incorporating metadata, recommender systems can effectively address the sparsity issue and provide more robust recommendations.

In the realm of collaborative filtering, metadata plays a crucial role in enhancing the accuracy and effectiveness of recommendation systems. One prominent

DOI: 10.1201/9781003511267-5

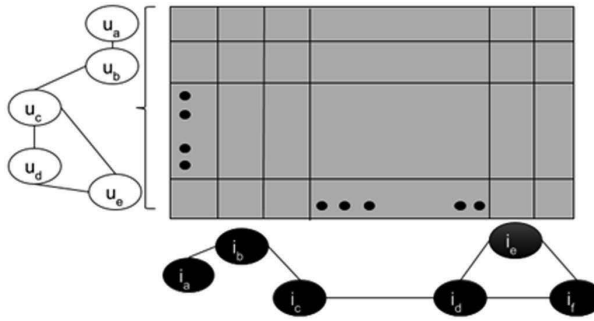

FIGURE 5.1 Schematic for Matrix Completion on Graphs: Along the columns are the items and the along the rows are the users. Both the users and the items are shown as nodes of a graph. The connection between the nodes are the similarities between them.

approach to incorporating metadata is through the construction of graphs, as illustrated in Figure 5.1. Unlike directly utilizing metadata features, this method involves constructing separate graphs for both users and items, with edges representing similarities between entities. These graphs are then employed to regularize the matrix completion algorithm, a technique used to predict missing ratings.

This graph-based approach fosters connections between similar users and similar items, leading to a regularization effect that encourages these entities to exhibit similar ratings. This strategy effectively bridges the gap between neighborhood-based models, which rely on explicit similarities, and latent factor models, which infer latent factors from rating data. By incorporating metadata-derived similarities into the matrix completion algorithm, the system can achieve a more comprehensive understanding of user preferences and item attributes, resulting in more personalized and accurate recommendations.

The graph-based approach offers several advantages over traditional collaborative filtering methods:

1. **Addressing Data Sparsity:** By incorporating metadata, the system can infer user preferences and item similarities even in the absence of explicit ratings, alleviating the issue of data sparsity.
2. **Enhancing Recommendation Accuracy:** The regularization effect introduced by the graphs reinforces the connections between similar users and items, leading to more accurate predictions of missing ratings and, consequently, improved recommendation performance.
3. **Capturing Complex Relationships:** The graph structure allows for the representation of complex relationships between users and items, going beyond mere ratings to capture underlying preferences and characteristics.

In the graph-based approach, each user becomes a node for the user's graph (W_U) and each item becomes a node for the item's graph (W_V). The edges between the nodes are computed as follows:

$$W_U(i,j) = \exp\left(-\frac{\left\|x_i^{\rightarrow} - x_i^{\rightarrow}\right\|_2^2}{\sigma^2}\right),\ x_i^{\rightarrow} \text{ denotes } i^{th} \text{ row of } X$$

$$W_V(i,j) = \exp\left(-\frac{\left\|x_i^{\downarrow} - x_i^{\downarrow}\right\|_2^2}{\sigma^2}\right),,\ x_i^{\downarrow} \text{ denotes } i^{th} \text{ column of } X$$

Here the parameter σ is appropriately chosen.

The graph Laplacians are used for regularizing the matrix completion problem. The Laplacians are defined as

$$L_U = I_U - W_U$$
$$L_V = I_V - W_V$$

where I_U and I_V are identity matrices having the same sizes as the user and item graphs, respectively.

5.2 MATRIX FACTORIZATION ON GRAPHS

As discussed in the previous chapter, the rating matrix can be modeled as a product of a user latent factor matrix and an item latent factor matrix, that is, $X = UV$. This model was incorporated into the observation model, which was expressed as $Y = R \cdot X = R \cdot (UV)$. Therefore, estimating the rating matrix boiled down to the estimation of the individual latent factor matrices. Usually, this was framed as an optimization problem: $\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right)$. The regularization terms did not do much apart from keeping the values in the latent factor matrices small and smooth.

Here, we will show how the matrix factorization problem can be regularized by graph Laplacian. The optimization problem would be

$$\arg\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + \gamma\left(Tr\left(U^T L_U U\right) + Tr\left(V L_V V^T\right)\right) \quad (1)$$

In contrast to traditional Tikhonov regularization, which enforces smoothness across all latent factors, graph regularization imposes smoothness specifically among latent factors of similar users and items. This approach aligns more logically with the underlying structure of recommender systems, where user preferences and item attributes exhibit inherent similarities.

The graph regularization framework achieves this selective smoothness by leveraging graph Laplacians, which encode the similarity relationships between users and items. These Laplacians effectively capture the local structure of the user–item interaction network, ensuring that latent factors of similar entities remain consistent.

This targeted smoothness approach offers several advantages over global smoothness regularization:

1. **Enhanced Personalization:** By focusing on similarities between users and items, graph regularization promotes more personalized recommendations that align with individual preferences and item characteristics.
2. **Improved Accuracy:** The emphasis on local smoothness ensures that latent factors of similar entities maintain consistency, leading to more accurate predictions of missing ratings and enhanced recommendation performance.
3. **Explanatory Power:** The graph structure provides a transparent representation of user–item relationships, enabling an understanding of the factors influencing recommendation decisions.

In essence, graph regularization serves as a more refined and intuitive approach to regularization in collaborative filtering. By tailoring smoothness to similar entities, the system effectively captures the underlying structure of user preferences and item attributes, leading to more personalized, accurate, and interpretable recommendations.

The solution to this problem requires majorization-minimization. This has been discussed in the last chapter. It requires the introduction of a temporary variable defined as $B = \left(UV\right)_k + Y - R \cdot \left(UV\right)_k$; here, "k" denotes the iteration number. Majorization-minimization of (1) leads to the iterative solution of the following expression:

$$\arg\min_{U,V} \left\|B - UV\right\|_F^2 + \gamma\left(Tr\left(U^T L_U U\right) + Tr\left(VL_V V^T\right)\right) \tag{2}$$

This is solved by iteratively updating the two variables via

$$U \leftarrow \min_U \left\|B - UV_k\right\|_F^2 + \lambda Tr\left(U^T L_U U\right) \tag{3}$$

$$V \leftarrow \min_V \left\|B - U_k V\right\|_F^2 + \lambda Tr\left(VL_V V^T\right) \tag{4}$$

For solving U, we take the derivative of the cost function and equate it to 0. This leads to

$$\lambda L_U U + UV_k V_k^T = BV_k^T \tag{5}$$

Note that (17) is a Sylvester's equation of the form $A_1 X + XA_2 = C$, where $A_1 = \lambda L_U$, $A_2 = V_k V_k^T$ and $C = BV_k^T$. There are many efficient solvers for the same.

Similarly, for solving V, we get after differentiating and equating to 0

$$U_{k+1} U_{k+1}^T V + \lambda L_V V = U_{k+1}^T B \tag{6}$$

As one can notice, this too is a Sylvester's equation.

This concludes the derivation of the algorithm. The complete pseudo-code looks like the following:

Initialize: U_0 and V_0

Outer loop k

$B = (UV)_k + Y - R \cdot (UV)_k$

Inner loop

$U \leftarrow Sylvester\left(\lambda L_U, V_k V_k^T, BV_k^T\right)$

$V \leftarrow Sylvester\left(U_{k+1} U_{k+1}^T, \lambda L_V, U_{k+1}^T B\right)$

End Inner Loop

End Outer Loop

Once the algorithm ends, the rating matrix is obtained as a product of the latent factor matrices.

In collaborative filtering, the Laplacian matrix, a component of graph regularization, captures the similarity relationships between users and items. Traditionally, metadata is utilized to construct the Laplacian, providing explicit information about user preferences and item attributes. However, when metadata is limited or unavailable, an alternative approach involves learning the Laplacian directly from the rating data.

Graph-based semi-supervised learning, a data-driven technique, extracts implicit similarity information embedded in the rating matrix to construct the Laplacian. This approach involves data preprocessing, neighbor identification, Laplacian matrix construction, and model integration.

Data-driven Laplacian construction offers adaptability, scalability, and flexibility, enabling Laplacian construction even when metadata is limited. The choice of the Laplacian construction technique depends on data availability and the desired level of granularity in similarity modeling.

5.3 NUCLEAR NORM MINIMIZATION ON MULTIPLE GRAPHS

In the context of nuclear norm minimization, we can use the Laplacians to regularize the complete matrix, once along the rows for the users' graph Laplacian and once along the columns for the items' graph Laplacian. The optimization problem would look like this:

$$\min_X \|Y - R \cdot X\|_F^2 + \lambda \|X\|_* + \gamma_1 Tr\left(X^T L_U X\right) + \gamma_2 Tr\left(X L_V X^T\right) \tag{7}$$

In this section, we will extend the graph Laplacian framework to incorporate diverse metadata. While the previous discussion focused on incorporating a single type of metadata, real-world scenarios often involve multiple sources of metadata. To effectively capture the underlying relationships between users and items, the graph Laplacian framework can be extended in several ways, including constructing multiple graph Laplacians, weighted graph Laplacians, kernel-based graph Laplacians,

multi-graph Laplacians, and metadata fusion. The choice of the most appropriate method depends on the specific characteristics of the data, the desired level of granularity, and the computational constraints. By carefully considering these factors, researchers and practitioners can develop effective collaborative filtering algorithms that leverage the rich information provided by multiple sources of metadata.

Simply put, we will regularize the matrix by each available graph Laplacian and add them up. This leads to

$$\min_X \left\|Y - R\cdot X\right\|_F^2 + \lambda\left\|X\right\|_* + \gamma_1 \sum_i Tr\left(X^T L_{U_i} X\right) + \gamma_2 \sum_j Tr\left(X L_{V_j} X^T\right) \tag{8}$$

Here, the L_{U_i}'s and the L_{V_j}'s are the multiple graph Laplacians. They can either correspond to different kernel measures of similarity, for example, nearest neighbor, linear, RBF, polynomial, etc.; or they can be defined from different types of metadata (already mentioned before for collaborative filtering and drug-target interactions), or the Laplacians can be a combination of both.

To solve (8), we introduce two proxy variables $Z=X^T$ and $W=Z$ into (8). The augmented Lagrangian is now expressed as

$$\begin{aligned}&\min_{X,Y,Z} \left\|Y - R\cdot X\right\|_F^2 + \lambda\left\|X\right\|_* + \gamma_1 \sum_i Tr\left(Z L_{U_i} Z^T\right) + \gamma_2 \sum_j Tr\left(W L_{V_j} W^T\right) \\ &+\mu_1 \left\|Z^T - X\right\|_F^2 + \mu_2 \left\|W - X\right\|_F^2\end{aligned} \tag{9}$$

Using the alternating direction method of multipliers (ADMM), (9) can be segregated into the following sub-problems:

$$\begin{aligned}&\text{P1: } \min_X \left\|Y - R\cdot X\right\|_F^2 + \lambda\left\|X\right\|_* + \mu_1 \left\|Z^T - X\right\|_F^2 + \mu_2 \left\|W - X\right\|_F^2 \\ &\text{P2: } \min_Y \gamma_2 \sum_j Tr\left(W L_{V_j} W^T\right) + \mu_2 \left\|W - X\right\|_F^2 \\ &\text{P3: } \min_Z \gamma_1 \sum_i Tr\left(Z L_{U_i} Z^T\right) + \mu_1 \left\|Z^T - X\right\|_F^2\end{aligned}$$

P1 is a nuclear norm regularized least square problem in X. It can be made to look familiar by rearranging the terms slightly.

$$\min_X \left\| \begin{pmatrix} Y \\ \sqrt{\mu_1} Z^T \\ \sqrt{\mu_2} W \end{pmatrix} - \begin{pmatrix} R\cdot \\ \sqrt{\mu_1} I \\ \sqrt{\mu_2} I \end{pmatrix} X \right\|_F^2 + \lambda\left\|X\right\|_* \tag{10}$$

We have already learned how to solve such problems in a previous chapter.

Taking the derivative of the cost function in P2 and equating it to zero leads to

$$\gamma_2 W \sum_j L_{V_j} + \mu_2 \left(W - X\right) = 0 \tag{11}$$

By rearranging (11), one can see that it turns out to be Sylvester's equation.

$$\gamma_2 W \sum_j L_{V_j} + \mu_2 IW = \mu_2 IX \tag{12}$$

Similarly, the solution to P3 turns out to be Sylvester's equation as well.

$$\gamma_1 Z \sum_i L_{U_i} + \mu_1 IZ = \mu_1 IX^T \tag{13}$$

This concludes the derivation. The pseudo-code can be expressed as follows:

loop k

$$X \leftarrow \min_X \left\| \begin{pmatrix} Y \\ \sqrt{\mu_1} Z^T \\ \sqrt{\mu_2} W \end{pmatrix} - \begin{pmatrix} R \cdot \\ \sqrt{\mu_1} I \\ \sqrt{\mu_2} I \end{pmatrix} X \right\|_F^2 + \lambda \|X\|_*$$

$$W \leftarrow Sylvester\left(\gamma_2 \sum_j L_{V_j}, \mu_2 I, \mu_2 IX\right)$$

$$Z \leftarrow Sylvester\left(\gamma_1 \sum_i L_{U_i}, \mu_1 I, \mu_1 IX^T\right)$$

End Loop

It must be noted that it appears that there is only one loop in the algorithm, but such is not the case. The updates for both nuclear norm minimization and solution to Sylvester's equations are iterative in nature and will have their internal loops.

5.4 LABEL-CONSISTENT NUCLEAR NORM MINIMIZATION

Traditionally, matrix completion algorithms in collaborative filtering focus solely on the rating matrix, overlooking the wealth of information available in user and item metadata. This metadata, such as age, gender, occupation, and item genres, holds immense potential for enhancing recommendation accuracy and personalization.

Label-consistent regularization, a technique inspired by supervised learning, integrates metadata into the matrix completion framework by introducing regularization terms that promote consistency between predicted ratings and the corresponding labels. This approach ensures that recommendations align with user preferences and item characteristics, leading to more relevant and effective suggestions.

For users, label-consistent regularization formulates penalties based on deviations between predicted ratings and labels derived from factors like age, gender, and occupation. This ensures that recommendations align with user demographics and preferences.

Similarly, for items, label-consistent regularization penalizes inconsistencies between predicted ratings and labels derived from item genres. This promotes consistency between item characteristics and user preferences, ensuring that recommended items match the user's interests.

The benefits of label-consistent regularization extend beyond improved accuracy. By enforcing consistency between ratings and labels, this approach enhances the interpretability of the recommendation system, providing insights into the factors influencing user behavior and recommendation outcomes.

Furthermore, the framework's ability to incorporate additional label information, such as social network connections, purchase history, and browsing behavior, makes it applicable to a wide range of scenarios. This versatility allows for tailoring recommendation strategies to specific domains and user preferences.

In essence, label-consistent regularization offers a powerful and versatile approach for incorporating metadata into matrix completion algorithms. By leveraging metadata to promote consistency between ratings and labels, this technique enhances the accuracy, interpretability, and applicability of collaborative filtering systems, paving the way for more personalized and effective recommendations.

Considering user metadata, we define multiple classes based on gender, age brackets, and different occupational profiles. A user can simultaneously belong to multiple classes. Using this labeling information, a user–class label consistency matrix $\left(L_u\right)$ is defined, such that $L_u\left(i,c\right)=1$ if user i belongs to class c and 0 otherwise. This class label matrix provides additional data to help predict the missing values in the rating matrix. The ratings are predicted under the add-on constraint of maintaining label consistency (appended as a regularization term) as shown next:

$$\min_{X,\mathrm{W}_u}\left\|Y-R\cdot(X)\right\|_F^2+\lambda\left\|X\right\|_*+\lambda_u\left\|L_u-XW_u\right\|_F^2 \tag{14}$$

where W_u is the linear map from user–item rating space to user–class space. It defines a relation between a user's class (assumed to reflect on his/her preferences) and the items; λ_u is the regularization parameter governing the relative importance given to rating data and demographic information.

A similar model can be built based on item metadata as well, establishing a relation between the item genre and the ratings given by the users. For items, multiple class labels are created based on different genres. Each item (movie, in this case) may belong to several classes (genres). A class–item label matrix $\left(L_v\right)$ similar to the user–class label matrix is constructed such that $L_v\left(c,j\right)=1$ if item j belongs to class c and 0 otherwise. Extending the formulation discussed in (15) to items, we propose an item metadata-based framework as follows:

$$\min_{X,\mathrm{W}_v}\left\|Y-R\cdot(X)\right\|_F^2+\lambda\left\|X\right\|_*+\lambda_v\left\|L_v-W_vX\right\|_F^2 \tag{15}$$

where W_v is the linear map (to be estimated) from user–item rating space to class–item space and λ_v is the regularization parameter.

In the final form, we club together both formulations to construct a model exploiting item and user metadata simultaneously (16).

$$\min_{X,W_v,W_u}\left\|Y-R\cdot(X)\right\|_F^2+\lambda\left\|X\right\|_*+\lambda_v\left\|L_v-W_vX\right\|_F^2+\lambda_u\left\|L_u-XW_u\right\|_F^2 \tag{16}$$

Next, we present the derivation of the algorithm for (16). It follows the framework of split Bregman. To enable the splitting of multiple norm terms, we introduce proxy variables (P and Q) in our formulation (17).

$$\begin{aligned}&\min_{X,W_v,W_v,P,Q}\left\|Y-R\cdot(X)\right\|_F^2+\lambda\left\|X\right\|_*+\lambda_v\left\|L_v-W_vP\right\|_F^2+\lambda_u\left\|L_u-QW_u\right\|_F^2\\&+\mu_v\left\|P-X-B1\right\|_F^2+\mu_u\left\|Q-X-B2\right\|_F^2\end{aligned}\tag{17}$$

where $B1$ and $B2$ are Bregman variables. The use of Bregman variables ensures that the equality between original and proxy variables need not be enforced (pushed) from the initial iteration itself. As Bregman variables are updated in subsequent iterations, it helps add back the error, thus making the algorithm self-correcting and helps in faster convergence. The use of proxy variables, along with alternating direction method of multipliers (ADMM) helps in splitting our formulation into simpler sub-problems as follows:

$$\begin{aligned}&\text{P1:}\min_X\left\|Y-R\cdot(X)\right\|_F^2+\lambda\left\|X\right\|_*+\mu_v\left\|P-X-B1\right\|_F^2+\mu_u\left\|Q-X-B2\right\|_F^2\\&\text{P2:}\min_P\lambda_v\left\|L_v-W_vP\right\|_F^2+\mu_v\left\|P-X-B1\right\|_F^2\\&\text{P3:}\min_Q\lambda_u\left\|L_u-QW_u\right\|_F^2+\mu_u\left\|Q-X-B2\right\|_F^2\\&\text{P4:}\min_{W_v}\left\|L_v-W_vP\right\|_F^2\\&\text{P5:}\min_{W_u}\left\|L_u-QW_u\right\|_F^2\end{aligned}$$

Sub-problem P1 can be recast as

$$\min_Z\left\|\begin{pmatrix}Y\\\sqrt{\mu_v}\left(P-B1\right)\\\sqrt{\mu_u}\left(Q-B2\right)\end{pmatrix}-\begin{pmatrix}R\cdot\\\sqrt{\mu_v}I\\\sqrt{\mu_u}I\end{pmatrix}X\right\|_F^2+\lambda\left\|Z\right\|_*\tag{18}$$

Equation (18) can be solved by soft thresholding of singular values as follows,

$$X\leftarrow Soft\left(Singular\ value\left(X+{}^{1}\!/_{\alpha}\left(\begin{pmatrix}R\cdot\\\sqrt{\mu_v}I\\\sqrt{\mu_u}I\end{pmatrix}^T\left(\begin{pmatrix}Y\\\sqrt{\mu_v}\left(P-B1\right)\\\sqrt{\mu_u}\left(Q-B2\right)\end{pmatrix}-\begin{pmatrix}R\cdot\\\sqrt{\mu_v}I\\\sqrt{\mu_u}I\end{pmatrix}Z\right)\right)\right),\frac{\lambda}{2\alpha}\right)\tag{19}$$

where $Soft\left(t,u\right)=sign\left(t\right)\ \max\left(0,\left|t\right|-u\right)$ and $\alpha\geq\max\left(eig\begin{pmatrix}A\\\sqrt{\mu_v}I\\\sqrt{\mu_u}I\end{pmatrix}^T\begin{pmatrix}A\\\sqrt{\mu_v}I\\\sqrt{\mu_u}I\end{pmatrix}\right)$.

Sub-problem P2 can be recast as a least square expression (20)

$$\min_{P} \left\| \begin{pmatrix} \sqrt{\lambda_v} L_v \\ \sqrt{\mu_v}\left(X + B1\right) \end{pmatrix} - \begin{pmatrix} \sqrt{\lambda_v} W_v \\ \sqrt{\mu_v} I \end{pmatrix} P \right\|_F^2 \tag{20}$$

Similarly, sub-problem P3 can be recast as follows:

$$\min_{Q} \left\| \begin{pmatrix} \sqrt{\lambda_u} L_u \\ \sqrt{\mu_u}\left(X + B2\right) \end{pmatrix} - Q \begin{pmatrix} \sqrt{\lambda_v} W_u \\ \sqrt{\mu_u} I \end{pmatrix} \right\|_F^2 \tag{21}$$

Equations (20) and (21) are simple least square expressions that can be efficiently solved using any conjugate gradient-type solver.

Solving for W_v and W_u is also fairly straightforward using least square expressions in P4 and P5.

After every iteration of each of the sub-problems, Bregman variables are updated as follows:

$$B2 = B2 + Z - Q \tag{22}$$
$$B1 = B1 + Z - P \tag{23}$$

The iterations continue till either the objective function converges or the maximum number of iterations is reached. The complete algorithm is given as follows:

Initialize variables

Continue till maximum number of iterations reached

//Solve for X

$$X \leftarrow Soft\left(Singular\ value\left(X + \frac{1}{\alpha}\left(\begin{pmatrix} R\cdot \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix}^T \left(\begin{pmatrix} Y \\ \sqrt{\mu_v}\left(P - B1\right) \\ \sqrt{\mu_u}\left(Q - B2\right) \end{pmatrix} - \begin{pmatrix} R\cdot \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix} Z \right) \right) \right), \frac{\lambda}{2\alpha} \right)$$

$$P \leftarrow solve\left(\begin{pmatrix} \sqrt{\lambda_v} L_v \\ \sqrt{\mu_v}\left(X + B1\right) \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_v} W_v \\ \sqrt{\mu_v} I \end{pmatrix} P \right)$$

$$\mathrm{Q} \leftarrow Solve\left(\begin{pmatrix} \sqrt{\lambda_u} L_u \\ \sqrt{\mu_u}\left(X + B2\right) \end{pmatrix} = Q \begin{pmatrix} \sqrt{\lambda_v} W_u \\ \sqrt{\mu_u} I \end{pmatrix} \right)$$

$$W_v \leftarrow \min_{W_v} \left\| L_v - W_v P \right\|_F^2$$

$$W_u \leftarrow \min_{W_u} \left\| L_u - Q W_u \right\|_F^2$$

//Update Bregman Variables

$B2 = B2 + Z - Q$

$B1 = B1 + Z - P$

End

5.5 LABEL-CONSISTENT MATRIX FACTORIZATION

Similar to the nuclear norm minimization approach, we can introduce label consistency into matrix factorization for collaborative filtering. This framework is based on the assumption that users can be grouped into distinct classes based on available metadata, such as age, gender, and occupation. As in the previous approach, we form classes based on these demographic factors, allowing users to belong to multiple classes simultaneously.

This class-label information is utilized to learn the latent factor vectors of users in a supervised manner. The learning process aims to ensure consistency between the latent factor vectors and the corresponding class labels. This consistency promotes alignment between predicted ratings and the underlying user preferences associated with their demographic characteristics.

By incorporating label consistency into matrix factorization, we enhance the accuracy and interpretability of the recommendation system. The regularization terms introduced by label consistency penalize deviations between predicted ratings and the corresponding labels, leading to more precise and relevant recommendations. Additionally, the framework provides insights into the factors influencing user behavior and recommendation outcomes.

This approach extends the applicability of matrix factorization by enabling the incorporation of additional label information, such as social network connections, purchase history, and browsing behavior. This versatility allows for tailoring recommendation strategies to specific domains and user preferences, making the framework adaptable to a wide range of scenarios.

To conclude, incorporating label consistency into matrix factorization offers a powerful and versatile approach for enhancing collaborative filtering algorithms. By leveraging metadata to promote consistency between ratings and labels, this technique improves the accuracy, interpretability, and applicability of recommendation systems, paving the way for more personalized and effective recommendations.

We model the above ideas as an additional regularization term added to the base formulation which penalizes any deviation from class label consistency as shown in (9).

$$\min_{U,V,C} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_F^2 + \mu_u \left\| W - UC \right\|_F^2 \tag{24}$$

where $W \in R^{M \times C_u}$, $C \in R^{F \times C_u}$, C_u is the number of classes constructed based on user metadata. Class information matrix (W) is constructed such that $W_{i,j} = 1$ if user i belongs to class j else 0. The matrix C is learned as part of the optimization process as a linear map from latent factor space to the classification domain. The user's latent factor matrix is learned to be consistent with the class label information.

The idea can be extended to items as well. All items can be categorized based on their genre, with each item belonging to one or multiple genres. Similar to (24), item information can be included as given in (25):

$$\min_{U,V,A} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_F^2 + \mu_v \left\| Q - AV \right\|_F^2 \tag{25}$$

where $Q \in R^{C_v \times N}$, $A \in R^{C_v \times F}$, C_v is the number of classes constructed based on item genres. Class information matrix (Q) is constructed such that $Q_{i,j} = 1$ if movie (item) i belongs to class (genre) j else 0.

The matrix A forms a linear map from the latent factor domain to the classification domain. Item's latent factor matrix is learned so as to be consistent with the available class label (genre) information.

We can also utilize item and user metadata together. Thus, one can combine formulation in (24) and (25) to yield our final formulation:

$$\min_{U,V,C,A} \left\|Y - R\cdot(UV)\right\|_F^2 + \lambda_u \left\|U\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \mu_u \left\|W - UC\right\|_F^2 + \mu_v \left\|Q - AV\right\|_F^2 \quad (26)$$

In our design latent factor vectors are learned in such a way that they satisfy (rating) data consistency as well as class discrimination.

We design an efficient algorithm for our formulation (26) using split Bregman and majorization-minimization techniques.

The use of the MM approach enables us to break complex optimization problems into simpler and more efficiently solvable steps.

$$\min_{U,V,C,A} \left\|Y - R\cdot(UV)\right\|_F^2 + \lambda_u \left\|U\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \mu_u \left\|W - UC\right\|_F^2 + \mu_v \left\|Q - AV\right\|_F^2$$

We first employ the method of alternating direction to split our formulation into separate simpler sub-problems, each minimizing over a single variable.

$$\text{P1:}\min_U \left\|Y - R\cdot(UV)\right\|_F^2 + \lambda_u \left\|U\right\|_F^2 + \mu_u \left\|W - UC\right\|_F^2 \quad (27)$$

$$\text{P2:}\min_V \left\|Y - R\cdot(UV)\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \mu_v \left\|Q - AV\right\|_F^2 \quad (28)$$

$$\text{P3:}\min_C \left\|W - UC\right\|_F^2 \quad (29)$$

$$\text{P4:}\min_A \left\|Q - AV\right\|_F^2 \quad (30)$$

Now considering P1, the same can be cast as the least square problem. It has a closed form solution, however, given the large size of rating matrices, computing the pseudo inverse becomes a computational challenge. To remove the need for such prohibitively large computations, we use the MM technique.

For the sub-problem in (27), we can replace the original problem by another function (31):

$$\begin{aligned}&\min_U \left\|Y - R\cdot(UV)\right\|_F^2 + \lambda_u \left\|U\right\|_F^2 + \mu_u \left\|W - UC\right\|_F^2 + (Z - Z_k)^T \\ &\left(\beta I - \bar{R}^{\cdot T}\,\bar{R}\right)(Z - Z_k)\end{aligned} \quad (31)$$

where $Z = vec(UV)$ and $\bar{R}$ is the block diagonal form of masking operator R. The add-on term being always non-negative ensures the new function is always a majorizer of the existing one. After some mathematical manipulations, it can be shown that (16) can be reduced to the form given in (17).

$$\min_U \|S - UV\|_F^2 + \lambda_u \|U\|_F^2 + \mu_u \|W - UC\|_F^2 \tag{32}$$

where $S = U_k V_k + Y - R \cdot (U_k V_k)$

Equation (32) is a simple least square that can be solved easily.

In the same fashion, using majorization-minimization we can rewrite (28) as

$$\min_V \|S - UV\|_F^2 + \lambda_v \|V\|_F^2 + \mu_v \|Q - AV\|_F^2 \tag{33}$$

which can be recast as a least squares problem as well.

Equations (29) and (30) are simple least squares efficiently solvable by any conjugate gradient method.

Both the sub-problems are alternately solved till a desired number of iterations are complete or the objective function converges. The complete algorithm is as follows:

Initialize; variables and parameters;

Set maximum no. of iterations, N

Till convergence

$U \leftarrow \min_U \|Y - R \cdot (UV)\|_F^2 + \lambda_u \|U\|_F^2 + \mu_u \|W - UC\|_F^2$

$V \leftarrow \min_V \|Y - R \cdot (UV)\|_F^2 + \lambda_v \|V\|_F^2 + \mu_v \|Q - AV\|_F^2$

$C \leftarrow \min_C \|W - UC\|_F^2$

$A \leftarrow \min_A \| Q - AV\|_F^2$

End

All the problems can be efficiently solved using a conjugate gradient. The algorithm is run till convergence. By that, we mean one can either run it for a fixed number of iterations or stop the iterations when the objective function does not change considerably in further iterations.

APPENDIX

Collaborative filtering (CF) has emerged as a cornerstone of modern recommendation systems, effectively predicting user preferences and delivering personalized suggestions. Two primary approaches dominate the CF landscape: neighborhood-based models and latent factor-based models. Each approach offers distinct advantages and limitations, leading to the development of hybrid models that combine the strengths of both approaches.

Neighborhood-based models, also known as local models, establish recommendations based on the similarity between users or items. They operate on the principle that users with similar preferences or items with similar characteristics are likely to be favored by the same individual.

User-based CF identifies similar users based on their rating history and recommends items that similar users have rated highly. This approach captures the local preferences of users within their immediate neighborhood.

Item-based CF identifies similar items based on their attribute profiles or collaborative filtering metrics and recommends items that are similar to those the user has rated highly. This approach captures the local relationships between items.

Latent factor-based models, also known as global models, assume that user preferences and item characteristics can be represented by a low-dimensional latent factor space. They infer these latent factors from the rating matrix, enabling the prediction of missing ratings and the generation of personalized recommendations.

Matrix factorization decomposes the rating matrix into two matrices, one representing user latent factors and the other representing item latent factors. The predicted ratings are then obtained by multiplying these two matrices.

Nuclear norm minimization assumes that the latent factor matrices have a low nuclear norm, which implies that the latent factors should be shared across users and items. This approach promotes the smoothness and interpretability of the latent factors.

Hybrid CF models combine the strengths of neighborhood-based and latent factor-based approaches, aiming to achieve both personalization and scalability.

Graph regularization incorporates neighborhood information into matrix factorization by constructing graphs from user and item similarities. This approach regularizes the factorization process to ensure that the latent factors are consistent with the local neighborhood structure. While hybrid CF models have demonstrated promising results, they face challenges related to graph construction. Since the graphs are based on a partially observed rating matrix, they may not accurately represent the true neighborhood structure.

For example, consider a situation of three users—U1, U2, and U3. Here are the actual ratings of these users on items:

U1: [1 0 0 1 0 1 1 0]
U2: [1 1 1 1 1 0 0 0]
U3: [0 0 0 1 0 0 1 0]

In collaborative filtering, the ratings are not fully observed. Assume that the partially observed ratings are

U1: [1 x x 1 x x 1 x]
U2: [1 x x 1 x x x 0]
U3: [0 x x x x 0 x 0]

Going by whatever we have learned so far, the similarity between U1 and U2 computed from the partially observed ratings will be high (Hamming distance: 0) and the similarity between U1 and U3 will be low (Hamming distance: 1). However, when the similarities are computed from the fully observed ratings, one can see that the similarity between U1 and U3 is actually lower (Hamming distance: 2) compared to the similarity between U1 and U2 (Hamming distance:

5). This toy example shows the importance of computing the similarities (thereby the graph Laplacians) from the completed matrix, rather than the partially observed one. The similarities and hence the generated graphs are likely to be incorrect when computed from the partially observed data. Unfortunately, the completed matrix is not available; in this scenario, one reasonable approach is to adaptively learn the similarities (and hence the graph) during the completion of the rating's matrix.

In the realm of collaborative filtering, matrix completion algorithms play a central role in predicting missing ratings and generating personalized recommendations. Traditional approaches often rely on fixed graphs, constructed from the partially observed rating matrix, as regularizers to guide the matrix completion process. However, these fixed graphs may not accurately represent the underlying user and item relationships due to the inherent sparsity of the ratings data.

To address this limitation, dynamic graph-based matrix completion introduces an iterative and adaptive approach that enhances the quality of the graphs alongside the matrix completion process. Unlike traditional methods, the graphs are not treated as fixed inputs but are continuously updated based on the most recent estimates of the rating matrix.

The algorithm commences with a fixed graph constructed from the partially observed ratings. This initial graph serves as a starting point for the iterative process. In each iteration, the following steps are performed:

1. **Matrix Completion:** Utilizing the current graph as a regularizer, the matrix completion algorithm estimates the missing ratings. This updated rating matrix provides a refined representation of user preferences and item characteristics.
2. **Graph Update:** Based on the updated rating matrix, the graph is reconstructed to reflect the most recent information about user and item similarities. This updated graph captures the evolving relationships between entities as the matrix completion process progresses.
3. **Iteration:** The updated graph is then employed as a regularizer in the subsequent iteration, leading to a further refinement of the rating matrix. This iterative process continues until convergence is achieved.

This dynamic graph-based approach offers several advantages over traditional methods:

1. **Adaptive Graph Representation:** The iterative update of the graph ensures that the regularizers adapt to the evolving rating matrix, capturing the most up-to-date user and item relationships.
2. **Improved Accuracy:** By continuously refining the graph and the rating matrix, the algorithm achieves enhanced prediction accuracy and personalization.
3. **Enhanced Interpretability:** The dynamic graph provides insights into the evolving relationships between users and items, offering a more transparent understanding of the recommendation process.

Dynamic graph-based matrix completion represents a significant advancement in collaborative filtering, enabling more accurate, personalized, and interpretable recommendation systems. By adopting the graph representation alongside the matrix completion process, this approach effectively captures the underlying structure of user preferences and item characteristics, paving the way for delivering exceptional user experiences and driving business success.

MATRIX FACTORIZATION

In matrix factorization-based approaches, the rating matrix X is modeled as a product of the user latent factor matrix U and the item latent factor matrix V, that is, $X = UV$. The complete rating matrix needs to be estimated; we only have a partially observed version of it, modeled as

$$Y = R \cdot (X) = R \cdot (UV) \tag{1}$$

Here, R is a binary mask that has ones in positions where the ratings are observed and zeroes elsewhere. Simple matrix factorization-based techniques solve the following to recover the factors

$$\arg\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + C(U,V) \tag{2}$$

where $C(.)$ is some regularization over the variables—usually the Frobenius norm.

Graph Laplacian penalties, on the other hand, impose similarity among users, that is, it enforces smoothness among similar users.

$$\arg\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + \gamma \left(Tr\left(U^T L_U U\right) + Tr\left(V L_V V^T\right) \right) \tag{3}$$

where L_U and L_V are the graph Laplacians computed along the rows and along the columns respectively (to be defined shortly). The graph Laplacians embed the similarity information of neighborhood-based models of collaborative filtering.

In the realm of collaborative filtering, the quest for accurate and personalized recommendations often leads to a trade-off between interpretability and predictive power. Neighborhood-based models offer interpretability by explicitly capturing local relationships between users or items, while latent factor models provide superior predictive power by representing user preferences and item characteristics in a latent space.

Seeking to bridge this gap, graph regularization emerges as a promising approach that combines the strengths of both neighborhood-based and latent factor models. By incorporating graph-based regularizers into matrix completion algorithms, we can simultaneously enhance predictive accuracy and interpretability.

However, a critical challenge arises when constructing these graphs from incomplete data. Traditional methods often employ fixed graphs derived from the partially observed rating matrix. As we have discussed, such estimates may be erroneous due to the inherent sparsity of the data.

To overcome this limitation, we propose an iterative approach that alternates between completing the matrix assuming a fixed graph and updating the graph based on the latest matrix completion results. This dynamic process continuously refines both the ratings matrix and the graph representation, ensuring that the regularizers capture the most up-to-date information about user and item relationships.

While this approach may not fully mitigate the inherent challenges of incomplete data, it represents a significant step forward in striking a balance between interpretability and predictive power in matrix completion for collaborative filtering. By leveraging graph regularization in an adaptive manner, we can achieve more accurate, personalized, and interpretable recommendation systems.

The mathematical formulation behind this intuition is expressed formally as

$$\begin{aligned}&\arg\min_{U,V,W_U,W_V} \left\|Y - R\cdot(UV)\right\|_F^2 + \gamma\left(\left\|W_U \circ Z_U\right\|_{1,1} + \left\|W_V \circ Z_V\right\|_{1,1}\right)\\&+\gamma\sigma^2\sum_{i,j} W(i,j)_U\left(\log(W(i,j)_U)-1\right)+\gamma\sigma^2\sum_{i,j} W(i,j)_V\left(\log(W(i,j)_V)-1\right)\end{aligned} \tag{4}$$

Here, $Z_X(i,j) = \left\|x_i - x_j\right\|_2^2$; Z_U is computed along the rows and Z_V along the columns. The definitions $\left\|W_U \circ Z_U\right\|_{1,1} = Trace\left(U^T L_U U\right)$ and $\left\|W_V \circ Z_V\right\|_{1,1} = Trace\left(VL_V V^T\right)$ are given by Kalofolias (2016), by using the standard Laplacian operator's definition $L_U = \Delta_U - W_U$, where Δ_U is a diagonal matrix with i^{th} diagonal term equals to $\Delta_U(i,i) = \sum_j W_U(i,j)$, and $L_V = \Delta_V - W_V$ with $\Delta_V(i,i) = \sum_j W_V(j,i)$. The terms $\sigma^2\sum_{i,j} W(i,j)_U\left(\log(W(i,j)_U)-1\right)$ and $\sigma^2\sum_{i,j} W(i,j)_V\left(\log(W(i,j)_V)-1\right)$ are required for learning the graph.

One can see that (4) is a complete formulation that jointly learns the latent factor matrices and estimates the graphs. We concentrate on its solution next. The first step is to remove the under-determinacy by majorization-minimization. This leads to the following in every iteration (k):

$$\begin{aligned}&\arg\min_{U,V,W_U,W_V} \left\|B - UV\right\|_F^2 + \gamma\left(\left\|W_U \circ Z_U\right\|_{1,1} + \left\|W_V \circ Z_V\right\|_{1,1}\right)\\&+\gamma\sigma^2\sum_{i,j} W(i,j)_U\left(\log(W(i,j)_U)-1\right)+\gamma\sigma^2\sum_{i,j} W(i,j)_V\left(\log(W(i,j)_V)-1\right)\end{aligned} \tag{5}$$

where $B = \left(UV\right)_{k-1} + Y - R\cdot\left(UV\right)_{k-1}$

In the k^{th} iteration (5) is solved using alternating direction method of multipliers (ADMM). Using ADMM, it can be segregated into the following sub-problems,

$$\arg\min_U \left\|B - UV\right\|_F^2 + \gamma\left\|W_U \circ Z_U\right\|_{1,1} \equiv \arg\min_U \left\|B - UV\right\|_F^2 + \gamma Trace\left(U^T L_U U\right) \tag{6}$$

$$\arg\min_V \left\|B - UV\right\|_F^2 + \gamma\left\|W_V \circ Z_V\right\|_{1,1} \equiv \arg\min_V \left\|B - UV\right\|_F^2 + \gamma Trace\left(VL_V V^T\right) \tag{7}$$

$$\arg\min_{W_U} \left\| W_U \circ Z_U \right\|_{1,1} + \sigma^2 \sum_{i,j} W(i,j)_U \left(\log(W(i,j)_U) - 1 \right) \tag{8}$$

$$\arg\min_{W_V} \left\| W_V \circ Z_V \right\|_{1,1} + \sigma^2 \sum_{i,j} W(i,j)_V \left(\log(W(i,j)_V) - 1 \right) \tag{9}$$

The solutions for (6) and (7) are similar. We will derive (6) in detail and (7) will follow. To compute the update for U in the l^{th} iteration (within k), we need to compute the derivative of (6) and equate it to 0. This leads to

$$\gamma L_U U + U V^{(l-1)} \left(V^{(l-1)} \right)^T = B \left(V^{(l-1)} \right)^T \tag{10}$$

It is the well-known Sylvester's equation. Many efficient solutions exist. Similarly, the solution for V can be expressed as Sylvester's equation:

$$\left(U^{(l)} \right)^T U^{(l)} V + \lambda V L_V = \left(U^{(l)} \right)^T B \tag{11}$$

The updates for (8) and (9) are also similar. The solution to the general form is given by Kalofolias (2016). We repeat it here for the sake of convenience.

$$W(i,j) = \exp\left(- \frac{\left\| x_i - x_j \right\|_2^2}{\sigma^2} \right) \tag{12}$$

W_U (12) is computed along the rows and for W_V along the columns.

This completes the derivation of the algorithm. Succinctly, it is shown next. In the algorithm, '→' denotes along rows, and '↓' denotes along columns.

$$Initialize: W_U(i,j) = \exp\left(- \frac{\left\| x_i^{\rightarrow} - x_j^{\rightarrow} \right\|_2^2}{\sigma^2} \right) \text{ and } W_V(i,j) = \exp\left(- \frac{\left\| x_i^{\downarrow} - x_j^{\downarrow} \right\|_2^2}{\sigma^2} \right)$$

from partial data.

In iteration k

$$\text{Compute: } B = (UV)_{k-1} + Y - R \cdot (UV)_{k-1}$$

In iteration l

$$U \leftarrow Sylvester\left(\lambda L_U, V^{(l-1)} \left(V^{(l-1)} \right)^T, B \left(V^{(l-1)} \right)^T \right)$$

$$V \leftarrow Sylvester\left(\left(U^{(l)} \right)^T U, \lambda L_V, \left(U^{(l)} \right)^T B \right)$$

$$X = UV$$

$$W_U(i,j) = \exp\left(-\frac{\left\|x_i^{\rightarrow} - x_j^{\rightarrow}\right\|_2^2}{\sigma^2}\right) \text{ from completed data}$$

$$W_V(i,j) = \exp\left(-\frac{\left\|x_i^{\downarrow} - x_j^{\downarrow}\right\|_2^2}{\sigma^2}\right) \text{ from completed data}$$

End loop (l)
End loop (k)

The algorithm in essence proceeds in two loops. The outer loop decouples the sampling operation. The inner loop solves the problem of matrix factorization on learned graphs. In the inner loop, the first three steps correspond to standard matrix factorization on graphs and the last two steps are for learning the graphs from the matrix. We run the inner loop for a fixed number of iterations and the outer loop to a local convergence.

Nuclear Norm Minimization

Matrix factorization is a bi-linear problem and hence is naturally non-convex. A convex formulation for completing a low rank matrix was proposed via nuclear norm minimization and is expressed as

$$\min_X \left\|Y - R(X)\right\|_F^2 + \mu\left\|X\right\|_* \tag{13}$$

Here, $\left\|X\right\|_*$ is defined as the sum of the singular values; it is the tightest convex envelope of the rank of a matrix.

The work on matrix completion on graphs via nuclear norm minimization has been discussed before. The formulation was

$$\min_X \left\|Y - R(X)\right\|_F^2 + \mu\left\|X\right\|_* + \lambda\left(Tr\left(XL_UX^T\right) + Tr\left(X^TL_VX\right)\right) \tag{14}$$

All the variables have been defined before. In our proposed formulation, we do not assume that the graph Laplacians L_U and L_V are given; we learn it from the data in a progressive fashion; the concept remains similar to that of our factorization formulation (4). We only replace the completion task from the bi-linear matrix factorization model (4) with that of nuclear norm minimization (14):

$$\begin{aligned}&\arg\min_{X,W_U,W_V} \left\|Y - R\cdot(X)\right\|_F^2 + \lambda\left\|X\right\|_{NN} + \gamma\left\|W_U \circ Z_U\right\|_{1,1} + \gamma\left\|W_V \circ Z_V\right\|_{1,1} \\ &+\gamma\sigma^2\sum_{i,j} W(i,j)_U\left(\log(W(i,j)_U) - 1\right) + \gamma\sigma^2\sum_{i,j} W(i,j)_V\left(\log(W(i,j)_V) - 1\right)\end{aligned} \tag{15}$$

The derivation proceeds in a manner similar to that of factorization. As before, the outer loop consists of majorization-minimization of (15) leading to

$$\begin{aligned}&\arg\min_{X,W_U,W_V}\left\|B-X\right\|_F^2+\lambda\left\|X\right\|_{NN}+\gamma\left\|W_U\circ Z_U\right\|_{1,1}+\gamma\left\|W_V\circ Z_V\right\|_{1,1}\\&+\gamma\sigma^2\sum_{i,j}W(i,j)_U\left(\log(W(i,j)_U)-1\right)+\gamma\sigma^2\sum_{i,j}W(i,j)_V\left(\log(W(i,j)_V)-1\right)\end{aligned}\tag{16}$$

where $B=X_{k-1}+Y-R\cdot\left(X_{k-1}\right)$.

In iteration k (16) is solved by ADMM. The sub-problems for updating W_U and W_V remain the same as before (8/9); hence we do not repeat them. The only difference is in the update for the matrix completion problem. This is given by

$$\begin{aligned}&\arg\min_X\left\|B-X\right\|_F^2+\lambda\left\|X\right\|_{NN}+\gamma\left(\left\|W_U\circ Z_U\right\|_{1,1}+\left\|W_V\circ Z_V\right\|_{1,1}\right)\\&\equiv\arg\min_X\left\|B-X\right\|_F^2+\lambda\left\|X\right\|_{NN}+\gamma\left(Trace\left(X^TL_UX\right)+Trace\left(XL_VX^T\right)\right)\end{aligned}\tag{17}$$

Solving (17) is not straightforward. We need to resort to Bregman variable splitting. We introduce two proxy variables $F = X^T$ and $G = X$ in the terms $Trace\left(X^TL_UX\right)$ and $Trace\left(XL_VX^T\right)$, respectively. The equality between the proxies and the variables is relaxed by the Bregman variables C_1 and C_2, respectively. The complete formulation is given by

$$\begin{aligned}&\arg\min_{X,F,G}\left\|B-X\right\|_F^2+\lambda\left\|X\right\|_{NN}+\gamma\left(Trace\left(FL_UF^T\right)+Trace\left(GL_VG^T\right)\right)\\&+\alpha\left\|F^T-C_1-X\right\|_F^2+\beta\left\|G-C_2-X\right\|_F^2\end{aligned}\tag{18}$$

To solve (18), one needs to resort to ADMM once more. The sub-problems are expressed as follows:

$$\arg\min_X\left\|B-X\right\|_F^2+\lambda\left\|X\right\|_{NN}+\alpha\left\|F^T-C_1-X\right\|_F^2+\beta\left\|G-C_2-X\right\|_F^2\tag{19}$$

$$\arg\min_F\gamma Trace\left(FL_UF^T\right)+\alpha\left\|F-C_1^T-X^T\right\|_F^2\tag{20}$$

$$\arg\min_G\gamma Trace\left(GL_VG^T\right)+\beta\left\|G-C_2-X\right\|_F^2\tag{21}$$

The update for (17) is one step of singular value shrinkage:

$$\begin{aligned}&B+\alpha\left(F^T-C_1\right)+\beta\left(G-C_2\right)=USV^T\\&\Sigma=diag\max\left(diag(S)-\frac{\lambda}{2},0\right)\\&X=U\Sigma V^T\end{aligned}$$

The update for (20) can be obtained by taking its derivative and equating it to 0.

$$\gamma F L_U + \alpha F = C_1^T + X^T \tag{22}$$

This turns out to be a Sylvester's equation. Similarly, the solution to (21) also turns out to be a Sylvester's equation.

$$\gamma G L_V + \beta G = C_2 + X \tag{23}$$

The final step is to update the Bregman relaxation variables:

$$C_1 \leftarrow F^T - C_1 - X \tag{24}$$

$$C_2 \leftarrow G - C_2 - X \tag{25}$$

This completes the derivation. A succinct version of the algorithm is given next.

Initialize : $W_U(i,j)$ and $W_V(i,j)$ as before
In iteration k
Compute: $B = (X)_{k-1} + Y - R \cdot (X)_{k-1}$
In iteration l
In iteration m
Initialize C1 and C2

$$B + \alpha\left(F^T - C_1\right) + \beta\left(G - C_2\right) = USV^T$$

$$\Sigma = diag \max\left(diag(S) - \frac{\lambda}{2}, 0\right)$$

$$X = U\Sigma V^T$$

$$F \leftarrow Sylvester\left(\gamma L_U, \alpha I, C_1^T + X^T\right)$$

$$G \leftarrow Sylvester\left(\gamma L_V, \beta I, C + X\right)$$

$$C_1 \leftarrow F^T - C_1 - X$$

$$C_2 \leftarrow G - C_2 - X$$

End loop (m)
Update $W_U(i,j)$ and $W_V(i,j)$ as before
End loop (l)
End loop (k)

This algorithm proceeds in three loops. In the outer loop, the problem is decoupled as in the matrix factorization case. In the second loop, the similarities and hence the graph Laplacians are updated. In the innermost loop, the nuclear norm minimization on graphs is solved. In contrast to the matrix factorization approach, the current

one requires three loops since the nuclear norm minimization on graphs needs to be solved iteratively. The two inner loops are run for a fixed number of iterations and the outer loop till local convergence.

BIBLIOGRAPHY

Gogna, A. and Majumdar, A., 2015a. A comprehensive recommender system model: Improving accuracy for both warm and cold start users. *IEEE Access*, 3, pp. 2803–2813.

Gogna, A. and Majumdar, A., 2015b. Matrix completion incorporating auxiliary information for recommender system design. *Expert Systems with Applications*, 42(14), pp. 5789–5799.

Gu, Q., Zhou, J. and Ding, C., 2010, April. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM international conference on data mining* (pp. 199–210). Society for Industrial and Applied Mathematics.

Han, D. and Yuan, X., 2012. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155, pp. 227–238.

Kalofolias, V., 2016, May. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics* (pp. 920–929). Proceedings of Machine Learning Research.

Mongia, A. and Majumdar, A., 2019. Matrix completion on multiple graphs: Application in collaborative filtering. *Signal Processing*, 165, pp. 144–148.

6 Diversity in Recommender Systems

6.1 INTRODUCTION

The accuracy of rating prediction has long been the cornerstone of recommender system (RS) evaluation, with the assumption that recommending items that closely match a user's past preferences would lead to higher satisfaction and engagement. While this approach has its merits, it has also led to several limitations in the effectiveness of RS, particularly in terms of breadth and diversity of recommendations.

6.1.1 The Monotony of Accuracy-Driven Recommendations

A fundamental issue with solely focusing on accuracy is that it can lead to a monotonous and repetitive recommendation experience. Recommending items that are highly similar to a user's past choices may initially provide a sense of familiarity and validation, but over time, it can become predictable and uninspiring. Users may feel like they are being confined to a narrow echo chamber of their preferences, missing out on the vast array of items that lie beyond their immediate purview.

6.1.2 The Dilemma of Popularity Bias

Another challenge associated with accuracy-driven recommendations is the tendency of algorithms to favor highly popular or frequently rated items. This popularity bias arises from the abundance of data available for these items, making them easier for algorithms to predict and recommend. However, this approach neglects the potential of less popular items, which may offer unique and valuable experiences to users. By constantly reinforcing the popularity hierarchy, RS risks overlooking hidden gems and failing to fulfill their role as curators of diverse experiences.

6.1.3 The Business Case for Diversity

The limitations of accuracy-driven recommendations extend beyond user satisfaction and encompass broader business objectives. Online portals rely on RS to increase product visibility and drive sales, and recommending only popular items limits the potential for discovering and promoting lesser-known products. A more diverse set of recommendations can attract a wider range of users, introducing them to new products and potentially expanding their interests. This can lead to increased sales, higher customer engagement, and a stronger brand reputation.

DOI: 10.1201/9781003511267-6

6.1.4 The Quest for Optimal Breadth and Diversity

The shortcomings of accuracy-centric RS have led to a growing movement toward designing systems that prioritize breadth and diversity of recommendations. This shift requires a redefinition of evaluation metrics and a focus on algorithms that can effectively explore the vast space of potential recommendations while maintaining a reasonable level of accuracy.

6.1.5 Balancing Accuracy and Diversity

Achieving a balance between accuracy and diversity is a complex challenge. Simply increasing the variety of recommendations may come at the cost of significantly reduced accuracy, potentially frustrating users who rely on RS for reliable suggestions. The key lies in finding algorithms that can effectively navigate the trade-off between these two objectives, maximizing diversity without compromising accuracy beyond an acceptable threshold.

6.1.6 The Future of Recommender Systems: diversity as a Driver of Value

As RS continues to evolve, the importance of breadth and diversity will only grow. By embracing these qualities, RS can move beyond merely replicating user preferences and become a true discovery engine, introducing users to new and exciting experiences while expanding their horizons. This shift will not only enhance user satisfaction but also drive business growth and establish RS as an essential tool for exploration and innovation.

6.1.7 Balancing Accuracy and Diversity in Recommender Systems

Balancing accuracy and diversity is a critical challenge in recommender system (RS) design. While accuracy measures the ability of the RS to predict user preferences accurately, diversity ensures that the recommendations are not limited to a user's past choices and provide them with exposure to new and potentially interesting items.

6.1.8 Two-Stage Recommendation Strategies

Two-stage recommendation strategies address the balance between accuracy and diversity by decoupling the recommendation process into two distinct phases. The first stage employs a conventional collaborative filtering (CF) technique to predict missing ratings for items that a user has not interacted with in the past. These predicted ratings are then used as input for the second stage, which implements a modified ranking strategy that promotes diversification.

The second stage typically involves setting a heuristic ranking threshold, and items with predicted ratings above the threshold are considered potential candidates for recommendation. This approach offers the advantage of flexibility, allowing the use of existing CF techniques for rating prediction. However, it cannot guarantee an optimal solution and may increase computational costs due to the two-stage process.

6.1.9 Unified Models

Unified models, on the other hand, present a single-stage framework that integrates accuracy and diversity considerations into a unified optimization problem. These models typically employ a weighted combination of diversity and accuracy-promoting functions, where the weights determine the relative importance of each objective.

Unified models offer several advantages over two-stage approaches. They provide a theoretically sound framework for balancing accuracy and diversity, ensuring that the optimal solution is achieved within the constraints of the joint formulation. Additionally, they eliminate the need for an intermediate ranking stage, reducing computational overhead.

Despite their theoretical advantages, unified models have not been as extensively explored as two-stage approaches. This is partly due to the challenges in designing and implementing effective optimization algorithms for the joint formulation.

The studies discussed here fall under the category of unified models. We propose a novel unified model that leverages a carefully designed optimization algorithm to balance accuracy and diversity effectively. Our approach addresses the limitations of existing unified models by introducing a new diversity metric that captures the novelty and uniqueness of recommended items. Additionally, we propose a novel regularization technique to prevent overfitting and ensure the robustness of our model.

6.1.10 Individual Diversity (ID) and Aggregate Diversity (AD)

Two primary metrics are used to evaluate the diversity of RS recommendations: individual diversity (ID) and aggregate diversity (AD). ID measures the breadth of items recommended to a specific user, ensuring that they are not confined to a narrow range of similar items. On the other hand, AD measures the overall diversity of recommendations across all users, ensuring that the system is not recommending the same set of popular items to everyone.

6.1.11 The Interplay of ID and AD

While both ID and AD are important for a successful RS, they are not always in perfect alignment. A system that prioritizes ID may achieve high diversity for individual users by recommending items from a wide range of categories or genres. However, this approach may not be beneficial from the system's perspective if the same diverse items are recommended to a large number of users, leading to reduced visibility for less popular items and potentially impacting profitability.

6.1.12 Balancing ID and AD for Optimal RS Design

Effective RS design requires striking a balance between ID and AD. A system that focuses solely on ID may fail to cater to the diverse interests of the entire user base, while a system that prioritizes AD may overlook the need for personalized recommendations.

By carefully balancing ID and AD, RS can achieve a more comprehensive and effective approach to recommendation, providing users with personalized and diverse suggestions while also maximizing system-level profitability.

6.2 PRIOR ART

Till recently, the prime focus of RS design algorithms was on minimizing error metrics such as root mean squared error (RMSE) and precision. However, studies suggested evaluating RS based on other measures such as diversity and unexpectedness in addition to accuracy.

Recommendations made based on accuracy as the only measure will suggest items that are in tune with the user's history. This may be monotonous for the users after a while. Also from a philosophical standpoint, we expect human beings to be well-rounded—we should have a certain curiosity and interest in all possibilities. Wearing the same type of clothes, reading similar books, or watching the same genre of movies deter our development. Hence, the argument is to introduce other diversity measures for RS evaluation.

Several attempts have been made to diversify the recommendation list while maintaining adequate relevance to the user's preference. Studies have focused on the business side alone and aimed to increase the aggregate diversity and/or suggest long tail items. This is another reason (apart from the user's well-being) to improve diversity from the business perspective. To acquire any item (however obscured), the portal has an associated cost. They need to recover this cost; they need users to buy or rent it. Thus, they need to recommend relatively unpopular/obscure items to the users.

Others studied the variation in recommendation and item ranking with parameters such as the item's net rating, its popularity, etc. to come up with several ranking strategies. They proposed a new ranking measure:

$$rank\left(i,T_R\right)=\begin{cases} rank_x\left(i\right) & \text{if } \mathrm{R}\left(u,i\right)\in\left[T_R,T_{\max}\right] \\ \alpha+rank\left(i\right) & \text{otherwise} \end{cases}$$

where T_R is the ranking threshold and $rank_x$ is the new ranking strategy applied to items ranked above a specific (chosen) ranking threshold; *rank* denotes the usual ranking strategy (highest rated ranked first). The heuristic selection of parameters and strategies in the work makes it less effective than pure optimization-based schemes. Some authors suggested recommending users to items rather than the traditional approach of recommending items to users to give a fair opportunity to all items. They also designed a probabilistic approach to address the same. Others borrowed concepts from association mining to create characterization vectors for long tail items and establish their correlation with more heavily rated items. Researchers also used an evolutionary algorithm to improve the visibility of long tail items; however, such a model cannot provide any guarantees on the optimality of the solution. Also, it is difficult to vary the model parameters to introduce varying degrees of diversity. The explicit focus of the previously mentioned works is only on suggesting a larger number of items that may not result in improved diversity for users, thereby affecting their interest in RS.

Other works have addressed the problem from the user's perspective and suggested schemes to improve individual diversity. They proposed a topic diversification-based method to increase individual diversity. The study used hierarchical classification of items combined with item-based CF to generate diverse yet reasonably accurate suggestions. The classification is done by using the item background information obtained from its taxonomic description. Some researchers introduced the idea of k-furthest neighbors. They showed that suggesting items disliked by furthest neighbors leads to higher diversity with a small reduction in accuracy. Both these methods are based on heuristic measures and neighborhood-based techniques that are not as effective as latent factor models. Only a few works have focused on pure optimization-based designs. One of them proposed a probabilistic framework that aims at generating a recommendation list such that the probability that the user will select an item from the list is maximized. The items (movies in this case) are segregated into multiple lists based on the available genre information. However, unlike our work, they proposed a greedy scheme for solving their optimization framework. Also, their model is suited for binarized ratings alone and suggests only one item at a time to the user. Others presented a more cohesive model for accuracy–diversity trade-off. They formulated a binary optimization problem combining two optimizing measures: one promoting diversity and the other accuracy.

$$\max_{y} (1-\theta)\alpha y^T D y + \theta\beta m^T y \quad s.t \quad 1^T y = p$$

where y is an indicator vector having value 1 at i^{th} position if the item i is recommended, D is a matrix defining pair wise distance (dissimilarity) between items, and m is the vector defining a user's affinity to each item. They also used a greedy scheme for solving the same owing to the quadratic nature of the problem.

6.2.1 Blind Compressed Sensing

The basic formulation for the latent factor model is given as follows,

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2 \right) \tag{1}$$

where Y is the partially observed ratings; U and V are the user and item latent factor matrices, respectively; R is a binary mask; and λ is the regularization parameter.

Conventional matrix factorization (MF) models often assume both the user and item factor matrices to be dense, meaning that each user is influenced by all latent factors and each item possesses all latent characteristics. However, this assumption may not accurately reflect reality.

In reality, a user's preferences may not encompass all possible latent factors, and an item may not exhibit all latent characteristics. For instance, a user's preferences may be limited to certain genres of movies, while a book may not belong to every genre.

To address this limitation, we propose a modification to the standard MF model by incorporating a blind compressive sensing (BCS) framework. This framework

assumes that the item factor matrix is inherently sparse, meaning that each item is associated with only a subset of latent factors.

This assumption aligns with the notion that no item possesses all possible latent characteristics. For example, a movie cannot belong to every genre. Consequently, an item's latent factor vector should primarily contain non-zero values for latent factors that accurately represent its characteristics.

In contrast, a user's preferences may be more comprehensive, encompassing a broader range of latent factors. Therefore, we assume that the user factor matrix can be dense, reflecting the possibility that a user may exhibit some affinity toward a wider range of latent characteristics.

By incorporating the BCS framework and distinguishing between the sparsity assumptions for user and item factor matrices, our modified MF model provides a more realistic representation of user preferences and item characteristics, leading to more accurate and effective recommendations.

The MF model (2) incorporating Frobenius norm constraint on U and a sparsity promoting l_1-norm constraint on V is dubbed as the BCS framework.

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_1 \tag{2}$$

Through extensive experimentation in one of our research papers, we demonstrated that the proposed model outperforms conventional MF techniques in terms of prediction accuracy. The incorporation of the BCS framework proved to be a significant factor in enhancing the model's predictive capabilities.

The superior performance of the BCS model can be attributed to its ability to capture the inherent sparsity of the item factor matrix. By assuming that each item is associated with only a subset of latent factors, the model more accurately reflects the true nature of item characteristics. This refined representation of item features leads to more precise and reliable recommendations.

Encouraged by the compelling evidence supporting the BCS model's effectiveness, we adopted this approach as the foundation for our design. By leveraging the BCS framework, we ensured that our system consistently delivers high-quality recommendations with a high degree of accuracy.

6.3 MATRIX FACTORIZATION-BASED DIVERSITY MODEL

Consider a set of explicit ratings (Y) available in the database, where $Y_{u,i}$ is the rating given by u^{th} user to i^{th} item.

We recover the complete interaction matrix as a product of two latent factor matrices: one representing the users (U) and the other the items (V). Our model is based on the blind compressive sensing-based formulation.

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_1 \tag{3}$$

Equation (3) warrants that the recovered latent factor matrices, U and V satisfy data consistency ($\left\| Y - M(UV) \right\|_F^2$ recovered in accordance with known values). Such a

model ensures that the predicted rating is high for items that are decidedly similar to ones liked by the user in the past. Thus, the traditional ranking strategy (ordered ratings in descending magnitude) generates a recommendation list characterized by high accuracy but very limited diversity.

To elevate the diversity of recommendations while maintaining their relevance to user preferences, we propose a modification to the base model (3) by introducing an additional penalty term that promotes diversity in the recommendations. This regularization term is derived from the premise that if a user exhibits similar inclinations toward all features (latent factors), it would translate into similar affinity across diverse items. In essence, the latent representation of such a user would resemble a vector whose individual elements are drawn from a uniform distribution.

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_1 + \lambda_d \sum_{u \in Users} \text{var}\ (U_u) \tag{4}$$

where $\text{var}(U_u)$ is the variance of the latent factor vector representing user u and is given by $\text{var}(U_u) = \sum_{l=1}^{F} \left(U_u(l) - m_u \right)^2$, where F is the number of latent factors considered; $U_u(l)$ is the l^{th} element of user u's latent factor vector; $m_u = \frac{1}{F} \sum_{l=1}^{F} U_u(l)$ is the mean of latent factor vector; and λ_d is the regularization parameter.

To incorporate this constraint into our base formulation (3), we introduce an additional regularization term, as shown in (Equation 4). This term minimizes the variance among the elements of a user's latent factor vector, effectively discouraging the model from assigning equal importance to all latent factors. By penalizing this tendency, we encourage the model to explore a wider range of latent factors, leading to more diverse recommendations.

This approach effectively addresses the potential monotony that can arise from relying solely on accuracy metrics. By explicitly promoting diversity, we ensure that users are exposed to a broader spectrum of recommendations, expanding their horizons and enriching their overall experience.

The incorporation of this diversity-promoting regularization term significantly enhances the effectiveness of our recommender system, providing users with a balance between relevant and diverse suggestions, ultimately leading to a more engaging and satisfying user experience.

The introduction of the variance minimization penalty term (4) serves a crucial role in preventing the predicted ratings for each user from being biased toward a particular item group. This bias can arise when the latent factor vector for a user becomes too uniform, indicating that the model is assigning similar importance to all latent factors, regardless of their relevance to that user's preferences.

A uniform latent factor vector implies that the user's predicted ratings will exhibit a similar pattern across all items, regardless of their disparate feature sets. This can lead to a recommendation list that is dominated by items from a single genre or category, limiting the user's exposure to diverse options.

To illustrate this concept, consider a movie recommendation scenario. If a user's latent factor vector is uniform, the model will predict similar ratings for movies of all genres, from action to comedy to drama. Consequently, the recommendation list for this user would likely be filled with movies from a single genre, failing to capture the breadth of their interests.

The variance minimization penalty effectively addresses this issue by discouraging the model from assigning equal importance to all latent factors. By penalizing uniform latent factor vectors, the model is encouraged to differentiate between latent factors, leading to more diverse and unbiased recommendations.

In essence, the variance minimization penalty ensures that the predicted ratings for each user are not skewed toward a specific item group, promoting a recommendation list that reflects the user's diverse preferences and interests. This approach fosters a more personalized and engaging user experience.

If considered a stand-alone, the uniform distribution constraint will ensure maximum diversity but very poor accuracy. However, in our proposed formulation, a balance between the (accuracy promoting) data consistency term $\left(\left\|Y - M\left(UV\right)\right\|_F^2\right)$ and the (diversity promoting) variance minimization regularization factor $\left(\sum_{u \in Users} \mathrm{var}\left(U_u\right)\right)$ is established using the regularization parameter λ_d. The increase in importance given to variance minimization term (via the high value of λ_d) makes diversity go higher at the cost of accuracy. The results shown in subsequent sections highlight the same.

We illustrate our proposed approach and the resulting accuracy–diversity balance with a small (toy) example shown in Figure 6.1. Let us consider an LFM-based

User 1	0.9	0.3	0.1	0.1
User 2	0.1	0.2	0.9	0.05
User 3	0.3	0.3	0.9	0.01

(a) User latent factor for standard LFM

User 1*	0.6	0.5	0.2	0.2
User 2*	0.2	0.3	0.6	0.1
User 3*	0.5	0.5	0.6	0.1

(c) User latent factor vector for proposed model

	Thril.	Drama	Rom.	Anim.
M1	0.9	0	0	0.1
M2	0.8	0	0.3	0
M3	0	0	0.7	0
M4	0	0.5	0.7	0
M 5	0.6	0.4	0	0
M 6	0	0.5	0.5	0

(b) Item latent factor vector

	User 1	User 2	User 3	User 1*	User 2*	User 3*
M1	**0.82**	0.10	0.27	**0.56**	0.19	0.46
M2	**0.75**	0.35	0.51	0.54	0.34	**0.58**
M3	0.07	0.63	**0.63**	0.14	0.42	0.42
M4	0.22	0.73	**0.78**	0.39	**0.57**	**0.67**
M 5	0.66	0.14	0.30	**0.56**	0.24	0.50
M 6	0.20	0.55	0.60	0.35	**0.45**	0.55

(d) Predicted rating values using standard model and proposed model

User n: Latent factor vectors (standard model); User n*: Latent factor vectors (proposed model)

FIGURE 6.1 Showing the Effect of Diversity-Incorporated Recommendations.

framework for movie recommendations where each movie and user are represented by a latent factor vector of length four (corresponding to four distinct genres: thriller, drama, rom-com, and animation). Given three users, one showing a higher preference for thriller and the other two for romance, Figure 6.1 (a) shows their probable latent factor vectors derived using a standard LFM-based approach (which is purely focused on accuracy). Latent factor vectors for six movies of diverse genres are given in Figure 6.1 (b). The impact of our formulation on the recovered user's latent factor representation is highlighted in Figure 6.1 (c). It shows that compared to the latent factor representation in Figure 6.1 (a), our model derives user's affinities having a flatter (more uniform) distribution than the former. Figure 6.1 (d) shows the interaction (correlation) between the users and items for both the standard formulation and our proposed design; the top two recommended movies (highest correlation) in each case are marked in bold. It can be observed that using the standard LFM recommends very similar movies to a user (like all thrillers for user 1). Our proposed design helps provide recommendations that give considerable attention to a user's preferred genre but also display diversity. For example, user 1 gets recommended a thriller as well as a movie high on drama. Similarly, user 3 is recommended a romantic thriller alongside a romantic movie. Also, our model can recommend overall five movies, across three users, compared to four suggested by the standard LFM, thereby improving aggregate diversity as well. The toy example validates our claim that our design criteria can maintain alignment with a user's past preference (high accuracy) while promoting variety in recommendation (high diversity).

$$\min_{U,V} \left\| Y - R \cdot (UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_1 + \lambda_d \sum_{u \in Users} \mathrm{var}(U_u)$$

Coming back to our formulation (4) (repeated for the reader's convenience), the variance minimization regularization term can be recast in matrix form as follows,

$$\left\| \begin{bmatrix} U_1(1) & U_1(2) & . & U_1(F) \\ U_2(1) & U_2(2) & . & . \\ . & . & . & . \\ U_m(1) & . & . & U_m(F) \end{bmatrix} - \frac{1}{F} \begin{bmatrix} U_1(1) & U_1(2) & . & U_1(F) \\ U_2(1) & U_2(2) & . & . \\ . & . & . & . \\ U_m(1) & . & . & U_m(F) \end{bmatrix} \bar{\mathbf{1}}_{F \times F} \right\|_F^2 \quad (5)$$

where m is the number of users; $\bar{\mathbf{1}}_{F \times F}$ and is a $F \times F$ matrix of all 1's.

Using (5) in (4), we can represent our proposed formulation as in (6) where $D = \mathbf{I}_{F \times F} - \frac{1}{F} \bar{\mathbf{1}}_{F \times F}$ and $\mathbf{I}_{F \times F}$ is an identity matrix of dimension $F \times F$.

$$\min_{U,V} \left\| Y - M(UV) \right\|_F^2 + \lambda_u \left\| U \right\|_F^2 + \lambda_v \left\| V \right\|_1 + \lambda_d \left\| UD \right\|_F^2 \quad (6)$$

Equation (6) defines our proposed model (**DiABlO—Di**versity **A**ccuracy **Bal**ance via **O**ptimization), which optimally combines two opposing features of a good recommender system, namely, accuracy and diversity. The predicted ratings are highly valued for items across distinct item groups, thus enabling a conventional ranking scheme to offer improved diversity to customers.

Effective recommender systems (RS) must not only provide accurate recommendations but also ensure that the recommendations are diverse, exposing users to a wide range of items beyond their immediate preferences. While existing diversity-focused RS frameworks have made significant contributions, our proposed framework offers several compelling advantages that set it apart.

A major limitation of existing diversity-focused frameworks is their reliance on secondary information, such as item category information, descriptors, or tags. This requirement often restricts their applicability, as not all datasets contain such auxiliary data.

Our proposed framework, on the other hand, requires only rating information, making it applicable to a broader range of datasets. This characteristic significantly enhances the practicality and versatility of our approach.

Existing diversity-focused frameworks often rely on a single notion of diversity, such as distinct authors for books or diverse genres for movies. This approach can lead to recommendations that, while diverse within a specific category, may still lack overall breadth.

Our proposed framework, in contrast, introduces diversity that spreads across all latent factor vectors. This means that the model considers a wider range of features, such as genre, language, director, or cast for a movie, leading to recommendations that are not only diverse within individual categories but also encompass a broader spectrum of items overall.

Extensive experimentation has shown that our proposed framework outperforms existing diversity-focused formulations. This is primarily attributed to the framework's ability to capture diversity in a more comprehensive and nuanced manner, considering a broader range of features and not relying on a single notion of diversity.

Our proposed framework for diversity-focused RS offers several significant advantages over existing approaches. By eliminating the need for secondary information, ensuring diversity across all latent factors, and demonstrating superior performance, our framework provides a more practical, effective, and versatile solution for enhancing the diversity of recommendations. This, in turn, leads to a more engaging, enriching, and personalized user experience.

6.3.1 Derivation

For our formulation (6), majorization-minimization converts it to the following form in every iteration.

$$\begin{aligned} &\min_{U,V} \|B-UV\|_F^2 + \lambda_u \|U\|_F^2 + \lambda_v \|V\|_1 + \lambda_d \|UD\|_F^2 \\ &\text{s.t. } B = (UV) + Y - R\cdot(UV) \end{aligned} \tag{7}$$

To solve (7), we employ the method of alternating direction to split it into simpler sub-problems, each minimizing over a single variable.

$$\text{P1}: \min_U \|B-UV\|_F^2 + \lambda_u \|U\|_F^2 + \lambda_d \|UD\|_F^2 \tag{8}$$

$$\text{P2}: \min_V \|B-UV\|_F^2 + \lambda_v \|V\|_1 \tag{9}$$

P1 can be rewritten as in (10), which is a linear system of equations that has a closed form solution but is usually solved using the conjugate gradient.

$$BV^T = U\left(VV^T + \lambda_u I + \lambda_d DD^T\right) \tag{10}$$

P2 can be solved via the use of the MM technique and soft thresholding wherein the solution is given by

$$V \leftarrow Soft\left(V + {}^{1}\!/\!_{\alpha}\left(U^T\left(B - UV\right)\right), \frac{\lambda_v}{2\alpha}\right) \tag{12}$$

where $Soft(t,u) = sign(t)\ \max(0,|t| - u)$; $\alpha = \max\left(eig\left(U^T U\right)\right)$

Both the sub-problems are solved alternately until convergence. The complete algorithm for our formulation is given next.

Initialize variables
Set regularization parameters;
Set maximum no. of iterations, N
while $k < N$ *or* $obj\_func(k) - obj\_func(k-1) \leq 1e-7$
// Solve for U
Solve $\left(BV^T\right) = U\left(VV^T + \lambda_u I + \lambda_d DD^T\right)$
// Solve for V

$$V \leftarrow Soft\left(V + {}^{1}\!/\!_{\alpha}\left(U^T\left(B - UV\right)\right), \frac{\lambda_v}{2\alpha}\right)$$

end while

6.4 NUCLEAR NORM-BASED DIVERSITY MODEL

Recommender systems (RS) play a crucial role in today's digital landscape, helping users navigate the vast array of products and services available online. However, traditional RS models often prioritize accuracy at the expense of diversity, leading to monotonous and repetitive recommendations. To address this limitation, we propose a novel framework that strikes a balance between diversity and accuracy, ensuring that users are exposed to a wider range of relevant items.

Unlike traditional latent factor-based models, our framework employs a convex formulation that allows for the simultaneous optimization of both accuracy and diversity. This convexity ensures that the model converges to a global optimum, providing a theoretically sound approach to balancing these competing objectives.

Additionally, our framework incorporates available item metadata, such as genre, language, director, or cast, to enrich the model's understanding of item characteristics. This enables the model to generate more diverse and relevant recommendations that align with user preferences.

Furthermore, our framework directly recovers the rating matrix using low-rank matrix completion, eliminating the need for complex latent factor inference. This approach leads to computational efficiency and improved scalability, making our framework suitable for large-scale RS applications.

$$\min_{X} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* \tag{13}$$

where R is a binary mask or subsampling operator and λ_n is the regularization parameter. The rating matrix is influenced by only a small number of latent factors; compared to the dimensions of the rating matrix (several thousands of users and items), the number of latent factors or independent variables is very low (~100). Thus, it has a (substantially) low rank structure justifying the use of the low rank matrix completion framework to recover the interaction matrix.

Equation (13) effectively recovers the interaction matrix with high precision, ensuring that the predicted ratings accurately reflect user preferences. However, it does not explicitly consider the diversity of the recommended items. Our goal is to modify this formulation to predict ratings such that the top-N recommended items have representation from diverse item sets while still aligning adequately with the users' past preferences.

Achieving maximally diversified recommendations, for a given drop in accuracy, requires that each diverse set of items has an equal probability of being recommended. In this work, we focus on the movie domain, where genre defines the diversification criterion. Ideally, the recommendation list should have an equal probability of representation or selection from each genre. While genre is particularly relevant for movies (and perhaps for books as well), our framework is generic and can incorporate other factors depending on the problem domain. For instance, in garment recommendations, one might want to consider different styles or different colors to improve diversity.

To incorporate diversity into the recommendation process, we introduce a new regularization term into the objective function in Equation (14). This term penalizes deviations from equal representation of items from different genres in the top-N recommended items for each user. By adjusting the weight parameter λ, we can control the trade-off between accuracy and diversity. A higher value of λ will lead to more diverse recommendations, while a lower value will prioritize accuracy.

Combining this diversifying criterion with the base model in Equation (13) ensures that diversity is incorporated into recommendations while maintaining adequate relevance to a user's preferences. The latter is attributed to the base accuracy-centric model.

To accommodate the diversification criterion, we include an additional regularization term in Equation (13) that promotes minimum variability among average ratings (by each user) for each genre. This term encourages the model to distribute ratings more evenly across genres, thereby increasing the likelihood of recommending diverse items.

Our formulation is as follows,

$$\min_{Z} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* + \lambda_d \sum_{u \in users} \mathrm{var}_{genre}(u) \tag{14}$$

where $\text{var}_{genre}(u) = \sum_{g \in genre} \left(X^A{}_{u,g} - m_{u,genre} \right)^2$ denotes the variance of the vector consisting of average ratings for each genre for a given user u; λ_d is the regularization parameter; $X^A{}_{u,g}$ is the average rating by user u for movies belonging to genre g; and $m_{u,genre} = \frac{1}{|genre|} \sum_{g \in genre} Z^A{}_{u,g}$ is the mean of ratings across all genres.

The nuclear norm constraint in (14) ensures a low rank structure yielding a high correlation with the previously recorded rating pattern of a user. In contrast, the diversifying criteria—uniform distribution term—in (14) promotes deviation away from set patterns by virtue of spreading the ratings uniformly across genres. The values of λ_d and λ_n determine the relative importance placed on prediction accuracy (promoted by nuclear norm constraint) and diversity (promoted by variance minimization constraint) in the recommendation list. As the model is based on a convex optimization framework, the optimality of the solution, that is, minimum loss in accuracy for a given degree of diversity is ensured.

The variance-based regularization term in (14) can be represented in matrix form as in (15) where m is the number of users; n is the number of items; $d = |genre|$ is the number of genres considered; μ_g is the number of movies belonging to genre g; and $\mathbf{1}_{d \times d}$ is a matrix of dimension $d \times d$ consisting of all 1's. Matrix G is defined such that $G_{i,j} = 1$ *if* movie i belongs to genre classification j.

$$\left\| \underbrace{\begin{bmatrix} X_{11} & X_{12} & . & X_{1n} \\ X_{21} & X_{22} & . & . \\ . & . & . & . \\ X_{m1} & . & . & X_{mn} \end{bmatrix} \begin{bmatrix} \frac{G_{11}}{\mu_1} & \frac{G_{12}}{\mu_2} & . & \frac{G_{1d}}{\mu_d} \\ \frac{G_{21}}{\mu_1} & \frac{G_{22}}{\mu_2} & . & . \\ . & . & . & . \\ \frac{G_{n1}}{\mu_1} & . & . & \frac{G_{nd}}{\mu_d} \end{bmatrix}}_{R_{u,g}\,\forall u \in users,\, g \in genre} - \underbrace{\frac{1}{d} \begin{bmatrix} X_{11} & X_{12} & . & X_{1n} \\ X_{21} & X_{22} & . & . \\ . & . & . & . \\ X_{m1} & . & . & X_{mn} \end{bmatrix} \begin{bmatrix} \frac{G_{11}}{\mu_1} & \frac{G_{12}}{\mu_2} & . & \frac{G_{1d}}{\mu_d} \\ \frac{G_{21}}{\mu_1} & \frac{G_{22}}{\mu_2} & . & . \\ . & . & . & . \\ \frac{G_{n1}}{\mu_1} & . & . & \frac{G_{nd}}{\mu_d} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{d \times d} \end{bmatrix}}_{m_{u,genre}\,\forall u \in users} \right\|_F^2 \tag{15}$$

Using (15) in (14), we can formulate our problem as follows,

$$\min_{X} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* + \lambda_d \left\| XG_\mu - XG_\mu \left[\overline{\mathbf{1}}_{d \times d} \right] \right\|_F^2 \tag{16}$$

where $G_\mu (i,j) = {G_{i,j}}/{\mu_j}$ and $\left[\overline{\mathbf{1}}_{d \times d} \right]$ is a matrix of dimension $d \times d$ with each element as ${1}/{d}$. Equation (16) can be written concisely as

$$\begin{aligned} &\min_{Z} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* + \lambda_d \left\| XF \right\|_F^2 \\ &\text{where, } F = G_\mu \left(I_{d \times d} - \left[\overline{\mathbf{1}}_{d \times d} \right] \right); \; I_{d \times d} \text{: Identity matrix} \end{aligned} \tag{17}$$

Equation (17) represents our proposed (convex) formulation. The resultant matrix Z obtained on optimizing (17) contains ratings (interaction component) recovered such that both prediction accuracy and diversity are jointly optimized.

Once X is recovered, the baseline estimates are added back to determine the (net) predicted rating values. As the predicted ratings are themselves an outcome of balancing diversity and accuracy, there is no need for a random/heuristic ranking strategy to be adopted. The ratings are ranked in order of decreasing predicted value to generate a top-N recommendation list for each user.

The incorporation of the diversity-promoting term into our recommendation model yields significant benefits in terms of both individual and aggregate diversity.

At the individual user level, our model effectively promotes diverse recommendations by ensuring that each user receives a balanced selection of items from different genres. This approach ensures that users are not confined to a narrow range of recommendations, catering to their diverse interests and preventing boredom.

For instance, a user who consistently prefers comedy movies will still receive a substantial number of comedy recommendations. However, other genres, such as action, drama, or romance, will also be represented in their recommendations, expanding their horizons and exposing them to new genres they might enjoy.

Our model also achieves high aggregate diversity by considering both accuracy and diversity for each user. This means that the model not only recommends items that are relevant to the user's preferences but also ensures that a wide range of genres are represented across all users.

In a scenario with multiple users, each with distinct preferences, our model will recommend a diverse set of movies, spanning various genres. This diversity extends beyond individual user recommendations, leading to a broader selection of movies being recommended across the entire user base.

The enhanced diversity in our recommendations translates into improved sales diversity. By exposing users to a wider range of genres, our model increases the likelihood of recommending movies that appeal to users who might not have otherwise considered them. This broader appeal leads to a more diverse sales portfolio, reducing over-reliance on a single genre and contributing to overall business growth.

As mentioned briefly before, our proposed framework can be modified to accommodate other diversifying criteria (beyond genre) across multiple application

domains (books, movies, travel plans, etc.). For example, in the case of book recommendations, we can include diversity in terms of authors, and demographic/nationalities, period of the text, or category such as fiction, drama, thriller, etc. In such a case, say for suggesting books belonging to various (time) periods, the variance term in (14) can be modified as $\text{var}_{period}(u) = \sum_{p \in period} \left(Z^{A}{}_{u,\text{p}} - m_{u,period} \right)^2$. Here, various distinct periods, early 1900 or late 2000, can be considered diverse subsets and books can be categorized as being written in these periods. In such a scenario, a user will be suggested books he/she will like across ages, instead of focusing on just new releases that anyway have much greater visibility than old texts. In addition, we may club multiple diversifying criteria also to cluster items—say, a combination of genre and tagging information for movies. In this case, we may modify our formulation to promote equal representation from each of these clusters. Here, the variance term can be modified to minimize variability in average ratings given to items in each cluster.

6.4.1 Derivation

We design an efficient algorithm for our proposed framework using the split Bregman technique. The idea behind the split Bregman technique is to enable the separation of multiple norm terms so that each can be efficiently (and separately) solved. To enable split Bregman-type separation of norm terms, we introduce a proxy variable (W) in our formulation as in (18).

$$\min_{Z,\text{W}} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* + \lambda_d \left\| WF \right\|_F^2 + \eta \left\| W - X - B \right\|_F^2 \tag{18}$$

In (18), instead of imposing a strict constraint of $W=X$, we use an augmented Lagrangian-type formulation where B is the Bregman variable (used to compensate for the difference between Z *and its proxy* W) and η is the regularization parameter. The use of the Bregman variable removes the need to impose an equality constraint between W and X from the first iteration itself. As iterations proceed, the Bregman variable is updated so that the error/difference between X and W is minimized. Updating the Bregman variable internally as part of the algorithm ensures that recovery error is minimized and convergence behavior improves.

Next, as the two variables W and X are separable, we split the problem in (18) into two simpler sub-problems each minimizing over a single variable as follows:

$$\text{P1}: \min_{Z} \left\| Y - R \cdot (X) \right\|_F^2 + \lambda_n \left\| X \right\|_* + \eta \left\| W - X - B \right\|_F^2 \tag{19}$$

$$\text{P2}: \min_{\text{W}} \lambda_d \left\| WF \right\|_F^2 + \eta \left\| W - X - B \right\|_F^2 \tag{20}$$

Considering P1, it can be recast as

$$\min_{Z} \left\| \begin{pmatrix} Y \\ \sqrt{\eta}(W - B) \end{pmatrix} - \begin{pmatrix} M \\ \sqrt{\eta} I \end{pmatrix} X \right\|_F^2 + \lambda_n \left\| X \right\|_* \tag{21}$$

Equation (21) is a nuclear norm minimization problem that can be solved using soft thresholding of singular values.

$$soft\left(singular\ value\left[X+\frac{1}{\alpha}\left(\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}^{T}\left(\begin{pmatrix}Y\\ \sqrt{\eta}(W-B)\end{pmatrix}-\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}X\right)\right)\right],\frac{\lambda_n}{2\alpha}\right) \tag{22}$$

Where $Soft(t,u)=sign(t)\ \max(0,|t|-u)$ and $\alpha \geq \max\left(eig\left(\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}^{T}\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}\right)\right)$.

P2 is a simple linear system of equations that can be solved using any gradient descent-type solver.

$$W\left(\eta I+\lambda_d FF^{T}\right)=\eta(X+B) \tag{23}$$

The two sub-problems are alternately solved with consecutive iterations of both interlaced with the updating of the Bregman variable as in (22)

$$B \leftarrow B+X-W \tag{24}$$

The iterations continue until convergence, that is, the maximum number of iterations reached or the reduction in objective function value drops below the threshold. The complete algorithm is summarized next.

Initialize variables

Set parameters : $max\_iter, \lambda_n, \lambda_d, \eta$

while $k < max\_iter$ *or* $obj\_func(k) - obj\_func(k-1) \leq 1e-7$

// Solve for Z

$$X \leftarrow Soft\left(sing\_val\left[Z+\frac{1}{\alpha}\left(\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}^{T}\left(\begin{pmatrix}Y\\ \sqrt{\eta}(W-B)\end{pmatrix}-\begin{pmatrix}R\\ \sqrt{\eta}I\end{pmatrix}Z\right)\right)\right],\frac{\lambda_n}{2\alpha}\right)$$

// Solve for W

Solve $W\left(\eta I+\lambda_d FF^{T}\right)=\eta(X+B)$

// Update Bregman variable

$B = B + X - W$

end while

BIBLIOGRAPHY

Gogna, A. and Majumdar, A., 2017a. DiABlO: Optimization based design for improving diversity in recommender system. *Information Sciences*, 378, pp. 59–74.

Gogna, A. and Majumdar, A., 2017b. Balancing accuracy and diversity in recommendations using matrix completion framework. *Knowledge-Based Systems*, 125, pp. 83–95.

7 Deep Latent Factor Models

7.1 INTRODUCTION

The realm of applied computer science has witnessed a transformative surge in the utilization of deep learning over the past five years. Its profound impact has permeated diverse domains, including speech processing, computer vision, natural language processing (NLP), and information retrieval. Deep learning has demonstrated remarkable effectiveness in enhancing search accuracy and relevance, propelling its adoption in information retrieval systems.

Inspired by the remarkable success of deep learning, we propose a novel deep latent factor model, specifically designed for collaborative filtering problems. Collaborative filtering, a cornerstone of recommender systems, aims to predict user preferences and suggest relevant items based on their past interactions and similarities with other users. While traditional latent factor models have been widely employed in collaborative filtering, they often struggle to capture complex nonlinear relationships and higher-order interactions within data. Deep latent factor models address these limitations by incorporating deep neural networks into the latent factor framework.

Traditional latent factor models, such as matrix factorization, assume that user preferences and item characteristics can be represented by a linear combination of latent factors. This assumption, however, often falls short of capturing the intricate and nonlinear relationships inherent in user behavior and item attributes. Deep latent factor models overcome this limitation by employing deep neural networks to learn latent representations that can effectively capture complex nonlinear patterns and higher-order interactions within the data.

The proposed deep latent factor model utilizes the power of deep networks to extract meaningful latent representations from user–item interactions. By employing multiple layers of interconnected neurons, the model effectively captures nonlinear relationships and interactions among users and items. This enables the model to extract deeper insights from user behavior and preferences, leading to more accurate and personalized recommendations.

7.1.1 Advantages of Deep Latent Factor Models

Deep latent factor models offer several advantages over traditional latent factor models:

1. **Handling High-Dimensional and Sparse Data:** Deep latent factor models can effectively handle high-dimensional and sparse data, which is a common characteristic of user–item matrices in collaborative filtering.

DOI: 10.1201/9781003511267-7

2. **Capturing Complex Nonlinear Relationships:** The model's deep architecture allows it to capture complex nonlinear relationships and interactions within the data, providing a more nuanced understanding of user preferences.
3. **Learning Latent Representations:** The model's ability to learn latent representations facilitates the identification of underlying patterns and relationships that may not be readily apparent from explicit user–item interactions.

The proposed deep latent factor model holds significant potential for enhancing the performance of recommender systems. Its ability to capture complex nonlinear relationships and extract deeper insights from user behavior can lead to more accurate and personalized recommendations, ultimately improving user satisfaction and engagement.

As deep learning continues to evolve, further advancements in deep latent factor models can be anticipated, paving the way for even more sophisticated and effective recommender systems. Future research directions include the following:

1. **Incorporating Additional Data Sources:** Integrating additional data sources, such as user demographics, item descriptions, and social network information, can further enrich the model's understanding of user preferences and item characteristics.
2. **Addressing Cold-Start Problems:** Developing effective strategies to address cold-start problems, where limited or no information is available for new users or items, is crucial for improving the model's performance.
3. **Real-Time Recommendations:** Enabling real-time recommendations, where the model can adapt to user preferences and item availability in real-time, is essential for enhancing user experience and engagement.

The advent of deep latent factor models represents a significant leap forward in the development of recommender systems. By harnessing the power of deep learning, these models offer the potential to provide more accurate, personalized, and timely recommendations, ultimately revolutionizing the way we interact with information and technology.

Numerous deep learning-based collaborative filtering methods such as the one used by Dong et al. (2017) employ a rather simplistic approach, utilizing only user and item IDs as inputs to the deep neural network. This strategy, while seemingly straightforward, faces significant drawbacks due to the inherent limitations of these IDs. User and item IDs, by themselves, do not convey any meaningful information about the user's preferences or the item's characteristics. They merely serve as labels for identifying users and items within the dataset.

Predicting ratings solely based on these IDs is akin to attempting to decipher a person's emotions by their phone number or a product's quality by its barcode. Such a direct mapping is unlikely to yield accurate or insightful results. User and item IDs are merely placeholders, lacking any inherent connection to the underlying preferences or characteristics they represent.

The arbitrary nature of using IDs as inputs stems from the fact that these identifiers do not capture any contextual or relational information. They provide no insight into the user's past interactions, their preferences for specific genres or categories, or the item's attributes, features, or popularity. Without this contextual information, the deep neural network is essentially operating in a vacuum, attempting to extract meaningful patterns from meaningless data.

As a result, relying solely on user and item IDs as inputs leads to models that are prone to overfitting and generalization errors. They fail to capture the true essence of user preferences and item characteristics, resulting in inaccurate predictions and limited understanding of user behavior.

To overcome these limitations and develop more effective deep learning-based collaborative filtering approaches, researchers need to incorporate additional sources of information that provide meaningful context and relationships. This may include user demographics, item descriptions, social network connections, and past interaction history. By enriching the input data with such contextual information, deep neural networks can gain a deeper understanding of user preferences and item characteristics, leading to more accurate and personalized recommendations.

While the approach proposed by He et al. (2017) represents a step forward compared to using solely user and item IDs, it still faces certain limitations due to the inherent challenges of using ratings as both inputs and outputs.

Employing ratings for both inputs (user and item) and as the output (class) introduces a certain level of circularity in the model. The model is essentially attempting to predict ratings based on ratings, which can lead to overfitting and limited generalization. The model may prioritize memorizing the existing ratings rather than learning meaningful patterns from the data.

Furthermore, using ratings as both inputs and outputs can obscure the underlying reasons behind user preferences and item characteristics. The model may focus on predicting the exact rating value rather than understanding the factors that influence those ratings. This can hinder the model's ability to provide insights into user behavior and identify trends or patterns.

Moreover, the justification for using ratings as both inputs and outputs is unclear in both Dong et al. (2017) and He et al. (2017). Without a clear theoretical foundation, the effectiveness of this approach remains questionable.

To address these limitations, researchers need to explore alternative deep learning architectures and input representations that avoid the circularity of using ratings as both inputs and outputs. Incorporating additional sources of information, such as user demographics, item descriptions, and social network connections, can enrich the model's understanding of user preferences and item characteristics, leading to more insightful and generalizable recommendations.

7.2 BRIEF INTRODUCTION TO REPRESENTATION LEARNING

Restricted Boltzmann Machines (RBMs) have gained popularity in recent years as a foundational component of deep belief networks (DBNs), a type of unsupervised generative model. However, RBMs were initially introduced in the context of collaborative filtering, a technique for predicting user preferences based on their past

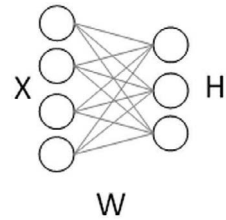

FIGURE 7.1 Restricted Boltzmann Machine.

behavior. Despite their initial promise, RBMs have not gained widespread adoption in collaborative filtering due to several inherent limitations.

One of the primary limitations of RBMs for collaborative filtering stems from the constraint that their input data must be binary, with values of either 0 or 1. This restriction poses a significant challenge in collaborative filtering, as user ratings typically lie on a continuous scale, such as one to five stars or 0 to 100%.

To address this limitation, Gaussian-Bernoulli RBMs (GBRBMs) were introduced, allowing for continuous-valued inputs within the range of 0 to 1. However, GBRBMs still face challenges in representing the diversity and granularity of user ratings.

Another limitation of RBMs in collaborative filtering lies in their training algorithm, contrastive divergence. Contrastive divergence is an approximation of the true gradient descent algorithm, but it is computationally efficient and less susceptible to local optima. However, its inherent approximations make it difficult to perform mathematical manipulations or analyze the model's behavior in detail.

Despite these limitations, RBMs offer certain advantages in collaborative filtering. The network weights can be interpreted as users' latent factors, representing their underlying preferences and characteristics. Similarly, the activation states of the hidden units can be interpreted as items' latent factors, capturing their essential attributes.

Despite their potential, the limitations of RBMs have hindered their widespread adoption in collaborative filtering. Alternative approaches, such as matrix factorization and latent Dirichlet allocation (LDA), have gained more traction due to their ability to handle continuous-valued inputs and their more well-defined training algorithms.

Recent advancements in deep learning, particularly the development of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), have opened new avenues for collaborative filtering. These models have demonstrated superior performance in capturing complex relationships between users and items, leading to more accurate and personalized recommendations.

While RBMs have not revolutionized collaborative filtering as initially envisioned, their contributions to the field of deep learning cannot be overlooked. Their ability to model latent factors and their efficient training algorithm have laid the groundwork for more sophisticated deep learning architectures that are now shaping the future of recommender systems.

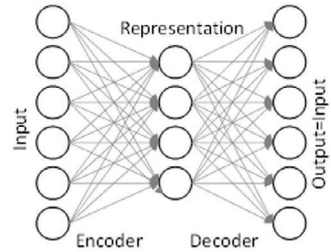

FIGURE 7.2 Autoencoder.

Autoencoder-based collaborative filtering techniques have emerged as a promising approach in recent years. These techniques leverage the power of neural networks to learn latent representations of user preferences and item characteristics, enabling them to make accurate recommendations.

Autoencoders are a type of neural network that aims to reconstruct its input data (Figure 7.2). They consist of two main components: an encoder and a decoder. The encoder takes the input data and compresses it into a lower-dimensional representation, known as the latent code. The decoder then attempts to reconstruct the original input data from this latent code.

In the context of collaborative filtering, the autoencoder's input and output are user–item rating matrices, which may contain missing values. When there are missing entries in the data, the corresponding network weights are not updated during training. This ensures that the model learns to reconstruct the missing ratings while preserving the observed ones.

Deep autoencoders extend the concept of shallow autoencoders by stacking multiple layers of encoders and decoders. The idea is that deeper autoencoders can capture more complex and nuanced relationships within the data, leading to improved recommendation performance.

However, research has shown that deep autoencoders don't always provide a significant advantage over shallow autoencoders for collaborative filtering tasks. This suggests that the complexity of the model may not always be directly proportional to its performance.

Autoencoder-based collaborative filtering techniques offer several advantages:

1. **Handling Missing Values:** Autoencoders are naturally suited to handle missing values in the user–item rating matrix, making them well-suited for real-world scenarios where data sparsity is common.
2. **Learning Latent Representations:** Autoencoders can effectively learn latent representations of user preferences and item characteristics, capturing the underlying patterns and relationships within the data.
3. **Scalability:** Autoencoder-based models can be efficiently trained and scaled to large datasets, making them applicable to real-world recommender systems.

Despite their advantages, autoencoder-based collaborative filtering techniques face certain limitations:

1. **Predictive Performance:** Autoencoders are primarily focused on reconstructing the input data, and their predictive performance may not always be optimal for recommendation tasks.
2. **Interpretation of Latent Code:** The latent code learned by autoencoders can be difficult to interpret, making it challenging to understand the underlying factors that influence user preferences and item characteristics.
3. **Sensitivity to Initialization:** Autoencoders can be sensitive to their initialization parameters, and finding the optimal initialization can be computationally demanding.

Overall, autoencoder-based collaborative filtering techniques offer a promising approach for recommendation systems. Their ability to handle missing values, learn latent representations, and scale to large datasets makes them a valuable tool in the field of recommender systems.

In the standard latent factor model, users are represented by a user latent factor matrix, denoted by U, and items are represented by an item latent factor matrix, denoted by V. The ratings matrix, denoted by X, is expressed as the product of U and V:

$$X = UV \tag{1}$$

This equation represents the core idea of latent factor models: that user preferences and item characteristics can be captured by a small number of latent factors. The user latent factor matrix U contains a vector of latent factors for each user, representing their underlying preferences and interests. Similarly, the item latent factor matrix V contains a vector of latent factors for each item, capturing their essential attributes and features.

By multiplying U and V, the latent factor model can reconstruct the ratings matrix R. This reconstruction process allows the model to infer missing ratings and make recommendations based on the similarity between users and items in the latent factor space.

The standard latent factor model has been widely used in collaborative filtering due to its simplicity and effectiveness. However, it also faces certain limitations, such as its inability to capture complex nonlinear relationships and interactions within the

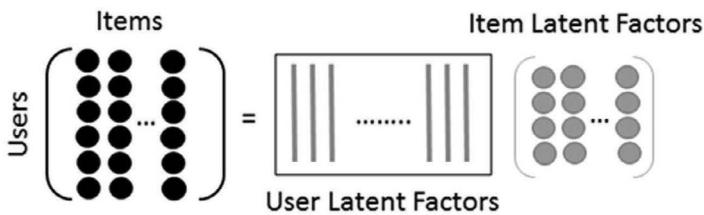

FIGURE 7.3 Latent Factor Model.

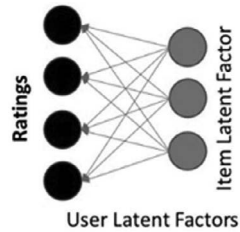

FIGURE 7.4 Neural Network Interpretation.

data. More advanced latent factor models, such as deep latent factor models, have been developed to address these limitations.

An alternative perspective on the latent factor model is to view it as a neural network. In this interpretation, the user latent factors are not represented as vectors but rather as connections between the item latent factors and the ratings. This is illustrated in Figure 7.4.

In this neural network representation, each item latent factor is connected to each rating through a corresponding weight, which represents the strength of the connection between the item's attributes and the user's preference for that item. The ratings are then calculated by summing the weighted contributions of all item latent factors.

This neural network interpretation of the latent factor model provides a more intuitive understanding of the model's behavior. It shows that the model is essentially learning a set of weights that represents the relationships between item attributes and user preferences.

The inputs to this neural network are all the users' ratings on the j^{th} item denoted by x_j (black nodes) in Figure 7.4. Their corresponding representation is the latent factors for the corresponding item v_j (gray nodes). Following (1), the relationship between both is expressed as $x_j = Uv_j$ —this is same as the equation of a neural network, albeit in an opposite direction from the representation to the input. Each row u_i of U is the user latent factor for the i^{th} user.

The neural network interpretation of collaborative filtering opens new possibilities for extending the latent factor model to deeper architectures. In deep learning, the latent representation learned by one layer serves as the input to the next layer. This principle can be applied to the latent factor model by stacking multiple layers of latent factors, each layer capturing increasingly abstract and nuanced representations of user preferences and item characteristics.

This approach, known as the deep latent factor model, allows the model to capture complex nonlinear relationships and interactions within the data, overcoming a key limitation of traditional latent factor models. By learning deeper latent representations, the deep latent factor model can make more accurate and personalized recommendations.

7.3 DEEP LATENT FACTOR MODEL

Collaborative filtering, a cornerstone of recommender systems, aims to predict user preferences and suggest relevant items based on their past interactions and similarities

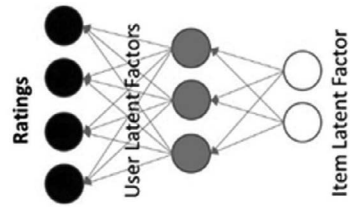

FIGURE 7.5 Deep Latent Factor Model.

with other users. Traditional latent factor models have been widely employed in collaborative filtering, but they often struggle to capture complex nonlinear relationships and higher-order interactions within data. Deep latent factor models address these limitations by incorporating deep neural networks into the latent factor framework.

The deep latent factor model, as depicted in Figure 7.5, extends the traditional latent factor model by stacking multiple layers of latent factors. This layered architecture allows the model to capture increasingly abstract and nuanced representations of user preferences and item characteristics. Unlike traditional latent factor models, which often rely on superficial factors like author, genre, or director, deep latent factor models can delve into deeper personality traits and underlying patterns.

As the model progresses through deeper layers, it distills information from the input data, extracting more abstract and meaningful representations. Consider a book recommendation system. A shallow latent factor model might simply consider factors like author, genre, or publisher. However, a deep latent factor model could capture more nuanced aspects, such as the book's tone, themes, or emotional impact.

Deeper latent representations offer several advantages for collaborative filtering:

1. **Enhanced Interpretability:** By capturing higher-order relationships and underlying patterns, deep latent factor models provide a more comprehensive understanding of user preferences and item characteristics.
2. **Improved Generalizability:** By learning abstract representations, the model can generalize better to unseen data, making recommendations more robust and accurate.
3. **Reduced Dependence on Superficial Features:** Deeper latent factor models can move beyond superficial features and capture deeper personality traits and underlying patterns, leading to more personalized recommendations.

While deep latent factor models hold immense promise, they also face challenges. One challenge is the increased complexity of the model, which can make it computationally expensive to train and interpret. Additionally, the model's reliance on deep neural networks introduces the risk of overfitting, where the model learns the training data too well and fails to generalize to new data.

Deep latent factor models represent a significant step forward in collaborative filtering, offering the potential for more accurate, personalized, and interpretable recommendations. As research continues to address the challenges of deep learning,

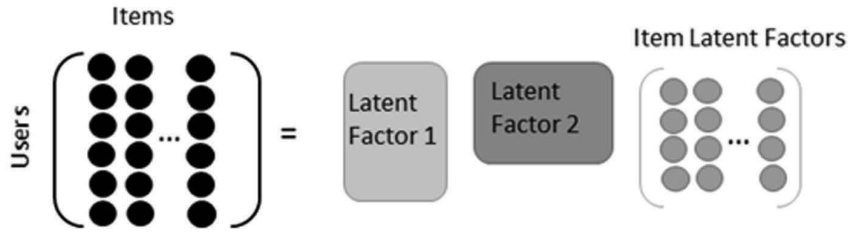

FIGURE 7.6 Factorization-Type Interpretation.

deep latent factor models hold the promise of revolutionizing the way we interact with information and technology.

The neural network interpretation of the deep latent factor model provides a more intuitive understanding of the model's behavior and its connection to other deep learning methods. However, for mathematical analysis and comparisons with traditional latent factor models, expressing the model in matrix factorization form is more convenient and common.

In the matrix factorization form, the deep latent factor model can be represented as a product of multiple matrices, each representing a layer of latent factors. This representation allows for efficient computation and facilitates comparisons with other matrix factorization-based latent factor models.

While the neural network interpretation provides a more intuitive understanding of the model's structure and behavior, the matrix factorization form is essential for mathematical analysis and comparisons with traditional latent factor models. Both representations are valid and complementary, offering different perspectives on the same underlying model.

7.3.1 Mathematical Formulation

In the standard latent factor model, the ratings matrix is expressed as a product of user and item latent factors.

$$X = UV \tag{6}$$

Similar to the rectified linear unit (ReLU) activation function in neural networks, non-negativity constraints are commonly applied to latent factors in deep latent factor models. This constraint restricts the values of latent factors to be non-negative, offering several advantages:

1. **Interpretability:** Non-negativity enhances the interpretability of latent factors. Since negative values lack clear meaning in the context of user preferences or item characteristics, restricting latent factors to non-negative values allows for a more intuitive understanding of their significance.

2. **Sparsity:** Enforcing non-negativity promotes sparsity in the latent factor matrices, meaning a majority of latent factor values are close to zero. This sparsity benefits computational efficiency and helps prevent overfitting.
3. **Biological Relevance:** In certain applications, such as biological data recommender systems, non-negativity constraints carry biological relevance. For instance, in gene expression analysis, non-negativity reflects the fact that gene expression levels cannot be negative.

In our deeper model, we will have multiple levels of user latent factors and one final level of item latent factors. In the case of two layers, this can be expressed as

$$X = U_1 U_2 V \tag{7}$$

Here, U_1 and U_2 are the two levels of item latent factors and V consists of the deep user latent factors. In each level, a ReLU-type activation was imposed by non-negativity constraints. The formulation (7) can be easily extended to N layers:

$$X = U_1 U_2 ... U_N V \tag{8}$$

Techniques for solving deep matrix factorization have already been developed when the data is fully observed. Unfortunately, in collaborative filtering, the ratings matrix is only partially observed, that is,

$$Y = R \cdot X \tag{9}$$

where X is the full ratings matrix, R the binary matrix of 1's and 0's, and Y the acquired ratings matrix.

We incorporate the deep matrix factorization framework (7) into the partially observed model (8) to yield our deep latent factor model.

$$Y = R \cdot X = R \cdot \left(U_1 U_2 ... U_N V \right) \tag{10}$$

7.3.2 Solution

We will show the algorithm for three layers; it will be generic enough for more or fewer layers. The formulation for three layers is

$$\begin{aligned} &\min_{U_1, U_2, U_3, V} \tfrac{1}{2} \left\| Y - R \cdot \left(U_1 U_2 U_3 V \right) \right\|_F^2 \text{ such that} \\ &U_1 U_2 U_3 V \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0 \end{aligned} \tag{11}$$

This is equivalent to the following,

$$\min_{U_1, U_2, U_3, V, X} \tfrac{1}{2} \left\| Y - R \cdot \left(U_1 U_2 U_3 V \right) \right\|_F^2 \tag{12}$$

such that $U_1 U_2 U_3 V = X$ and $X \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0$.

We propose a projected gradient method, with an inner loop based on alternating projection to solve (12). The general form of our algorithm is as follows:

$$\begin{aligned}
&\text{Initialize: } X^0, U_1^0, U_2^0, U_3^0, V^0 \\
&\text{For } k = 1,2,... \\
&1) \quad \bar{X}^k = X^k - \gamma\left(R^T \cdot \left(R \cdot X^k - Y\right)\right) \\
&2) \quad \left(X^{k+1}, U_1^{k+1}, U_2^{k+1}, U_3^{k+1}, V^{k+1}\right) \text{ solution of} \\
&\min_{X,U_1,U_2,U_3,V} \left\|\bar{X}^k - X\right\|_F^2 \text{ s.t. } U_1 U_2 U_3 V = X \\
&\text{and } X \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0
\end{aligned}$$

The second sub-problem corresponds to projecting X^k on the constrained domains $U_1U_2U_3V=X$ and $X \geq 0$, $U_1 \geq 0$, $U_2 \geq 0$, $U_3 \geq 0$, $V \geq 0$. This projection problem is complex, as the equality constraint defines a non-convex set. Moreover, it has no closed form. We thus propose to perform alternating projections on the constraints, to derive an approximate solution to this sub-problem. Each variable will be treated sequentially, in a Gauss–Seidel fashion, and then projected on its associated constraint set. For variables U_i and V, we propose to approximate the projection of the intersection of positive and equality constraint, by the composition of the respective projectors, in order to avoid inner iterations.

Step 2 will read as follows:

$$\begin{aligned}
X^{k+1} &= P_+\left(\bar{X}^k\right) \\
U_1^{k+1} &= P_+\left(P_{\left\{U_1 U_2^k U_3^k V^k = X^{k+1}\right\}}\left(U_1^k\right)\right) \\
U_2^{k+1} &= P_+\left(P_{\left\{U_1^{k+1} U_2 U_3^k V^k = X^{k+1}\right\}}\left(U_2^k\right)\right) \\
U_3^{k+1} &= P_+\left(P_{\left\{U_1^{k+1} U_2^{k+1} U_3 V^k = X^{k+1}\right\}}\left(U_3^k\right)\right) \\
V^{k+1} &= P_+\left(P_{\left\{U_1^{k+1} U_2^{k+1} U_3^{k+1} V = X^{k+1}\right\}}\left(V^k\right)\right)
\end{aligned}$$

Here, P_+ denotes the projector onto the positive orthant, that is, capping the negative entries of an input matrix to 0. We can then apply the general property that the projection of a matrix U on a linear constraint $X=AUB$ is given by

$$\hat{U} = U - A^\dagger\left(AUB - X\right)B^\dagger \tag{13}$$

with † the pseudo-inverse operation, so that step 2 finally reads:

$$\begin{aligned}
X^{k+1} &= P_+\left(\bar{X}^k\right) \\
U_1^{k+1} &= P_+\left(U_1^k - \left(U_1^k U_2^k U_3^k V^k - X^{k+1}\right)\left(U_2^k U_3^k V^k\right)^\dagger\right)
\end{aligned}$$

$$U_2^{k+1} = P_+\left(U_2^k - \left(U_1^{k+1}\right)^\dagger \left(U_1^{k+1}U_2^kU_3^kV^k - X^{k+1}\right)\left(U_3^kV^k\right)^\dagger\right)$$

$$U_3^{k+1} = P_+\left(U_3^k - \left(U_1^{k+1}U_2^{k+1}\right)^\dagger \left(U_1^{k+1}U_2^{k+1}U_3^kV^k - X^{k+1}\right)\left(V^k\right)^\dagger\right)$$

$$V^{k+1} = P_+\left(V^k - \left(U_1^{k+1}U_2^{k+1}U_3^{k+1}\right)^\dagger \left(U_1^{k+1}U_2^{k+1}U_3^{k+1}V^k - X^{k+1}\right)\right)$$

Steps 1 and 2 can be understood as a gradient projection method. The complicated form of the constraint requires the use of an inner step for approximating the projection step. The convergence of the whole scheme cannot be established easily because of the presence of the coupling in the equality constraint $U_1U_2U_3V = X$. However, in our experimental results, we will show that the algorithm converges empirically. Here, we have shown the algorithm for three layers. This can be generalized to any number of layers.

7.4 GRAPHICAL DEEP LATENT FACTOR MODEL

In the realm of collaborative filtering, deep matrix factorization (DMF) has emerged as a powerful technique for capturing complex relationships and patterns within user–item interactions. DMF excels in extracting latent factors from sparse and noisy data, leading to improved recommendation performance compared to traditional shallow matrix factorization models.

However, when additional auxiliary information, such as user social connections or item similarity networks, is available, graph regularization offers an alternative approach to incorporating this information into the matrix factorization framework. Graph regularization utilizes the graph structure to constrain the latent factors, ensuring that users with similar connections or items with similar attributes are represented by similar latent factors.

While both DMF and graph regularization have demonstrated their effectiveness in collaborative filtering, recent research has shown that combining the two approaches can lead to even further improvements in recommendation accuracy. This synergy arises from the complementary strengths of each method:

- DMF excels at capturing latent factors from user–item interactions, even when the data is sparse and noisy. It effectively learns abstract representations of user preferences and item characteristics, enabling it to make accurate predictions for unseen items.
- Graph regularization, on the other hand, leverages the structure of auxiliary information to refine the latent factors. By ensuring that users with similar connections or items with similar attributes are represented similarly, graph regularization guides the model toward more meaningful and interpretable latent factors.

Combining DMF and graph regularization allows the model to capitalize on the strengths of both approaches. DMF provides the foundation for learning latent factors from user–item interactions, while graph regularization refines these latent

factors by incorporating the structure of auxiliary information. This combination leads to more accurate and personalized recommendations.

In the context of deep latent factor models, graph regularization can be applied to regularize the user (U) and item (V) latent factors at each layer of the model. This involves incorporating graph Laplacians, which represent the connectivity structure of the corresponding graphs, into the loss function of the deep matrix factorization model.

The first task when applying graph regularization to deep latent factor models is identifying the user and item latent factors at each layer of the model. These latent factors represent the underlying preferences and characteristics of users and items, respectively.

In traditional matrix factorization models, the latent factors are directly represented by the matrices U and V. However, in deep latent factor models, the latent factors are more abstract and are embedded within the neural network architecture. Identifying these latent factors requires understanding the structure of the neural network and the flow of information within the model.

Once the latent factors have been identified, graph regularization can be applied by incorporating the corresponding graph Laplacians into the loss function. This regularization term encourages the latent factors to be consistent with the structure of the auxiliary graphs, leading to more accurate and personalized recommendations.

A four-layer deep factorization model can be expressed as

$$X_{M\times N} = \underbrace{U_{M\times p}}_{user} F_{p\times q} G_{q\times r} \underbrace{V_{r\times N}}_{item}$$

Here, X is the rating's matrix for M users and N products. It is factored into four latent factors—U, F, G, and V. Looking at the dimensions of each, one can easily identify that U corresponds to the user latent factors and V the item latent factors; F and G are deep latent factors.

Once the user and item latent factors are identified, the cost function to be solved can be expressed as

$$\min_{U,F,G,V} \left\| Y - R(UFGV) \right\|_F^2 + \lambda \left(Tr\left(U^T L_U U\right) + Tr\left(V L_V V^T\right) \right) \text{s.t. } F \geq 0,\ G \geq 0 \text{ and } V \geq 0$$

The standard approach to graph-regularized deep matrix factorization (DMF) involves incorporating only one type of similarity for users and items. However, in reality, there can be multiple types of similarities arising from different definitions, such as cosine, correlation, or Jaccard similarity, as well as different sources, such as user demographics, social network information, or item descriptions.

To effectively capture these diverse similarities, it is necessary to extend the DMF framework to handle multiple graphs. This involves defining separate graph Laplacians for each type of similarity and incorporating them into the loss function of the DMF model.

Utilizing multiple graphs in graph-regularized DMF offers several benefits:

1. **Enhanced Modeling Flexibility:** By incorporating multiple graphs, the model can capture a wider range of similarities, leading to a more comprehensive understanding of user preferences and item characteristics.

2. **Improved Recommendation Accuracy:** Incorporating multiple types of similarities can refine the latent factors, leading to more accurate and personalized recommendations.
3. **Interpretability:** Multi-graph regularization provides a more granular understanding of the factors influencing user preferences and item characteristics, enhancing the interpretability of the model.

While multi-graph regularization offers promising benefits, it also presents certain challenges:

1. **Parameter Tuning:** The model involves a larger number of parameters, requiring careful tuning to balance the contributions of different graphs.
2. **Computational Complexity:** Incorporating multiple graphs increases the computational complexity of the model, requiring efficient optimization algorithms.
3. **Data Availability:** Acquiring and processing multiple types of similarity data can be challenging, particularly for social network information or item descriptions.

Despite these challenges, multi-graph regularization offers a promising approach for enhancing the performance and interpretability of deep matrix factorization models in collaborative filtering and other domains.

Our final formulation of deep matrix factorization incorporating multiple graphs is as follows—

$$\begin{aligned} &\min_{U,F,G,V} \left\|Y - R(UFGV)\right\|_F^2 + \lambda\left(\sum_i Tr\left(U^T L_{U_i} U\right) + \sum_j Tr\left(VL_{V_j} V^T\right)\right) \\ &\text{s.t. } F \geq 0,\ V \geq 0 \text{ and } V \geq 0 \end{aligned} \tag{14}$$

Ideally, one needs different regularization parameters for each penalty term, but this would increase the tuning time drastically. To keep it simple, we have kept only one regularization parameter λ.

To solve (14), we first decouple it using the majorization-minimization (MM) framework. In an iterative fashion, we solve the following:

$$\begin{aligned} &\min_{U,F,G,V} \left\|B - UFGV\right\|_F^2 + \lambda\left(\sum_i Tr\left(U^T L_{U_i} U\right) + \sum_j Tr\left(VL_{V_j} V^T\right)\right) \\ &\text{s.t. } F \geq 0,\ G \geq 0 \text{ and } V \geq 0 \end{aligned} \tag{15}$$

where $B = \left(UFGV\right)_{k-1} + \frac{1}{\alpha} R^T\left(Y - R\left(UFGV\right)_{k-1}\right)$; α is the highest eigenvalue of $R^T R$. Here, "k" denotes the iteration.

The solution for (15) proceeds by updating the variables in an alternating fashion. Here, the superscript "l" indicates the inner iterations (within k).

$$U \leftarrow \min_{U} \left\| B - UF^{(l-1)}G^{(l-1)}V^{(l-1)} \right\|_F^2 + \lambda \sum_i Tr\left(U^T L_{U_i} U\right)$$

$$F \leftarrow \min_{F} \left\| B - U^{(l)}FG^{(l-1)}V^{(l-1)} \right\|_F^2$$

$$G \leftarrow \min_{G} \left\| B - U^{(l)}F^{(l)}GV^{(l-1)} \right\|_F^2$$

$$V \leftarrow \min_{V} \left\| B - U^{(l)}F^{(l)}G^{(l)}V \right\|_F^2 + \lambda \sum_j Tr\left(VL_{V_j}V^T\right)$$

Here, we have abused the notations slightly. We have not shown the positivity constraints explicitly. These will be incorporated during the updates of individual variables.

The solutions to the updates of F and G are fairly straightforward. It involves applying the right and the left Moore–Penrose pseudoinverse appropriately.

$$F = \left(U^{(l)}\right)^{\dagger} B\left(G^{(l-1)}V^{(l-1)}\right)^{\dagger} \tag{16}$$

$$G = \left(U^{(l)}F^{(l)}\right)^{\dagger} B\left(V^{(l-1)}\right)^{\dagger} \tag{17}$$

Here, $(\cdot)^{\dagger}$ denotes pseudoinverse. Once the variables F and G are updated given the formulae, non-negativity constraint is applied by forcing the negative entries in them to zero. This is not an exact solution but has been used profusely in the past for non-negative matrix factorization and deep matrix factorization. The exact formula would require fast iterative soft thresholding-type algorithm or another general dual forward-backward splitting. This would make the updates for F and G iterative and time-consuming. This is the reason we follow the approximate solution.

For solving U, we take the derivative of the cost function and equate it to 0. This leads to

$$\lambda L_U U + UF^{(l-1)}G^{(l-1)}V^{(l-1)}\left(F^{(l-1)}G^{(l-1)}V^{(l-1)}\right)^T = B\left(F^{(l-1)}G^{(l-1)}V^{(l-1)}\right)^T \tag{18}$$

where $L_U = \sum_i L_{U_i}$.

Note that (18) is a Sylvester's equation of the form $A_1X + XA_2 = C$ with $\lambda L_U = A_1$, $A_2 = F^{(l-1)}G^{(l-1)}V^{(l-1)}\left(F^{(l-1)}G^{(l-1)}V^{(l-1)}\right)^T$, and $C = B\left(F^{(l-1)}G^{(l-1)}V^{(l-1)}\right)^T$. There are efficient solvers for the same.

The solution for V can be similarly obtained. Since there is a non-negativity constraint on V, the negative values in V after solving Sylvester's equation are forced to 0.

7.5 DIVERSITY IN DEEP LATENT FACTOR MODEL

In the realm of recommendation systems, diversity plays a pivotal role in enhancing user satisfaction and engagement. By introducing a variety of items into the recommendation list, diversity fosters serendipitous discoveries, broadens users' horizons, and prevents the system from becoming overly repetitive or predictable. However, achieving a balance between diversity and relevance remains a significant challenge in recommendation system design.

A common approach to incorporating diversity into recommendation systems involves a two-stage process. The first stage employs traditional collaborative filtering (CF) techniques to predict item ratings. This initial stage establishes a baseline understanding of the user's long-term static preferences.

The second stage introduces diversity by employing a ranking scheme that prioritizes items from a broader spectrum of interests and categories. Various strategies can be employed to achieve diversity, such as clustering, topic diversification, item popularity measures, and greedy optimization-based design.

Some approaches leverage user interests and social influences to generate the recommendation list. These methods often rely on explicit or implicit signals of user interests, such as their past interactions, search history, or social connections. The recommendation list is then re-ranked based on item popularity and diversity metrics.

Another approach, known as Exploitation-Exploration Diversification (XPLODIV), utilizes a trade-off between exploiting the user profile and exploring novel items. This technique can be applied to traditional CF models, effectively balancing diversity with relevance.

A binary optimization formulation has also been proposed to achieve a balance between diversity and matching quality. This method introduces an input control parameter that allows for fine-tuning the trade-off between the two objectives.

Researchers have also explored the concept of serendipity, the quality of making pleasant, unexpected discoveries, in relation to diversity. A graph-based re-ranking algorithm has been proposed to enhance diversity by considering connections between items.

While two-stage approaches offer a straightforward way to incorporate diversity, they often suffer from a lack of coordination between the stages. This can lead to suboptimal recommendations and the need for heuristic selection of ranking thresholds and strategies. Additionally, secondary information like item metadata may be required to provide recommendations, and this information may not always be available.

Session-based recommender systems (SBRS) focus on capturing dynamic short-term user preferences within a single session. Researchers have proposed various methods to improve diversity in SBRS.

The TailNet architecture was introduced to enhance the performance of long-tail recommendations. It utilizes a dual-stream attention mechanism to capture both short-term interests and long-term preferences.

Another approach, the ComiRec framework, employs a controllable multi-interest approach to capture multiple interests from the user's behavior sequence. This method is particularly effective for sequential recommendations.

A recurrent neural network (RNN) has also been employed to learn temporal preferences on both general and diverse items. This approach effectively captures the dynamic nature of user interests.

Our discussion focuses on modeling long-term static user preferences using historical data. We aim to develop a method that effectively balances diversity and relevance while addressing the limitations of existing two-stage approaches.

Achieving a balance between diversity and relevance remains a critical challenge in recommendation system design. Researchers have explored various approaches,

including two-stage methods, exploitation-exploration diversification, binary optimization formulations, and serendipity-based strategies. However, further research is needed to develop more effective and efficient methods for incorporating diversity into recommendation systems.

To address the problems associated with the two-stage approach, a single-stage approach utilizing latent factor model (LFM) with an additional diversity-promoting term was proposed and has been discussed in the chapter on diversity. They utilized a variance minimization term and incorporated bias and user-implicit feedback to improve the accuracy and diversity of the recommendations. Recently, graph convolutional neural networks (GCNNs) have also been used to establish accuracy–diversity trade-offs by learning joint convolutional representations from the nearest neighbor and furthest neighbor graph. Different configurations utilizing user and item similarity and dissimilarity graphs have been explored for linear and nonlinear rating and ranking prediction. These techniques do not depend on heuristics and secondary information and rely only on the rating data to provide novel and diverse recommendations to the user while maintaining the desired level of accuracy.

To enhance the accuracy of LFM further, recently, deep latent factor models (DLFMs) have been explored, with multiple layers of latent factors, and have been shown to outperform their shallow counterparts for applications related to recommender systems. However, applications on recommender systems all focus on increasing the accuracy alone and do not consider diversity.

Motivated by the remarkable performance of deep learning in various domains, this chapter ventures into the realm of DLFMs to elevate recommender systems to a new level of effectiveness. We unveil DeepDive, a novel DLFM meticulously crafted to strike a harmonious balance between recommendation accuracy and diversity.

Traditional recommender systems often rely on shallow latent factor models, restricted to a single-layer matrix factorization framework. While these shallow models have proven useful, they may fall short of fully capturing the intricate relationships between users and items that underpin accurate and engaging recommendations.

UNLOCKING THE POTENTIAL OF DEEP REPRESENTATIONS

DeepDive boldly steps beyond these constraints by embracing the power of deep representations. Its core premise lies in learning multiple layers of latent factors, meticulously crafted to mirror the complexities of user–item interactions. This enhanced ability to extract meaningful patterns and nuances from the data paves the way for predictions that are both more precise and diversified.

HARMONIZING ACCURACY AND DIVERSITY

At the heart of DeepDive's design lies a delicate balance between accuracy and diversity. To achieve this equilibrium, we weave a multifaceted strategy into its fabric:

- **Promoting Diversity Through Latent Factor Variance:** DeepDive actively encourages diversity by fostering user latent factors that exhibit minimal variance across all latent factors. This ingenious approach ensures

that recommendations encompass a wider spectrum of interests, circumventing the pitfalls of repetitive suggestions.

- **Joint Optimization Formulation:** The model expertly navigates the accuracy–diversity interplay through a meticulously crafted joint optimization formulation. This formulation simultaneously guides the model to achieve high prediction accuracy while maintaining ample diversity within recommended items.

At the heart of DeepDive lies a sophisticated joint optimization framework that harmonizes prediction accuracy and recommendation diversity. This framework skillfully employs deep latent factor models (DLFMs) to achieve high prediction accuracy while simultaneously promoting diversity through a novel latent factor variance minimization approach.

Leveraging the superior performance of DLFMs over traditional collaborative filtering techniques, DeepDive harnesses the power of deep representations to capture intricate user–item interactions with unprecedented precision. By learning multiple layers of latent factors, DeepDive unveils nuanced patterns and relationships that elude shallow models, paving the way for more accurate and insightful recommendations.

To foster diversity, DeepDive ingeniously minimizes the variance of user latent factors across diverse items. This approach ensures that users are exposed to a broader spectrum of recommendations, encompassing items that fall beyond their immediate preferences. By promoting a uniform affinity for all latent factors, DeepDive levels the playing field for diverse items, increasing their likelihood of being recommended.

The joint optimization framework masterfully navigates the delicate balance between accuracy and diversity. Through careful optimization, DeepDive simultaneously maximizes prediction accuracy while maintaining ample diversity within the recommended items. This interplay ensures that users receive recommendations that are both relevant to their interests and encompass a variety of perspectives.

The proposed diversity-promoting joint formulation for three-layer DeepDive is expressed as

$$\min_{U_1,U_2,U_3,V} \left\|Y - R\cdot(U_1U_2U_3V)\right\|_F^2 + \lambda_u \left\|U_1\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \lambda_d \sum_{u\in M} \mathrm{var}(U_{1u}) \qquad (19)$$

where U_1, U_2, U_n are the n-layer deep user latent factors, V is the item latent factors, and M is the number of users. The first three terms constitute the DLFM for promoting accuracy and the last term introduces diversity in the recommendations. Hyperparameters λ_u, λ_v, and λ_d provide an accuracy–diversity balance. They can be tuned to the desired level of accuracy for providing diverse recommendations. Following the previous discussion, ReLU type nonlinearity is introduced between the layers for learning the latent factors. Following the shallow diversity model discussed in a previous chapter, the variance term can be expressed as $\sum_{u\in M} \mathrm{var}(U_{1u}) = \left\|UD\right\|_F^2$ where $D = I_{F\times F} - \frac{1}{F}\bar{1}_{F\times F}$. Here, I is the identity matrix, $\bar{1}_{F\times F}$ is the matrix of all ones,

and F is the number of user latent factors for the first layer. Incorporating this into the formulation, we get

$$\min_{U_1,U_2,U_3,V} \left\|Y - R\cdot(U_1U_2U_nV)\right\|_F^2 + \lambda_u \left\|U_1\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \lambda_d \left\|U_1D\right\|_F^2 \tag{20}$$

We are omitting the explicit positivity constraints to keep the expression uncluttered.

Using the majorization-minimization (MM) technique, (20) can be written as

$$\min_{U_1,U_2,U_3,V} \left\|B - U_1U_2U_3V\right\|_F^2 + \lambda_u \left\|U_1\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 + \lambda_d \left\|U_1D\right\|_F^2 \tag{21}$$

where $B = U_1U_2U_3V + Y - R\cdot(U_1U_2U_nV)$

Alternate minimization is used to solve for the variables. This involves solving for each of the latent factor matrices individually, keeping the others constant. The sub-problems to solve and their corresponding closed form updates are given next.

$$\begin{aligned}
&U_1 \leftarrow \min_{U_1} \left\|B - U_1U_2U_3V\right\|_F^2 + \lambda_u \left\|U_1\right\|_F^2 + \lambda_d \left\|U_1D\right\|_F^2 \\
&\Rightarrow U_1 = B\left(U_2U_3V\right)^T \left[\left(U_2U_3V\right)\left(U_2U_3V\right)^T + \lambda_u I + \lambda_d DD^T\right]^\dagger \\
&U_2 \leftarrow \min_{U_2} \left\|B - U_1U_2U_3V\right\|_F^2 \\
&\Rightarrow U_2 = U_1^\dagger B\left(U_3V\right)^\dagger \\
&U_3 \leftarrow \min_{U_3} \left\|B - U_1U_2U_3V\right\|_F^2 \\
&\Rightarrow U_3 = U_1U_2^\dagger BV^\dagger \\
&V \leftarrow \min_{V} \left\|B - U_1U_2U_3V\right\|_F^2 + \lambda_v \left\|V\right\|_F^2 \\
&\Rightarrow V = \left[\left(U_1U_2U_3\right)^T U_1U_2U_3 + \lambda_v I\right]^\dagger \left(U_1U_2U_3\right)^T B
\end{aligned}$$

The latent factors are updated till the model converges. Once the latent factors are learned, the complete rating matrix X is computed using $X = U_1U_2U_3V$. Subsequently, the user estimated ratings on the unknown items are arranged in descending order for a particular test user, and the top T-rated items are recommended to the user.

BIBLIOGRAPHY

Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L. and Zhang, F., 2017. A hybrid collaborative filtering model with deep structure for recommender systems. In *AAAI* (pp. 1309–1315). AAAI.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.-S., 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on World Wide Web. International World Wide Web conferences steering committee* (pp. 173–182). ACM.

Kumar, K., Majumdar, A. and Chandra, M.G., 2022, November. DeepDive: Deep latent factor model for enhancing diversity in recommender systems. In *2022 IEEE international conference on data mining workshops (ICDMW)* (pp. 171–179). IEEE.

Li, Z. and Tang, J., 2017. Weakly supervised deep matrix factorization for social image understanding. *IEEE Transactions on Image Processing*, 26(1), pp. 276–288.

Mongia, A., Jain, V. and Majumdar, A., 2020. Deep matrix factorization on graphs: Application to collaborative filtering. In *Neural information processing: 27th international conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part IV 27* (pp. 754–762). IEEE.

Mongia, A., Jhamb, N., Chouzenoux, E. and Majumdar, A., 2020. Deep latent factor model for collaborative filtering. *Signal Processing*, 169, p. 107366.

Xue, H.-J., Dai, X., Zhang, J., Huang, S. and Chen, J., 2017. Deep matrix factorization models for recommender systems. *IJCAI* (pp. 3203–3209). AAAI.

8 Conclusion and Note to Instructors

8.1 INTRODUCTION

Having dedicated a decade to teaching Recommender Systems and Collaborative Filtering, I felt compelled to embark on the journey of writing this book. When I first began teaching this subject ten years ago, there existed only a single anthology devoted to the topic. As a result, the material presented was not cohesive, compiled by experts with diverse perspectives. Over the years, a multitude of books have emerged, yet the majority remain anthologies, while a select few are geared either toward research or toward practitioners aiming to construct such systems. Only one book claims to be a textbook, yet its length exceeds 500 pages, rendering it impractical for a single-semester course in any educational setting worldwide. While an experienced instructor could condense the contents of such a lengthy time, it would demand a substantial investment of time and effort.

In light of these shortcomings, I have endeavored to provide a comprehensive yet concise textbook that encompasses the fundamentals of Recommender Systems and Collaborative Filtering. This text is designed to serve as a valuable resource for both undergraduate and graduate students embarking on their exploration of this fascinating field. The book's structure meticulously guides readers through the theoretical underpinnings, algorithms, and practical applications of recommender systems, ensuring a thorough understanding of the subject matter.

To achieve this goal, I have adopted a unique approach that interweaves theoretical concepts with real-world examples and case studies. This pedagogical strategy effectively bridges the gap between abstract principles and practical implementation, enabling readers to grasp the intricacies of recommender systems and their applications in diverse domains.

Furthermore, I have meticulously curated the content to ensure its relevance to current industry practices and emerging trends in the field of recommender systems. This ensures that readers are equipped with the knowledge and skills necessary to navigate the rapidly evolving landscape of recommender systems and contribute meaningfully to their advancement.

Tailored specifically for instructors, my book stands as a concise yet comprehensive resource for imparting the intricacies of Recommender Systems and Collaborative Filtering. Its compact size, encompassing just over 100 pages, seamlessly adapts to the standard 13-week semester. Culminating years of teaching experience, this final chapter delves into my meticulously crafted approach to effectively disseminating the subject matter.

The evolution of my Recommender Systems and Collaborative Filtering course has been a journey of refinement and discovery. Initially, I experimented with two

 DOI: 10.1201/9781003511267-8

contrasting approaches: a purely theoretical course with rigorous mathematical underpinnings and a completely applied course focused on hands-on coding exercises. While both approaches offered valuable insights, I found that neither fully captured the essence of the subject matter.

Over time, I sought a middle ground, striking a balance between theoretical rigor and practical application. This evolved curriculum delved into the theoretical foundations of recommender systems while simultaneously equipping students with the practical skills to implement these concepts in real-world scenarios.

The results were transformative. Students not only grasped the underlying principles but also demonstrated the ability to apply their knowledge to solve practical problems. This newfound proficiency was a testament to the effectiveness of the balanced approach.

Inspired by the progress of my students, I embarked on the endeavor of writing this book. I aimed to encapsulate the lessons learned from years of teaching, providing a comprehensive yet practical guide to Recommender Systems and Collaborative Filtering.

Throughout the book, I have carefully interwoven theoretical concepts with real-world examples and case studies. This approach not only fosters a deeper understanding of the subject matter but also prepares readers to navigate the ever-evolving landscape of recommender systems.

Enriching the content further, I have included hands-on coding exercises and projects that allow readers to apply their newfound knowledge in a practical setting. These exercises not only reinforce key concepts but also cultivate valuable problem-solving skills.

Given the dynamic and evolving nature of recommender systems, I discovered that continuous evaluation proved to be the most effective approach for this course. In the initial stages, I experimented with a traditional assessment structure, employing only mid-semester and final proctored exams, supplemented by a couple of take-home assignments sprinkled throughout the semester. However, this method yielded less-than-satisfactory results, as students struggled to maintain consistent engagement and understanding.

Recognizing the limitations of this approach, I opted for a more comprehensive evaluation strategy that emphasized continuous assessment. This involved incorporating regular quizzes, both formative and summative, along with in-class exercises and group projects. This shift toward continuous evaluation yielded significant improvements in student performance.

The frequent assessments served several purposes. First, they provided timely feedback, allowing students to identify and address any knowledge gaps promptly. Second, they encouraged active participation and engagement, fostering a more dynamic learning environment. Third, they promoted a deeper understanding of the subject matter, as students were consistently revisiting and applying key concepts.

In conclusion, continuous evaluation emerged as a cornerstone of my teaching methodology for the Recommender Systems and Collaborative Filtering course. This approach proved instrumental in enhancing student engagement, understanding, and overall performance.

8.2 COURSE ORGANIZATION

Drawing upon my extensive teaching experience, I have crafted a comprehensive weekly course plan meticulously tailored to a 13-week semester, the increasingly prevalent standard. Assuming two 90-minute class sessions per week, the plan seamlessly integrates theoretical concepts with practical applications, ensuring a thorough understanding of recommender systems and collaborative filtering.

Week #	Topic	# lectures
1	Course organization, evaluation policy, etc.	1
	Quiz 0: Linear algebra and optimization	1
2	Chapter 1	1
3	Chapter 2: User-based neighborhood model	1
	Chapter 2: Item-based neighborhood model	1
4	Chapter 2: Neighborhood selection, variance weighting, significance weighting, coverage	1
	Quiz 1: Neighborhood models	1
5	Chapter 3: Baseline/biases. Estimation	1
	Chapter 3: Neighborhood models with baseline correction	1
6	Chapter 4: Latent factor model	1
	Chapter 4: Matrix factorization	1
7	Chapter 4: Nuclear norm minimization	1
	Quiz 2: Biases and latent factor model	1
8	Chapter 5: Matrix factorization on graphs	1
	Chapter 5: Nuclear norm minimization on graphs	1
9	Chapter 5: Label consistency	1
	Quiz 3: Recommendations with metadata	1
10	Chapter 6: Need for diversity	1
	Chapter 6: Matrix factorization-based solution	1
11	Chapter 6: Nuclear norm minimization and diversity	1
	Quiz 4: Diversity in recommendations	1
12	Chapter 7: Deep latent factor model	1
	Chapter 7: Deep latent factor model on graphs	1
13	Chapter 8: Deep latent factor model and diversity	1
	Quiz 5: Deep learning in collaborative filtering	1

While the provided course plan serves as a valuable framework, it's essential to recognize that flexibility is paramount in the hands of an experienced instructor. The plan can be dynamically adjusted to accommodate the varying readiness levels and learning styles of students.

It's crucial to acknowledge that the initial phase of the course, encompassing the first two chapters, is designed to be relatively straightforward, relying on intuitive concepts. This gentle introduction often lulls students into a sense of comfort, leading them to underestimate the complexity of the material that lies ahead.

As the course delves into the intricacies of the latent factor model, a noticeable shift in pace occurs, potentially overwhelming unprepared students. To mitigate this challenge, it's imperative to explicitly inform students of this impending acceleration early in the course. This forewarning allows them to mentally prepare, adjust their study habits, and seek additional support if necessary.

In addition to forewarning students, experienced instructors can employ various strategies to bridge the gap between the initial simplicity and the subsequent complexity. These strategies include the following:

- **Emphasize Connections between Early Concepts and Later Ones:** Regularly revisit and reinforce fundamental principles, demonstrating their relevance to more advanced topics.
- **Provide Real-World Examples and Case Studies:** Illustrate theoretical concepts with practical applications, helping students grasp the tangible impact of the subject matter.
- **Offer Diversified Learning Resources:** Supplement lectures with supplementary materials, such as practice problems, interactive simulations, and recommended readings, catering to different learning preferences.
- **Encourage Active Participation and Collaboration:** Facilitate in-class discussions, group projects, and peer-to-peer learning opportunities, promoting active engagement and deeper understanding.

By adopting these strategies, instructors can effectively guide students through the transition from the initial ease of the course to the subsequent complexities, ensuring a smooth and rewarding learning experience for all.

8.3 EXPECTATION FROM PUPILS

Given the target audience of senior undergraduate or graduate students, one would reasonably expect them to possess a strong foundation in linear algebra, a prerequisite for grasping the intricacies of recommender systems. However, my experience has consistently revealed a gap in students' linear algebra proficiency, necessitating a comprehensive assessment of their preparedness.

Therefore, I strongly recommend administering a diagnostic quiz during the first week of classes to gauge their level of linear algebra knowledge. This assessment will not only inform the instructor of the student's strengths and weaknesses but also provide valuable insights into the need for additional support or remedial measures.

In addition to linear algebra, optimization plays a crucial role in recommender systems. Unfortunately, many students lack prior exposure to optimization concepts, posing a significant challenge. To address this deficiency, I suggest encouraging students to enroll in remedial online courses or workshops specifically tailored to optimization.

Alternatively, considering the fundamental importance of linear algebra and optimization to recommender systems, it might be prudent to establish these subjects as compulsory prerequisites for the course. This approach would ensure that students

possess the necessary mathematical foundation to fully comprehend and engage with the course material.

Furthermore, to effectively bridge the knowledge gap and facilitate a smooth learning experience, I recommend incorporating the following strategies:

1. **Provide a Comprehensive Linear Algebra Review:** Dedicate the initial sessions to reviewing key linear algebra concepts, ensuring that students have a solid grasp of the fundamentals before delving deeper into recommender systems.
2. **Integrate Optimization Concepts Seamlessly:** Gradually introduce optimization principles within the context of recommender systems, illustrating their practical applications and reinforcing their relevance to the subject matter.
3. **Offer Supplementary Resources:** Provide students with access to additional resources, such as online tutorials, practice problems, and recommended readings, to supplement their understanding of linear algebra and optimization.
4. **Encourage Peer-to-Peer Learning:** Foster a collaborative learning environment by encouraging students to work together on assignments, discuss concepts, and share their knowledge.
5. **Seek Feedback and Adapt Accordingly:** Regularly solicit feedback from students to gauge their understanding of the material and identify areas for improvement. Adapt the teaching approach as needed to address any lingering challenges.

By implementing these strategies, instructors can effectively address the lack of linear algebra and optimization proficiency among students, ensuring that they are well-equipped to successfully navigate the course and gain a comprehensive understanding of recommender systems.

8.4 EVALUATION

Recommender systems are undoubtedly a practical field, and students must be equipped with a thorough understanding of both theoretical underpinnings and practical applications. This balance should be reflected in the evaluation strategy employed for the course.

To effectively assess students' proficiency in recommender systems, a combination of theoretical and practical evaluation methods is essential. Theoretical assessments, such as exams and quizzes, can gauge students' understanding of key concepts, algorithms, and mathematical principles. Practical assessments, such as coding exercises, projects, and case studies, can evaluate students' ability to apply their knowledge to real-world scenarios and implement recommender systems in a practical setting.

The ideal evaluation strategy should strike a balance between these two approaches, ensuring that students are not only knowledgeable about the theoretical foundations but also capable of applying their knowledge to solve practical problems.

This balanced approach will effectively assess students' overall competence in recommender systems.

Quiz 0 serves as a comprehensive pen-and-paper assessment designed to gauge students' preparedness for the course. Standard questions encompassing inverse problems, vector spaces, matrix decomposition, and least square solutions form the foundation of this quiz. While this quiz may appear as an outlier, it plays a crucial role in providing me with valuable insights into students' overall grasp of linear algebra and optimization. Although I refrain from grading this quiz, I meticulously gather statistical data to evaluate the collective readiness of the class on various linear algebra and optimization topics. This information proves invaluable in tailoring the course structure and pace to effectively cater to the student's needs.

Quiz 1 maintains the pen-and-paper format, further assessing students' understanding of fundamental concepts. The following is an illustrative example of a Quiz 1 question:

> *Employ user-based recommendations to fill in the missing ratings. For similarity, use inverse absolute distance. Use a neighborhood of two; if there is a tie, consider three nearest neighbors. Use nearest neighbor interpolation. Round off to the nearest integer.*

	M1	**M2**	**M3**	**M4**	**M5**	**M6**
U1		3	5		2	
U2	4		4	3		1
U3	3	5	3	2		
U4	2			4	3	4
U5	2	1	3			3
U6			2	5	4	
U7		2		4	1	4

To effectively assess students' programming abilities in the context of recommender systems, a take-home assignment utilizing the MovieLens dataset from https://movielens.org/ proves to be a valuable tool. This dataset provides a rich source of real-world movie rating data, enabling students to apply their programming skills to practical scenarios.

In assignment 1, students are tasked with implementing both user-based and item-based recommender systems. This assignment not only assesses their programming skills but also their understanding of the underlying principles of collaborative filtering.

To further delve into the nuances of recommender systems, students are instructed to implement different weighting strategies and compare their impact on accuracy and coverage. This exercise encourages them to explore the trade-offs between these two metrics and gain insights into the effectiveness of different weighting schemes.

To provide a concrete illustration, a sample assignment is outlined.

Build a simple user-based and item-based recommender system (papers have been provided).

The dataset will have "available" and "missing" ratings. For evaluation, you should use five-fold cross-validation. In each run, use four parts for training and the remaining one part for testing. Use the training set to predict the ratings of the test set.

To test how good or bad your recommender system is, you should compute the mean absolute error (MAE) on the test set.

For measuring similarity, you should use the ***Cosine Similarity****.*

Report how the MAE coverage changes when the number of neighbors "K" varies (K = 10, 20, 30, 40, 50) for each of both user-based and item-based approaches.

8.4.1.1 Bonus

If you implement significance weighting and variance weighting and show the results, you will receive one bonus mark for each type, that is, if you implement both, you will get two.

Ideally, Quiz 2 would comprise a combination of pen-and-paper and coding questions to comprehensively assess students' understanding of the latent factor model. However, implementing such a hybrid quiz poses practical challenges. Consequently, Quiz 2 focuses primarily on theoretical concepts, incorporating questions related to baseline correction. The practical application of the latent factor model is evaluated separately through a dedicated assignment. A typical example of Quiz 2 is given next:

8.4.1.1.1 Consider the Following User–Item Matrix

	M1	M2	M3	M4	M5
U1		3	5		2
U2	4		4	3	
U3	2			4	3
U4	3	1	3		
U5		2		4	2

a. *Write the Potter estimation formula for user and item biases.*
b. *Compute the user biases and item biases for each user and each item using Potter estimation. Assume* $\lambda_1=2$ *and* $\lambda_2=2$.
c. *Using the computed biases, compute the baseline ratings for the missing values.*

Assignment 2 puts students' coding prowess to the test by challenging them to implement the nuclear norm minimization and matrix factorization algorithms from scratch. Armed with the MovieLens dataset, a treasure trove of real-world movie ratings, students apply these algorithms to generate recommendations and evaluate their effectiveness. This hands-on exercise not only assesses their programming skills but also reinforces their grasp of latent factor model principles.

Quiz 3 delves into the realm of recommender systems that incorporate metadata, broadening students' understanding of real-world applications. Students can

demonstrate their theoretical comprehension by elaborating on various metadata-based algorithms discussed in the course material. Alternatively, they can showcase their coding proficiency by implementing one algorithm during the in-class quiz and the remaining ones as a corresponding take-home assignment. This flexibility accommodates different learning styles and ensures that students are evaluated on both their theoretical knowledge and practical application skills.

Quiz 4 takes an in-class coding approach to assess students' understanding of recommendation diversification algorithms. Students are tasked with implementing one of these algorithms, demonstrating their ability to translate theoretical concepts into practical code. However, in-class coding quizzes raise concerns about plagiarism. To address this challenge, I employ a strategic seating arrangement, assigning students sitting next to each other different algorithms. For instance, one student might code the factorization-based diversity model, while the other implements the nuclear norm-based diversity model.

Quiz 5 adopts an in-class coding format to assess students' understanding of deep learning models in recommender systems. Recognizing the limitations of rote memorization and written explanations, this quiz directly engages students in implementing different deep learning models. To prevent plagiarism, students are assigned distinct models, ensuring that adjacent students are not working on the same model. This approach effectively evaluates students' comprehension and implementation of deep learning techniques while maintaining academic integrity.

Index

B

baseline correction, 32, 33, 124
biases in ratings, 8, 30–35

C

cold-start problem, 8, 22, 26, 45, 99
collaborative filtering (CF), 5–8, 13–19, 21–23, 25, 26, 29–32, 36, 37, 44, 45, 49, 50, 57–60, 62–66, 69, 71–75, 82, 98–105, 107, 109, 111, 113, 115, 118–120, 123

D

deep learning models, 11, 12, 125
diversity in recommendations, 120

G

graphical latent factor models, 9, 10, 12

I

interpolation weights, 18, 21, 39–42, 48
item-based collaborative filtering, 7, 15, 19–27, 35, 38, 39, 72, 85, 120, 123, 124

L

label-consistency, 65–71, 120
latent factor models, 8–12, 43–58, 60, 74, 85, 98–116, 120, 121, 124

M

machine learning, 4, 5, 40, 46, 47
matrix completion, 49–53, 55, 57, 58, 60, 61, 65, 66, 73–75, 77, 78, 92
matrix factorization, 46–50, 52–54, 57, 58, 61–63, 69–72, 74–77, 79, 85–91, 98, 101, 106, 107, 109–112, 114, 120, 124
memory-based techniques, 27
metadata, 9–12, 59–80, 91, 113, 120, 124, 125

N

neighborhood-based models, 7–9, 13–27, 60, 71, 72, 74
nuclear norm minimization, 54–58, 63–69, 72, 77–80, 96, 120, 124

R

recommender systems, 4–13, 44, 46, 47, 55, 57, 59, 61, 81–96, 98, 99, 101–104, 107, 113, 114, 118–125

S

significance weighting, 35–39, 120, 124

U

user-based collaborative filtering, 7, 15–25, 35, 37–40, 72, 120, 123, 124
user demographics, 9, 10, 59, 65, 99, 100, 110