Learning for Top-N Recommendations

High-Dimensional and Heterogeneous Information

Yifan Chen

Learning for Top-N Recommendations

High-Dimensional and Heterogeneous Information

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op 8 oktober 2019, te 12.00 uur

door

Yifan Chen

geboren te Sichuan

Promotiecommissie

Promotor:		
	prof. dr. M. de Rijke	Universiteit van Amsterdam
Co-promotor:		
	dr. I. Markov	Universiteit van Amsterdam
Overige leden:		
	prof. dr. H. Haned	Universiteit van Amsterdam
	prof. dr. A. Hanjalic	Technische Universiteit Delft
	dr. ir. J. Kamps	Universiteit van Amsterdam
	prof. dr. E. Kanoulas	Universiteit van Amsterdam
	prof. dr. Y. Wang	Hefei University of Technology

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the China Scholarship Council.

Cover by Feng Li
Printed by Off Page, Amsterdam, The Netherlands

ISBN: 978-94-6182-966-5

Acknowledgements

The road towards a PhD degree has been tough for me. Fortunately, I am tougher. I cannot fully experience the joy of successfully completing my PhD without thousands of failures on the way. Although the frustration caused by the failures was impressive, I am grateful for the failures since they have taught me lessons and the most important lesson I have learned is, quoted from Aristotle, "I know nothing except the fact of my ignorance." The PhD degree is not the end, but the start of my academic career and the key of which, I believe, is the acknowledgment of my ignorance.

I would not have been able to reach this stage of my PhD studies without the help, support and guidance from my supervisors, family and friends. First, I would like to thank my supervisor, Prof. Maarten de Rijke, who always trusted and supported me and my research no matter how many mistakes I made, how many failures I have encountered. I am fully impressed by your rigorous attitude towards research and profound insights into science. I am grateful to have you as my supervisor. Before you helped me develop my research plans, I could not imagine how to pursue a PhD at the University of Amsterdam. You also regularly and kindly helped me back on track whenever I got stuck or fell behind. I also thank my co-promoter Ilya Markov. Meeting with you is always enjoyable, and your comments on my research are always helpful. Furthermore, I am honored to have Alan Hanjalic, Evangelos Kanoulas, Hinda Haned, Jaap Kamps, and Yang Wang as my committee members and thank you for reviewing my thesis.

I thank all the members of the Information and Language Processing Systems (ILPS) group. Studying and learning at ILPS has been the best time of my life. It has been my pleasure to study and work with such talented and helpful colleagues. I am not an out-going man, but due to your hospitality, I have never suffered a culture shock. My sincere thanks go to: Ali A, Ali V, Alexey, Amir, Ana, Andreas, Anna, Arezoo, Artem, Bob, Boris, Chang, Christof, Christophe, Chuan, Dan, Dat, David D, David S, Evangelos, Fei, Hamid, Harrie, Hinda, Hosein, Ilya, Jiahuan, Jie, Jin, Julia, Julien, Katya, Ke, Maarten dR, Maarten M, Maartje, Mahsa, Mariya, Marzieh, Maurits, Mostafa, Mozhdeh, Nikos, Olivier, Pengjie, Petra, Praveen, Rolf, Sami, Shaojie, Spyretta, Svitlana, Trond, Vera, Wanyu, Xiaohui, Xinyi, Yangjun, Zhaochun, and Ziming. I would like to especially thank Anna, Chang, Dan, Julia, Mostafa, Nikos, Pengjie, Wanyu, Xinyi, and Ziming for all the nice discussions, which definitely inspired my research. I would also like to thank Anna, Chang, Dan, Harrie, Hosein, Jiahuan, Jie, Julia, Maurits, Mostafa, Nikos, Pengjie, Rolf, Shaojie, Spyretta, Wanyu, Xinyi, Yangjun, and Ziming. I really enjoyed your company, which undoubtedly made my PhD life brilliant and interesting. I also specially thank Petra. I have bothered you many times for many trivial things, but you have always been nice to me and help me out.

Besides, I also need to thank Feng Li, who designed the cover for me. The design truly reflects your talents in art. I thank Hosein, Mostafa and Xinyi, who are always willing to answer my questions regarding the thesis. I thank Harrie and Maurits who spent time and effort to translate the summary to Dutch. I thank Prof. Weiming Zhang, who constantly encouraged me and cared about me. I thank Pengjie Ren, Xiang Zhao and Yang Wang. Whenever I lost my direction, you showed me the direction. I also

thank the China Scholarship Council, who sponsored my research.

Pengjie once told me that “one of the biggest challenges in studying abroad is to endure loneliness.” Fortunately, I have been able to overcome this challenge smoothly, thanks to the many nice people I have met during my stay in Amsterdam. I express my sincere thanks to: Chenxi, Huan Z, Junjiao L, Kun D, Lin H, Mateusz, Min L, Shihan, Shuai L, Shuo C, Song Y, Songyu, Wei Zh, Xiao T, Xiaomeng, Yang H, Yang L, Ye D, Yunlu, Zhe L, Ziyang, Zijian, Qi W, Zhihui. My friends from China also constantly support me, which definitely deserve my thanks: Chao C, Jie M, Junjiao G, Song X, Qingyang, Xiaorong, Xuexiao. I also want to thank Gareth Bale, the English football player. I always feel encouraged when I see how hard you have endured and how much you have achieved. Last but not least, I express my deepest gratitude to my parents and relatives. Mum and dad are always my primary motivations to work. My younger cousin, Yuang, also never forgets me. Ever since my grandma learned to use WeChat, I have always received your remote greetings. And my grandpa always expected to see my graduation.

Yifan Chen
August 9th, 2019

Contents

1 Introduction **1**
- 1.1 Research Outline and Questions . . . 2
  - 1.1.1 Leveraging high-dimensional information . . . 3
  - 1.1.2 Integrating heterogeneous information . . . 4
  - 1.1.3 High-dimensional and heterogeneous information . . . 5
- 1.2 Main Contributions . . . 6
  - 1.2.1 Theoretical contributions . . . 6
  - 1.2.2 Algorithmic contributions . . . 6
  - 1.2.3 Empirical contributions . . . 7
- 1.3 Thesis Overview . . . 7
- 1.4 Origins . . . 8

I High-Dimensional Information **11**

2 Clustering Items for Item-based Collaborative Filtering **13**
- 2.1 Introduction . . . 13
- 2.2 Related Work . . . 15
  - 2.2.1 Item-based collaborative filtering . . . 15
  - 2.2.2 Local models . . . 16
  - 2.2.3 Subspace clustering . . . 16
- 2.3 Notation . . . 17
- 2.4 Block-Aware Regularizations . . . 17
  - 2.4.1 Block-aware similarity regularization . . . 17
  - 2.4.2 Block regularized similarity model . . . 18
  - 2.4.3 Overcoming the negative effect . . . 18
  - 2.4.4 Connection to sparsity and transitivity . . . 19
  - 2.4.5 Optimization . . . 20
- 2.5 Scalable Block-Aware Regularization . . . 21
  - 2.5.1 Offline computation of F . . . 22
  - 2.5.2 Block-aware similarity dropout . . . 22
  - 2.5.3 Block regularized factored similarity model . . . 23
- 2.6 Experimental Setup . . . 23
  - 2.6.1 Research questions . . . 23
  - 2.6.2 Datasets . . . 23
  - 2.6.3 Methods used for comparison . . . 24
  - 2.6.4 Evaluation methodology . . . 25
- 2.7 Experimental Results . . . 25
  - 2.7.1 RQ1.1: Top-N recommendation performance . . . 27
  - 2.7.2 RQ1.2: Impact of block-aware similarity regularizations . . . 28
  - 2.7.3 RQ1.3: Training convergence and stability . . . 29
- 2.8 Summary . . . 30

3 Top-N Recommendation with High-dimensional Side Information **31**

3.1 Introduction . . . 31
3.2 Related Work . . . 32
3.3 The Proposed Approach . . . 32
3.3.1 Notation . . . 32
3.3.2 Model description . . . 33
3.3.3 Solution . . . 34
3.4 Experiment . . . 35
3.4.1 Setup . . . 35
3.4.2 Results and analysis . . . 36
3.5 Summary . . . 38

4 Collective Variational Auto-encoder for Top-N Recommendation 39
4.1 Introduction . . . 39
4.2 Preliminaries . . . 41
4.2.1 Notation . . . 41
4.2.2 Linear models for top-N recommendation . . . 41
4.2.3 Autoencoders for collaborative filtering . . . 42
4.3 Method . . . 43
4.3.1 Collective variational auto-encoder . . . 43
4.3.2 Variational inference . . . 44
4.3.3 Implementation details . . . 45
4.4 Experiments . . . 46
4.4.1 Experimental setup . . . 46
4.4.2 Experimental results . . . 48
4.5 Related Work . . . 50
4.5.1 Top-N recommendation with side information . . . 50
4.5.2 Deep learning for hybrid recommendation . . . 50
4.6 Summary . . . 51

II Heterogeneous Information 53

5 Local Variational Feature-based Similarity Model 55
5.1 Introduction . . . 55
5.2 Problem Definition . . . 57
5.3 Related Work . . . 58
5.3.1 Feature-based methods . . . 59
5.3.2 Feature-based similarity models . . . 60
5.3.3 Local collaborative filtering . . . 61
5.3.4 Review-based recommendation . . . 61
5.4 Local Variational Feature-based Similarity Models . . . 62
5.4.1 Overview . . . 62
5.4.2 Model description . . . 63
5.5 Model Optimization . . . 65
5.5.1 Variational inference . . . 66
5.5.2 Variational E-step . . . 68

5.5.3 Variational M-step . . . 68
5.5.4 Computational analysis . . . 69
5.6 Experimental Setup . . . 70
5.6.1 Research questions . . . 70
5.6.2 Datasets . . . 70
5.6.3 Evaluation protocol . . . 71
5.6.4 Methods used for comparison . . . 72
5.6.5 Parameter settings . . . 73
5.6.6 Experiments . . . 78
5.7 Results and Analysis . . . 78
5.7.1 Performance comparison . . . 78
5.7.2 Effect of modeling global and local similarities . . . 79
5.7.3 Effect of feature sparsity . . . 82
5.7.4 Effect of item cold-start . . . 83
5.7.5 Performance on a large-scale dataset . . . 84
5.8 Summary . . . 86

6 Personalised Reranking of Paper Recommendations 87
6.1 Introduction . . . 87
6.2 Models . . . 90
6.2.1 Production baseline . . . 90
6.2.2 Proposed model . . . 90
6.3 Experiments . . . 99
6.3.1 Research questions . . . 99
6.3.2 Dataset . . . 99
6.3.3 Experimental setup . . . 100
6.4 Results and Analyses . . . 101
6.4.1 RQ5.1: Overall comparison . . . 102
6.4.2 RQ5.2: Utility of content and behavior features . . . 102
6.4.3 RQ5.3: Utility of paper aspects . . . 103
6.5 Related work . . . 104
6.5.1 Academic search . . . 104
6.5.2 Academic paper recommendation . . . 105
6.5.3 Citation recommendation . . . 106
6.5.4 Top-N recommendation . . . 106
6.5.5 Reranking the output of a production system . . . 107
6.6 Summary . . . 107

7 Personalized Interaction Selection for Factorization Machines 109
7.1 Introduction . . . 109
7.2 Preliminaries . . . 111
7.2.1 Factorization machines . . . 111
7.2.2 Bayesian variable selection . . . 112
7.3 Model Description . . . 113
7.3.1 Personalized factorization machines . . . 113
7.3.2 Bayesian personalized feature interaction selection . . . 113

7.3.3 Hereditary spike-and-slab prior . . . 114
7.3.4 Variational inference . . . 116
7.4 Experimental Setup . . . 119
7.4.1 Research questions . . . 119
7.4.2 Dataset . . . 119
7.4.3 Baselines . . . 120
7.4.4 Evaluation . . . 120
7.4.5 Implementation details . . . 121
7.5 Experimental Result and Analysis . . . 121
7.5.1 RQ6.1: Results . . . 121
7.5.2 RQ6.2: Impact of embedding size . . . 122
7.5.3 RQ6.3: Impact of training . . . 123
7.5.4 RQ6.4: Explainability for recommendation . . . 124
7.6 Related Work . . . 125
7.6.1 Factorization machines . . . 126
7.6.2 Feature selection . . . 126
7.7 Summary . . . 127

8 Conclusions **129**
8.1 Main Findings . . . 129
8.1.1 Item clustering for top-N recommendation . . . 129
8.1.2 Leveraging high-dimensional side information . . . 130
8.1.3 Cold-start recommendation . . . 130
8.1.4 Personalized feature interaction selection . . . 131
8.2 Future work . . . 132
8.2.1 RQ1: Scalability and online grouping . . . 132
8.2.2 RQ2 & RQ3: Fine-designed neural extension . . . 132
8.2.3 RQ4 & RQ5: Incorporating semantic information . . . 133
8.2.4 RQ6: Group-level personalized selection . . . 133

Acronyms **135**

Summary **153**

Samenvatting **155**

1 Introduction

In the early years of information retrieval, *reactive* search systems that return a ranked list of results in response to queries issued by users were overwhelmingly prevalent [12, 26, 194]. With the explosive growth of information, users may be unaware of information that is potentially useful to them. Systems that passively wait for queries may no longer be able to fulfill the information need of users [8]. Instead, *proactive* search systems [2, 191] that actively push information to users have gained popularity; they provide a type of functionality known as *zero-query ranking* [253]. Top-N recommender systems, which are at the heart of such systems, provide effective zero-query ranking by utilizing either demographic, content or historical information.

Top-N recommendation models have been deployed in various online applications, ranging from the recommendation of music [117, 252], movies [85], news [59, 131, 132], products [146, 157], articles [229] to the recommendation of points-of-interest [79, 128, 140, 147, 260] and social links [44, 161, 257, 261]. Over the last decades, various efforts have been made to apply machine learning methodologies to provide top-N recommendations. Collaborative filtering (CF)-based methods, which make use of historical interactions between users and items, have achieved significant success [199]. Originally proposed by Goldberg et al. [84], the core idea behind CF is to recommend to one user items that are preferred by other similar minded users; this idea has been widely implemented into top-N recommender systems.[1]

Although CF-based methods have achieved great success, they have recently faced severe challenges due to the fast growth of real-world recommender systems. The increasing number of users and items involved in the recommendation systems increases the dimensionality of ratings, which challenges the scalability of CF methods. Normally, the complexity of CF models is at least linear in the number of users and items [63]. This becomes a bottleneck for practical systems, which may have millions of users and items. Another issue brought by high-dimensional rating information is rating sparsity, that is, the number of ratings is extremely small compared to the number of ratings between users and items. The performance of CF methods can greatly suffer when ratings are very sparse [170]. In addition, CF methods suffer from the cold-start problem: they cannot recommend items to new users since they have not provided any ratings or recommend new items to users since they have never received

[1] Hereafter, we refer to user-item historical interactions (e.g., rated, clicked, viewed, purchased and etc.) as rating information.

any ratings.

Due to the wide availability of information associated with users or items, referred to as *side information*, researchers have recently been interested in utilizing such information to compensate for the sparsity of ratings. Unfortunately, in many scenarios, side information is also high-dimensional [43]. The scalability and sparsity issues mentioned above are a recurring problem if such information is utilized.

So far, we have described scenarios where the available information, ratings or side information, is homogeneous. In practice, the available information is often heterogeneous, i.e., multi-behaviors of users or multiple types of auxiliary information associated with users or items [33, 108, 267]. For example, a user can rate, click, view or purchase an item, each of which reflects a certain aspect of the user's behavior. Similarly, information describing items is not necessarily homogeneous. For example, in the movie domain, meta information (actor and actress, director or producer), plots, posters or trailers are all relevant to recommendation. The information that made up the profile of a user can also be of multiple types, e.g., social networking, user tags, etc. It remains a challenging task how to effectively fuse heterogeneous information for top-N recommendation. Besides, the fusion of multiple types of information can dramatically increase the dimensionality. It is even more challenging to leverage high-dimensional and heterogeneous information.

In this thesis, we investigate solutions to the top-N recommendation task by leveraging high-dimensional and heterogeneous information. (1) To address the high-dimensionality of ratings, we rely on item-based collaborative filtering, by modeling item-item relations rather than user-item relations, the complexity of which is unaffected by the number of users. We also categorize items into subgroups to ensure the scalability when we have a large number of items. (2) We utilize high-dimensional side information to enhance top-N recommendations. We investigate dimension reduction for side information in a supervised way. The dimension reduction technique is embedded into the training procedure of recommendation models. (3) When high-dimensional side information also contains noise, we provide a novel method based on a variational auto-encoder (VAE), where feature embeddings are collectively learned with user factors via the inference network and the generation network of the VAE. (4) We further research the problem of recommending top-N new items by combining ratings of warm-start items and item features. We learn low-dimensional item representations from item features, based on which we learn item similarity functions. (5) We study the problem of paper reranking where we have multi-behavioral information and paper content information. We propose a hybrid reranking model that includes multiple content-based metrics and a joint matrix factorization method. (6) Finally, we investigate personalized feature interaction selection for factorization machines to utilize high-dimensional and heterogeneous information. We select the personalized feature interactions by forming a Bayesian variable selection method.

1.1 Research Outline and Questions

We divide the thesis into chapters based on the data sources utilized for top-N recommendations. The thesis contains three research themes: top-N recommendation with (1) high-dimensional information (Chapter 2–4); and (2) heterogeneous informa-

tion (Chapter 5–7). (3) high-dimensional and heterogeneous information (Chapter 7). Below, we list the main research question of every chapter.

1.1.1 Leveraging high-dimensional information

In recent years, tremendous growth of customers and products has been witnessed in many online e-commerce systems, where the rating information is increasingly high-dimensional. It becomes extremely difficult to produce high-quality recommendations efficiently for millions of customers and products [196]. Fortunately, the issue of scalability caused by the number of users can be addressed via item-based collaborative filtering (ICF). ICF methods use item-item relations, the complexity of which does not depend on the number of users. State-of-the-art performance for top-N recommendation is achieved by learning a *sparse* item similarity matrix [169]. The sparsity also ensures the efficiency of personalization as only a few item-item relations will be examined during recommendation. Clustering provides a feasible way to handle the large number of items. By categorizing items into sub-groups, it is possible to feed ratings within each sub-group to a separate model. However, item clustering has not yet been studied for ICF methods. A straightforward solution that statically clusters items based on rating and separately estimates local models for each sub-group is not ideal. This has motivated us to find a better way to integrate item clustering with ICF methods, which leads to the following research question:

RQ1 How can we effectively perform item clustering for item-based collaborative filtering methods?

To answer **RQ1**, we propose a novel regularization term for ICF methods. The introduced regularization encourages a block-diagonal structure of the item similarity matrix, where each block captures a latent item group. Through block-diagonal regularization, the latent item groups can be identified adaptively according to the item similarities, which is constantly optimized during the training procedure. Item similarities can be optimized globally rather than separately within each item group.

Besides the scalability issue, another negative effect caused by high-dimensionality is that rating information becomes extremely sparse. The sparsity of ratings impacts the performance of CF-based methods. To overcome rating sparsity, there is great interest in taking advantage of side information, i.e., the additional information associated with users or items, e.g., product reviews, movie plots, etc. Methods for top-N recommendation with side information have been studied widely [170, 229, 230]. Side information is also widely available, especially in multimedia scenarios, which can be in the form of text [229], images [91] or videos [66]. Therefore, side information is generally high-dimensional [43]. For example, when side information is the textual description of items, by regarding each unique term in the corpus as one dimension, it is indisputably high-dimensional. Nonetheless, existing methods overlook this fact when utilizing side information, and hence, they are facing problems of efficiency and accuracy due to the curse of high dimensionality. This has led to the following research question:

RQ2 Can we reduce the dimension of side information for effective top-N recommendation?

To answer **RQ2**, we provide a joint learning model that simultaneously performs dimension reduction and learns an item similarity model for top-N recommendation. We introduce a projection matrix that maps side information from high-dimensional feature space to low-dimensional space. The motivation for using a projection is based on the assumption that while the original space of side information is high, the intrinsic dimensionality of it is low [96]. We introduce locality preserving projection (LPP) to an ICF model, which achieves dimension reduction of side information while preserving locality information (the relations among items in the original space of side information).

However, the intrinsic low-dimensional space does not necessarily exist for all side information, especially when side information contains noise. Therefore, effective dimension reductions can hardly be expected via the linear projection methods like LPP and a process of denoising side information will be necessary. Denoising auto-encoder (DAE) have been proposed to denoise features by recovering clean inputs from manually corrupted inputs [239]. However, the corruption methods are tailored for inputs of different types, which has restricted the generalization of DAE. In comparison, variational auto-encoder (VAE) provides a better way to denoise features, as it automatically fits the noise based on data [134, 141]. Therefore, we are motivated to raise the following research question:

RQ3 How can we utilize high-dimensional side information with noise for top-N recommendation?

To answer **RQ3**, we take advantage of VAEs. Unlike existing VAE-based methods [134, 141], we provide a collective VAE, which feeds both ratings and side information into the same inference and generation network. The dimensions of the side information is taken as the number of samples rather than the input dimensions of the neural network. Therefore, the complexity of the model is free from the dimensionality of side information. This is similar to the idea drawn from ICF that overcomes the scalability issue caused by the large number of users. The quality of latent representations encoded by the VAE when feeding ratings and side information respectively, can be collectively improved.

1.1.2 Integrating heterogeneous information

One practical problem regarding CF methods for top-N recommendation is that their recommendations of new items that received no ratings from users is no better than random. This problem is typically referred to as the *cold-start* problem. Item features (content information describing items) are typically utilized to address this problem. Although utilizing similar information as the above-mentioned problems, cold-start top-N recommendation will have higher requirements on the quality of item features. This is because we need to infer ratings directly from item features for the new items. An effective fusion of ratings and item features will then be called for, which has led to a research question regarding the integration of heterogeneous information.

RQ4 How can we effectively fuse item features with ratings for the recommendation of top-N new items?

Solutions for the previous research questions are not an idea answer for **RQ4** since

treating item features as side information fails to build a connection between the two source of information from different feature spaces. Instead, we formulate it as a regression problem where we input item features and output ratings. We establish the connections between ratings and item features by estimating similarity functions. These functions measure similarity among items based on item features. The learning of these functions is supervised by rating information. Therefore, the calculated similarities are domain-specific, which can be utilized for top-N recommendation. We study both global and local functions to comprehensively measure item similarities.

A typical example in need of heterogeneous information integration can be found in academic paper recommendations. If we look at the recommender system of ScienceDirect,[2] where a weekly email with recommended papers is sent to users, we can see that the email newsletter displays the title, venue (journal), authors, and publication date of each recommended paper. On clicking on a recommendation, the user is also linked to the paper on ScienceDirect. The system then logs on which recommendation(s) the user clicks. Since the ScienceDirect paper recommender was released, an increasing number of users have signed up. It is especially challenging to make recommendations for these new users due to the lack of ratings, which has led to the following research question:

RQ5 Can we address the challenge of recommending papers to cold-start users by effectively utilizing available heterogeneous information?

To answer **RQ5**, we propose a hybrid reranking model that combines paper content and user behavior to rerank candidate paper recommendations generated by the ScienceDirect recommender. First, we propose several content-based measures that are derived from various paper aspects, such as word space similarity, and author similarity from an embedding space. Next, we use a joint matrix factorization to learn a mapping from a user's browsed articles on the search engine to a user's clicks on the recommendations, to alleviate the sparsity of the recommendation click data. We use a pairwise learning model to rerank the candidate paper recommendation, which eventually leads to better results in offline evaluations based on real email click data.

1.1.3 High-dimensional and heterogeneous information

While methods have been proposed to integrate heterogeneous information for top-N recommendation [108, 267], they inevitably face the challenge of high-dimensionality. Therefore, we seek to answer the following research question:

RQ6 How should we integrate high-dimensional and heterogeneous information for top-N recommendation?

To answer **RQ6**, we provide a generic method based on factorization machines (FMs), which models the interactions between features within one signal or among different signals. The model has the capability to capture the complex relationships of the heterogeneous information. To address the high-dimensionality, we conduct feature interaction selection (FIS), which selects relevant interactions and filters out irrelevant interactions. We propose to select the personalized feature interactions, which is

[2] `https://www.sciencedirect.com/`

shown to improve the accuracy of top-N recommendation significantly.

1.2 Main Contributions

In this section, we list theoretical, algorithmic and empirical contribution of the thesis. For each contribution, we list the chapter from which it originates.

1.2.1 Theoretical contributions

The theoretical contributions of this thesis come in the form of five models:

1. *Block-aware similarity regularization* (BSR): a novel regularization method for ICF which captures clustering property of items (Chapter 2). The proposed method
   - introduces a new type of regularization for ICF models, which theoretically guarantees the block-diagonal structure of item similarity matrix; and
   - reveals the connections of the block-diagonality constraint on the item similarity matrix to sparsity and transitivity.
2. *Projection regularized item similarity model* (Prism): a joint learning model for top-N recommendation and feature reduction on side information (Chapter 3). The proposed method
   - integrates locality preserving projection into item similarity models.
3. *Collective variational auto-encoder* (cVAE): a VAE-based recommendation method that utilizes high-dimensional and noisy side information to overcome rating sparsity (Chapter 4). The proposed method
   - provides a new network structure of VAEs to collectively learn user factors and feature embeddings.
4. *Local variational feature-based similarity model* (LVSM): a deep generative model that learns a global item similarity function and multiple local similarity functions to comprehensively understand user's behavior (Chapter 5). The proposed method
   - seamlessly integrate item-based collaborative filtering with user clustering and deep learning.
5. *Hybrid reranking model* (HRM): a hybrid reranking model that utilizes the metadata of academic papers and user interactions (Chapter 6). The proposed method encompasses
   - various measures for comparing paper similarity built on paper metadata; and
   - a behavioral model that integrates hybrid user behaviors for recommendation.
6. *Bayesian personalized feature interaction selection* (BP-FIS): a Bayesian variable selection (BVS) model to select personalized feature interactions in order to utilize high-dimensional and heterogeneous features for factorization machines (Chapter 7). The proposed method includes
   - a new prior distribution of Bayesian variable selection (BVS) for personalized feature interaction selection (P-FIS).

1.2.2 Algorithmic contributions

7. An alternating minimization algorithm to optimize block regularized similarity model (BSM) with theoretical guarantee of convergence (Chapter 2).
8. An alternating minimization algorithm to optimize projection regularized item similarity model (Prism), where each alternation has a closed-form solution (Chapter 3).

9. An efficient algorithm based on stochastic gradient variational Bayes (SGVB) that pretrains cVAE by high-dimensional side information and fine-tuning it by rating information (Chapter 4).
10. A variational expectation maximization (EM)-algorithm to efficiently optimize LVSM (Chapter 5).
11. An efficient algorithm based on SGVB with novel reparameterization tricks to train BP-FIS (Chapter 7).

1.2.3 Empirical contributions

12. (1) An empirical comparison of BSM and BFSM with other state-of-the-art top-N recommendation models. (2) Analysis of the impact of block-diagonal regularization on the performance of top-N recommendation. (Chapter 2)
13. An empirical comparison of Prism with other state-of-the-art top-N recommendation with side infomration models (Chapter 3).
14. (1) An empirical comparison of cVAE with other state-of-the-art VAE-based top-N recommendation models. (2) Analysis of the recommendation performance achieved by different methods when the number of recommended items grows (Chapter 4).
15. (1) An empirical comparison of local variational feature-based similarity model (LVSM) with other state-of-the-art item cold-start recommendation models. (2) Evaluation of the effect of modeling global and local similarities on the performance of recommendation. (3) Analysis the impact of the fraction of cold-start items and the sparsity of features on the performance of recommendation (Chapter 5).
16. (1) Evaluating whether Bayesian personalized feature interaction selection (BP-FIS) can improve the performance of FMs for top-N recommendation. (2) Analysis of the impact of embedding size and training interactions on the performance of BP-FIS. (3) A case study to showcase the explainability provided by BP-FIS (Chapter 7).

1.3 Thesis Overview

The thesis is organized into two parts: *high-dimensional side information* and *heterogeneous information* for top-N recommendation, as depicted in Figure 1.1.

The first part consists of three chapters. In Chapter 2, we propose to utilize high-dimensional rating information for top-N recommendation; in Chapter 3, we propose to utilize high-dimensional rating information for top-N recommendation; in Chapter 4, we further propose to harness noisy high-dimensional side information to enhance top-N recommendation.

The second part consists of three chapters. In Chapter 5, we propose a deep generative model for recommending top-N new items; in Chapter 6, we propose to utilize both paper content and user behavior to rerank the paper for recommendation; in Chapter 7, we propose to select personalized feature interactions for top-N recommendation with heterogeneous features.

Finally, in Chapter 8, we conclude the thesis and discuss limitations and future directions.

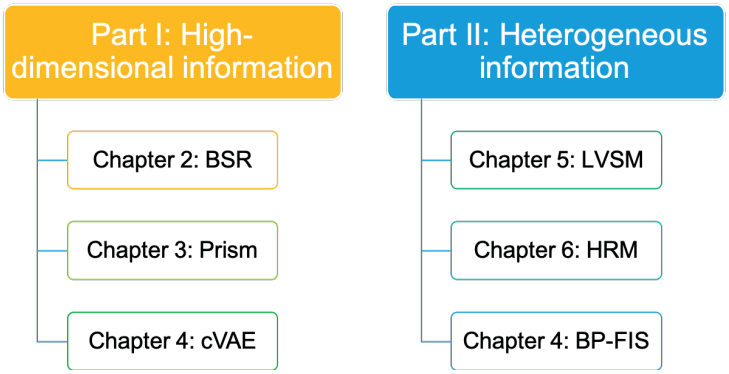


Figure 1.1: Thesis structure.

1.4 Origins

In this section, we list the publications each chapter is based on and explain the role of each author.

– **Chapter 2** is based on the following paper:
  - Yifan Chen, Yang Wang, Xiang Zhao, Jie Zou and Maarten de Rijke. 2019. Block-aware similarity regularizations for item-based collaborative filtering. *ACM Trans. Inf. Syst.* Under review.

  YC designed the model, ran the experiments and did most of the writing; JZ helped with running the experiments; YW, XZ and MdR contributed to the writing.
– **Chapter 3** is based on the following paper:
  - Yifan Chen, Xiang Zhao and Maarten de Rijke. 2017. Top-n recommendation with high-dimensional side information via locality preserving projection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '17). ACM, 985–988.

  YC designed the model, ran the experiments and did most of the writing; XZ helped with model design; XZ and MdR contributed to the writing.
– **Chapter 4** is based on the following paper:
  - Yifan Chen and Maarten de Rijke. 2018. A collective variational autoencoder for top-n recommendation with side information. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems* (DLRS '18@RecSys), 3–9.

  YC designed the model, ran the experiments and did most of the writing; MdR contributed to the writing.
– **Chapter 5** is based on the following paper:
  - Yifan Chen, Yang Wang, Xiang Zhao, Hongzhi Yin, Ilya Markov and Maarten de Rijke. 2019. Local variational feature-based similarity models for recommending top-n new items. *ACM Trans. Inf. Syst.* Under review.

  YC designed the model, ran the experiments and did most of the writing; YW and HY helped with model design; XZ, IM and MdR contributed to the writing.
– **Chapter 6** is based on the following paper:
  - Xinyi Li, Yifan Chen, Benjamin Pettit and Maarten de Rijke. 2019. Personalised reranking of paper recommendations using paper content and user behavior. *ACM Trans. Inf. Syst.*, 37, 3, Article 31.

YC and XL designed the model, ran the code for the experiments; all authors contributed to the writing.

– **Chapter 7** is based on the following paper:
  - Yifan Chen, Pengjie Ren, Yang Wang and Maarten de Rijke. 2019. Bayesian personalized feature interaction selection for factorization machines. In *Proceedings of the 42th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '19), 665–674.

  YC designed the model, ran the experiments and did most of the writing; PR helped with the model design; all authors contributed to the writing.

We also mention publications that contributed to the thesis indirectly.

- Yifan Chen, Xiang Zhao, Junjiao Gan, Junkai Ren and Yanli Hu. 2016. Content-based top-n recommendation using heterogeneous relations. In *Proceedings of the 27th Australasian Database Conference* (ADC '16), 308–320.
- Yifan Chen, Xiang Zhao, Jinyuan Liu, Bin Ge and Weiming Zhang. 2019. Learning to select user-specific features for top-n recommendation of new items. *Journal of Computer Science and Technology*. Under review.
- Jie Zou, Yifan Chen and Evangelos Kanoulas. 2020. Towards question-based recommender systems. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining* (WSDM '20). Under review.
- Yifan Chen, Xiang Zhao, Xuemin Lin, Yang Wang and Deke Guo. 2019. Efficient mining of frequent patterns on uncertain graphs. *IEEE Trans. Knowl. Data Eng.*, 31, 2, 287–300.

Part I

High-Dimensional Information

2

Clustering Items for Item-based Collaborative Filtering

In Chapter 1, we have set the scene for the research chapters. In this chapter, we propose block-aware similarity regularizations to cluster items for item-based collaborative filtering, which answers the following question asked in Chapter 1:

RQ1 How can we effectively perform item clustering for item-based collaborative filtering methods?

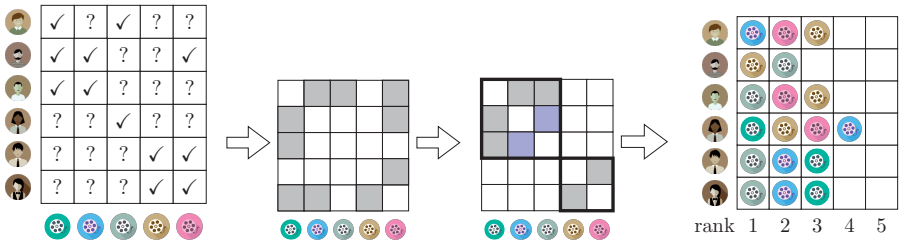
2.1 Introduction

Given a user profile with previous purchases or ratings, the top-N recommendation task is to *effectively* and *efficiently* help users identify the services and products that best fit their taste. A top-N recommendation algorithm should predict the recommendation scores for each user for each item in the pool of products and recommend the top-N items with the highest scores.

Collaborative filtering (CF) has been successfully used for top-N recommendations [192]. CF-based methods include latent space models [57] and neighborhood-based methods [63]. While latent space models can be utilized to generate lists of recommendations, they were originally designed for rating prediction tasks and are sub-optimal for top-N recommendations. Neighborhood-based methods (user-based or item-based) identify similar users or items; they have been shown to be better for the top-N recommendation problem [4, 63, 107, 169], and item-based methods outperform user-based methods [51].

Early *item-based collaborative filtering* (ICF) methods use statistical measures, e.g., Pearson coefficient or cosine similarity, to estimate item similarities [63, 197]. Recommendations by such heuristic-based approaches are fast but sacrifice performance. Sparse linear method (SLIM) [169] is a later proposal; it makes high-quality recommendations and ensures efficiency of recommendation by learning a *sparse* item similarity matrix from data. An inherent limitation of sparse linear method (SLIM) is that it can only model relations between items that have been co-rated by at least some users; their performance suffers when ratings are sparse. To address the issue, factored item similarity model (FISM) [107] factorizes the similarity matrix into low-

This chapter was published as [38].



(a) Rating matrix. (b) Without BSR. (c) With BSR. (d) Recommendation.

Figure 2.1: Example to show the effect of BSR. Figure 2.1(a) is a rating matrix from the movie recommendation domain, where rows and columns represent movies and users, respectively. If a user has rated a movie, the corresponding entry is marked with "✓", otherwise with "?"; Figure 2.1(b) is the item similarity matrix obtained by an ICF method without BSR, where non-zero entries are grayed; Figure 2.1(c) is the learned item similarity matrix with BSR when $c = 2$; Figure 2.1(d) is the sorted list of recommendations of unrated movies. The item similarity matrix in Figure 2.1(c) has a block-diagonal structure, with two blocks inside the rectangles with thick borders. Sparsity is achieved as off-block similarities are penalized. Transitive relations are also recovered within the block (the blue grids).

rank matrices so that *transitive* relations between items can be captured. However, the item similarity matrix generated by FISM is dense.

To ensure sparsity while enforcing low-rankness, the low-rank sparse linear method (lorSLIM) [47] introduces rank regularization to the item similarity matrix. The learned similarity matrix by lorSLIM is empirically shown to have a *block-diagonal* structure. This block-diagonal structure is important to top-N recommendations: it captures *latent item groups*. Latent item groups are subsets of items so that items contained in them are more similar to each other than to items from other subsets. Latent item groups are common to a wide spectrum of real-world collaborative filtering applications. For instance, in the movie domain, "Inception" would be similar to "Interstellar" as both are science fiction and suspense movies, whereas its degree of similarity with "Titanic" is low as the latter belongs to the categories of romantic and disaster movies. Low-rankness enforced by lorSLIM is an *indirect* way of pursuing a block-diagonal structure. Theoretically, the block-diagonal matrix can only be generated under restrictive conditions [151]. Practically, the learned item similarity matrix is far from being block-diagonal [73, 150]. Even if the similarity matrix is block-diagonal, we cannot require it to exactly have a pre-specified number of blocks.

An alternative way to capture latent item groups is to cluster items into sub-groups based on rating information. While clustering is prevalent in the context of CF [51, 238, 246, 249, 255, 268], it has been less studied for ICF. Recent work [6, 51, 52] studies user clustering for ICF. These authors cluster users into subgroups based on ratings and estimate a local ICF model for each cluster; they treat clustering and the estimation of local models as separate procedures.

In this chapter, we propose an approach called *block-aware similarity regularization* (BSR) to capture latent item groups for ICF methods. BSR encourages the learned item similarity matrix to be, or to be close to, a c-block diagonal, where c is the number of blocks. BSR integrates item clustering into the learning of item similarities, where in-block similarities are encouraged and off-block similarities are penalized.

The block-diagonal structure achieved by BSR is adaptively optimized during the training process. Besides, BSR can also encourage sparsity and transitivity in item similarities, which are crucial to the performance of top-N recommendations [107, 169]. BSR integrates item clustering into the learning of item similarities, where in-block similarities are encouraged and off-block similarities are penalized. The block-diagonal structure achieved by BSR is adaptively optimized during the training process. Besides, BSR can also encourage sparsity and transitivity in item similarities, which are crucial to the performance of top-N recommendations [107, 169].

Specifically, (1) we apply BSR to similarity models (SMs) [119, 169] and propose a block regularized similarity model (BSM); the effectiveness of BSM is theoretically guaranteed. (2) as SMs do not scale, we resort to feature-based similarity models (FSMs) [93, 107, 248], which are scalable; we extend BSR to *block-aware similarity dropout* (BSD) and use it to regularize feature-based similarity models (FSMs), based on which we propose a block regularized factored similarity model (BFSM). Figure 2.1 gives an illustrative example of how BSR works for ICF. By comparing BSMs and BFSMs with state-of-the-art SMs and FSMs we verify the effectiveness of BSR.

Our key technical contributions in this chapter are:

1. we propose BSR to capture the block-diagonal structure behind item similarities so as to improve ICF methods;
2. we apply BSR to SMs, whose effectiveness is theoretically guaranteed; we then extend BSR and propose BSD, which is used to regularize FSMs;
3. we conduct extensive experiments to assess BSR for ICF; BSM and BFSM are shown to outperform the state-of-the-art.

2.2 Related Work

2.2.1 Item-based collaborative filtering

Item-based collaborative filtering (ICF) methods are widely studied for the top-N recommendation task. Similarity models (SMs) that learn item similarities from data demonstrate strong performance. Ning et al. [169] have proposed sparse linear method (SLIM), which learns a sparse item similarity matrix. Low-rankness has been introduced to SLIM in order to recover transitive relations. To achieve low-rankness while ensuring sparsity, Cheng et al. [47] proposed low-rank sparse linear method (lorSLIM), which introduces a rank regularization term to SLIM. Kang et al. [110] improve lorSLIM with a better rank approximation.

Due to scalability issues with SMs, which require a quadratic number of parameters of item similarities, feature-based similarity models (FSMs) have been studied, which factorize item similarities [119]. Kabbur et al. [107] have proposed factored item similarity model (FISM) to factorize the item similarity matrix into two low-dimensional matrices. He et al. [93] improve FISM by applying the attention mechanism. Xue et al. [248] provide a more expressive FSM by modeling non-linear and higher-order relations among items.

A recent trend is to extend ICF by using auto-encoders. Wu et al. [239] learn to recover the rating matrix through denoising auto-encoder. Liang et al. [141] introduce variational auto-encoder for top-N recommendations. The auto-encoders are item-

side: they encode from and decode to user rating vectors of all items, which can be regarded as a generalization of ICF. However, the recommendations generated by these models have limited interpretability. Similar to FSMs, they also fail to achieve sparsity.

Other ICF methods consider different aspects to improve top-N recommendations. Ning et al. [170] and Chen et al. [43] utilize side information to overcome rating sparsity. Kang et al. [109] and Hu et al. [99] leverage graphs to address rating sparsity for top-N recommendation. Wang et al. [236] and Zhao et al. [269] investigate ranking loss functions for top-N recommendation.

2.2.2 Local models

Clustering has been well studied for collaborative filtering models [21, 81, 124, 173, 246, 249, 268]. These methods cluster users or items based on user ratings into subgroups and estimate a local model for each cluster. Results from all subgroups are aggregated to produce recommendations. Christakopoulou et al. [52] propose local latent factor models, where the assignments of the users to subsets are constantly updated. Wang et al. [232] introduce a probabilistic model to cluster items as topics. Wu et al. [238] propose a mixture model to infer memberships of users or items to subgroups. Lee et al. [123] describe an iterative way for estimation where first the latent factors representing the anchor points are estimated and then based on the similarities of observed entries to the anchor points, the latent factors are re-estimated.

A few publications specifically investigate clustering for ICF methods. Christakopoulou et al. [51] explore user subsets to learn user-specific local SMs, which is combined with a global SM. Al-Ghossein et al. [6] study online recommendation, where a user's membership is adaptively updated during incremental learning. However, these models only investigate user subsets rather than item groups. Clustering and the estimation of local models in these methods are also treated as separate tasks.

Unlike these methods, we propose to cluster items for ICF. We introduce block-aware similarity regularization (BSR) to encourage a block-diagonal structure to ICF methods, which embeds the clustering into the learning.

2.2.3 Subspace clustering

Learning block-diagonal representations has originally been studied for subspace clustering [73, 150, 243]. block-diagonal representation (BDR) [150] learns a block-diagonal representation matrix by utilizing block-diagonal regularization. While BDR can be utilized to learn a block-diagonal item similarity matrix, it fails to produce desirable item similarities for the top-N recommendation task. In this chapter, we apply BSR to SMs and propose a block regularized similarity model (BSM). BSM improves over BDR for top-N recommendations in the following manner: a) BSM adds a similarity constraint to overcome the negative effect on top-N recommendation caused by BSR (discussed in § 2.4.3); b) BDR introduces an intermediate term to facilitate optimization; the item similarity matrix generated by BDR has a certain bias due to the use of the intermediate term; in comparison, we directly penalize the item similarity matrix by BSR to avoid the bias; c) BSM can capture transitive relations within blocks, which is crucial to the performance of top-N recommendation (as discussed in § 2.4.4 below).

2.3 Notation

We first introduce our notation. All vectors are column vectors and represented by bold lowercase letters (e.g., $\boldsymbol{x}$). All matrices and constants are represented by uppercase letters (e.g., X) and Greek letters (e.g., α), respectively. Given a matrix X, x_{ij} represents the entry at the i-th row and j-th column. $\|X\|_1 = \sum_{i,j} |x_{ij}|$ and $\|X\|_F = (\sum_{i,j} x_{ij}^2)^{1/2}$ are the ℓ_1-norm and ℓ_F-norm of X, respectively. We write I to denote the identity matrix.

We use m and n to denote the number of users and items, respectively. We write $R \in \mathbb{R}^{m \times n}$ for user ratings, either explicit or implicit; The item similarity matrix is denoted by $S \in \mathbb{R}^{n \times n}$, where s_{ij} represents the similarity between item i and j. Given S, ICF methods predict the score of user u to the target item i by:

$$\tilde{r}_{ui} = \sum_{j \in \mathcal{R}_u^+} s_{ji}, \tag{2.1}$$

where $\mathcal{R}_u^+$ indicates the set of items rated by user u.

2.4 Block-Aware Regularizations

In this section, we first propose a regularization to achieve block-diagonality in § 2.4.1. We then apply it to SMs and introduce BSM in § 2.4.2. We discuss the effect of the similarity constraint and analyze the theoretical properties of BSM in § 2.4.3 and § 2.4.4, respectively.

2.4.1 Block-aware similarity regularization

We recall basic results from spectral graph theory [53]. We define the Laplacian matrix of S, denoted by L_S, as:

$$L_S = \text{Diag}(A\mathbf{1}) - A, \tag{2.2}$$

where $A = \frac{S+S^T}{2}$. $\text{Diag}(\boldsymbol{x})$ forms a diagonal matrix from $\boldsymbol{x}$ with its i-th element on the diagonal being x_i. It is easy to see that L_S is positive semidefinite as $\boldsymbol{x}^T L_S \boldsymbol{x} \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n$. We recall the following theorem to capture the connection between the Laplacian matrix and clusters of items.

Theorem 1 ([164]). *Let S be an item similarity matrix. The multiplicity c of the eigenvalue 0 of the Laplacian matrix L_S is equal to the number of connected components of the graph underlying S.*

Theorem 1 says that if $\text{rank}\,(L_S) = n - c$, then S provides an ideal assignment for items by partitioning items into c groups. To capture latent item groups, we can require the item similarity matrix S learned by ICF methods to follow this rank constraint, in order to learn S with a c-block structure. However, the rank constraint brings great difficulty for optimization. Besides, having exactly c blocks is not always desirable for S, as in many cases, item groups are not non-overlapping. Instead, we introduce regularization to S, in order to enforce the rank of L_S, in place of the rank constraint.

We first recall Ky Fan's Theorem [72]:

$$\sum_{i=1}^{c}\sigma_i = \min_{F}\sum_{i,j}^{n}\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 s_{ij}, \text{ such that } F\in\mathbb{R}^{n\times c}, F^TF=I, \tag{2.3}$$

where σ_i denotes the i-th smallest eigenvalue of L_S; F is an auxiliary matrix and $\boldsymbol{f}_i$ is the i-th row of F. As L_S is positive semidefinite, e.g., $\sigma_i \geq 0$, we can enforce $\sum_{i=1}^{c}\sigma_i$ to be zero, so as to achieve the c block-diagonal structure. Thus, the block-aware similarity regularization (BSR) is given as:

$$\|S\|_B = \min_{F^TF=I}\sum_{i,j}^{n}\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 s_{ij}, \tag{2.4}$$

2.4.2 Block regularized similarity model

We regularize SMs by BSR and propose a block regularized similarity model (BSM):

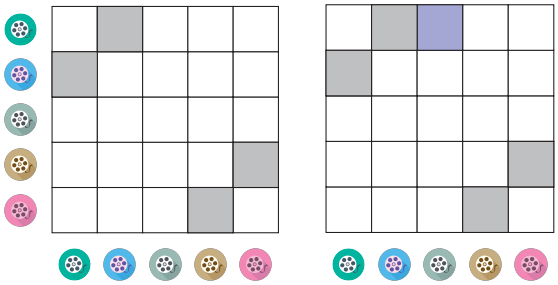
$$\begin{aligned}\arg\min_{S,F}\quad & \frac{1}{2}\|R-RS\|_F^2+\alpha\|S\|_1+\frac{\beta}{2}\|S\|_F^2+\lambda\|S\|_B,\\ \text{such that}\quad & \forall i,j, s_{ij}\geq 0, s_{ii}=0, \sum_{i=1}^{n}s_{ij}=1,\\ & F\in\mathbb{R}^{n\times c}, \text{ and } F^TF=I.\end{aligned} \tag{2.5}$$

Let us explain BSM. The first term in the objective forms the loss function by ICF, as given in Eq. (2.1). The constraint $s_{ii}=0$ is added to Eq. (2.5) to avoid the trivial solution that $S=I$. Requiring s_{ij} to be non-negative is to learn meaningful similarities. ℓ_1-norm regularization encourages sparsity to S, as suggested by [169]. Together with the ℓ_F-norm regularization leads to an elastic net problem [279].

Additional remarks of Eq. (2.5) are discussed as follows: (1) We introduce BSR to S. The structure of S is close to c block-diagonal if λ is large enough. (2) As the block-diagonal structure is already sparse (as discussed in § 2.4.4 below), in practice, we can fix α to be a small value just to avoid 0's in the denominator during optimization (see Eq. (2.13)). (3) We further require the column summation of S to be 1 to overcome the negative effect caused by BSR, which is crucial to the performance (detailed in § 2.4.3).

2.4.3 Overcoming the negative effect

Note that in Eq. (2.5) we include a similarity constraint on S, requiring the column summation of S to be 1 ($\sum_{i=1}^{n}s_{ij}=1$). This similarity constraint helps to overcome a negative effect brought by BSR. To be more specific, if c is larger than the intrinsic number of latent item groups, some lonely items that do not show much affiliation with any of the groups could be sacrificed. Recall the example rating in Figure 2.1, without the similarity constraint, if we set $c=3$, the third column of S is learned to be all-zero, as shown in Figure 2.2(a). This is justifiable as the block-diagonal regularization tries to encourage three blocks, where the third item is itself a block, so that every off-diagonal entry within the third column is encouraged to be zero. While



(a) Without similarity constraint (b) With similarity constraint

Figure 2.2: Block-aware similarity regularization with $c = 3$.

this conforms to three blocks, it is not desirable for recommendation purposes as the movie in gray cannot be recommended. Similarity regularization can avoid all-zero columns by encouraging the summation of columns to be 1. As shown in Figure 2.2(b), s_{13} is preserved to be non-zero.

2.4.4 Connection to sparsity and transitivity

Sparsity. BSM can achieve sparsity as the block-diagonal structure is also sparse. To see this, we provide Theorem 2 to reveal the connection.

Theorem 2. *BSR is a weighted ℓ_1-norm regularization if $S \geq 0$.*

Proof. Suppose $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are the eigenvectors for L_S, which are in ascending order of eigenvalues. For all i, j, if $i = j$, we have: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = 0$, else we have $\boldsymbol{x}_i^T \boldsymbol{x}_j = 0$ and $\boldsymbol{x}_i^T \boldsymbol{x}_i = 1$, and we can derive $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2$ as:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = \boldsymbol{x}_i^T \boldsymbol{x}_i + \boldsymbol{x}_j^T \boldsymbol{x}_j - 2 \cdot \boldsymbol{x}_i^T \boldsymbol{x}_j = 2.$$

As we require $S \geq 0$, we can rewrite the block-diagonal regularization as:

$$\|S\|_B = \sum_{i,j}^{n} \|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 s_{ij} = \sum_{i,j}^{n} |d_{ij} s_{ij}| = \|D \circ S\|_1,$$

where D is a Euclidean distance matrix with $d_{ij} = \|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2$. Therefore, BSR is a weighted ℓ_1-norm regularization and d_{ij} can be written as:

$$d_{ij} = \begin{cases} 2 - \sum_{l=c+1}^{n} (x_{il} - x_{jl})^2, & i \neq j \\ 0, & \text{otherwise.} \end{cases} \quad \square$$

Theorem 2 shows that BSR can encourage sparsity of S. As d_{ij} is non-decreasing with the growth of c, when $c \to n$, we have $d_{ij} \to 2$, so that the block-diagonal regularizer has a similar effect as ℓ_1-norm regularization.

Transitivity. We further show that BSM can also capture transitive relations by providing Theorem 3. We focus on the following problem:

$$S^* = \arg\min_S \|S\|_B + \frac{\beta}{2} \|S\|_F^2, \text{such that} \sum_{i=1}^{n} s_{ij} = 1. \tag{2.6}$$

As shown by Nie et al. [168], the closed-form solution for Eq. (2.6) is:

$$s_{ij}^* = \max\left\{-\frac{d_{ij}}{\beta} + \theta, 0\right\}, \tag{2.7}$$

where θ is the Lagrangian multiplier, which is a constant.

Theorem 3. *We define the relation $\sim$. We say $i \sim j$ if $s_{ij}^* > 0$. Given the optimal solution S^* that complies with the similarity constraint w.r.t. block-diagonal regularization and ℓ_F-norm regularization, if $i \sim j$, $j \sim k$, and $s_{ij}^*, s_{jk}^* > \frac{3}{4}\theta$, then $i \sim k$.*

Proof. According to Eq. (2.7), as $s_{ij}^*, s_{jk}^* > 0$, we can have $q_{ij} = \beta(\theta - s_{ij}^*)$ and $q_{jk} = \beta(\theta - s_{jk}^*)$. As $s_{ij}^*, s_{jk}^* > \frac{3}{4}\theta$, we have $q_{ij}, q_{jk} < \frac{\beta\theta}{4}$. As Q is a Euclidean distance matrix, the triangle inequality holds: $\sqrt{q_{ik}} \leq \sqrt{q_{ij}} + \sqrt{q_{jk}}$. We have:

$$s_{ik}^* \geq -\frac{q_{ik}}{\beta} + \theta > \theta - \frac{1}{\beta}\left(2\sqrt{\beta\theta/4}\right)^2 = 0.$$

Thus, we have $i \sim k$. □

As shown by Theorem 3, transitive relations among items can be captured as long as the original relations are strong (i.e., the similarity is above a certain threshold). If S^* is sparse, then θ is small, and the similarity threshold is more likely to be met. Thus the transitivity can hold simultaneously with sparsity.

2.4.5 Optimization

The use of BSR for SMs brings in an additional variable F. Therefore, we introduce an alternating minimization algorithm to optimize BSM.

Fix S and update F. When fixing S, Eq. (2.5) is reduced to the following problem:

$$\arg\min_{F^TF=I} \mathrm{Tr}\left(F^T L_S F\right), \tag{2.8}$$

where L_S is the Laplacian matrix of S (see Eq. (2.2)). A closed-form solution for F can be obtained as the c eigenvectors corresponding to the c smallest eigenvalues of L_S.

Fix F and update S. Note that optimizing Eq. (2.5) is difficult due to the similarity constraint. We relax the problem in Eq. (2.5) by transforming the similarity constraint to a similarity regularization. The relaxation of the similarity constraint is not only necessary for efficiency reasons but also helpful for recommendation (§ 2.7.2).

$$\begin{aligned} \arg\min_{S} \quad & \frac{1}{2}\|R - RS\|_F^2 + \alpha\|S\|_1 + \frac{\beta}{2}\|S\|_F^2 + \\ & \lambda\sum_{i,j}^{n}\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 s_{ij} + \frac{\gamma}{2}\|\mathbf{1}^T S - \mathbf{1}^T\|_2^2, \\ \text{such that} \quad & \forall i, j, s_{ij} \geq 0, s_{ii} = 0. \end{aligned} \tag{2.9}$$

To optimize Eq. (2.9) with fixed F, we write J as shorthand for the objective function in Eq. (2.9). Due to the non-negative constraint on S, we apply the multiplicative

update method [122] to efficiently update S. The multiplicative update method is an iterative updating method, which ensures that during each iteration, the variables to be updated are non-negative. We can write the objective function of Eq. (2.5) as:

$$J = \frac{1}{2}\|R - RS\|_F^2 + \sum_{i,j}^{n} d_{ij}s_{ij} + \frac{\beta}{2}\|S\|_F^2 + \frac{\gamma}{2}\|\mathbf{1}^T S - \mathbf{1}^T\|_2^2, \tag{2.10}$$

where

$$d_{ij} = \alpha + \lambda\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2. \tag{2.11}$$

Note that we write $|s_{ij}| = s_{ij}$ as s_{ij} is ensured to be non-negative in each iteration. Then the partial derivative over S is:

$$\frac{\partial J}{\partial S} = R^T RS - R^T R + D + \beta S + \gamma\left(\mathbf{1}\cdot\mathbf{1}^T S - \mathbf{1}\cdot\mathbf{1}^T\right). \tag{2.12}$$

Applying the Karush-Kuhn-Tucker (KKT) first-order optimality conditions [54] to J, we derive

$$S \geq 0, \frac{\partial J}{\partial S} \geq 0, S \circ \frac{\partial J}{\partial S} = 0,$$

where $\circ$ is the element-wise multiplication between two matrices of the same dimension. This leads to the following update rule:

$$S \leftarrow S \circ \frac{\left[R^T R + \gamma\mathbf{1}\cdot\mathbf{1}^T\right]}{\left[(R^T R + \gamma\mathbf{1}\cdot\mathbf{1}^T)S + D + \beta S\right]}, \tag{2.13}$$

where $\frac{[\cdot]}{[\cdot]}$ denotes the element-wise matrix division operator. We omit the proof of convergence due to the limit of space.

Complexity of BSM. During training, when optimizing F, we only need the c eigenvectors corresponding to the c smallest eigenvalues, the complexity of which is $O(n^2c)$. Packages like ARPACK[1] provide additional benefit to calculate the eigenvectors when S is sparse. For optimizing S, the biggest source of complexity lies in the matrix multiplication. Fortunately, as $R^T R$ and S are both sparse, the complexity of multiplying the two matrices can be reduced to $O(nz_1z_2)$ [200], where z_1, z_2 are the average number of non-zeros in the rows of $R^T R$ and in the columns of S. The overall complexity is $O(n^2c + nz_1z_2)$.

2.5 Scalable Block-Aware Regularization

BSM fails to scale to datasets with large numbers of items, which is unfortunately common in real-world applications. The scalability issue of BSM comes both from BSR and SMs. To address the issue, we optimize F offline (§ 2.5.1), rather than learn adaptively. We also extend the regularization to a dropout (§ 2.5.2), which can be used to regularize FSMs. We then apply the dropout to FSM and propose a block regularized factored similarity model (BFSM) in § 2.5.3.

[1] `https://www.caam.rice.edu/software/ARPACK/`

2.5.1 Offline computation of F

In order to compute F offline, we need an initial laplacian matrix L, according to Eq. (2.8). Rather than calculating L based on S, which is unknown initially, we construct L offline based on the rating matrix R [53]. We take F as the representations of items. The *manifold assumption* [20] states that if $\boldsymbol{r}_i$ (all ratings of item i) and $\boldsymbol{r}_j$ are close in the intrinsic geometry of rating, the representations of the two items $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ are also close. The manifold assumption, which is widely used to derive graph regularizations [28, 198], plays an essential rule in developing various kinds of algorithms. In practice, the data manifold is usually unknown. Work on spectral graph theory [53] and manifold learning theory [19] has demonstrated that the local geometric structure can be effectively modeled through a nearest neighbor graph on a scatter of data points.

The nearest neighbor graph is defined as follows: consider a graph with n vertices where each vertex corresponds to an item. By defining the distance between item i and j as $\|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2^2$, the edge weights, denoted by a_{ij}, can be binarized (1 if $\boldsymbol{r}_i$ is in the nearest neighbor of $\boldsymbol{r}_j$ or $\boldsymbol{r}_j$ in the nearest neighbor of $\boldsymbol{r}_i$, 0 otherwise). Given edge weights A of the graph, the laplacian matrix L can be calculated based on Eq. (2.2) and F can be calculated as the c eigenvectors corresponding to the c smallest eigenvalues of L.

2.5.2 Block-aware similarity dropout

Compared with SMs, feature-based similarity models (FSMs) that factorize the similarity matrix S into two low-rank matrices $P \in \mathbb{R}^{n\times k}$ and $Q \in \mathbb{R}^{n\times k}$ are scalable [119]. However, BSR cannot regularize FSMs directly. Regularizing $S = PQ^T$ by BSR is not effective as item similarities are correlated due to the factorization. Instead, we resort to *Dropout* [213], the regularization technique that widely utilized for deep neural network (DNN). Dropout works by adding multiplicative noise to the input of layers of DNN. To regularize $S = PQ^T$ by dropout, we add the noise to S:

$$S = (PQ^T) \circ \xi, \tag{2.14}$$

where $\xi \in \mathbb{R}^{n\times n}$ stands for the noise and $\circ$ is the element-wise multiplication between matrices. Hinton et al. [97] proposed to draw the elements of ξ from a Bernoulli distribution. Later it was shown that a continuous distribution, such as a Gaussian with the same mean and variance works as well [213]. As proved in § 2.4.4, BSR is a weighted version of ℓ_1-norm regularization, which can encourage sparsity. Therefore, we take a Bernoulli distribution for ξ, which is shown to encourage sparsity [213].

To be aware of the block-diagonal structure, we propose a block-aware similarity dropout (BSD), which assumes ξ_{ij} to follow the Bernoulli distribution with respective dropout rates:

$$\xi_{ij} \sim Bernoulli\left(\sigma(-\lambda d_{ij})\right),$$

where d_{ij} is calculated by Eq. (2.11) and λ is a parameter to control the effect of block-aware similarity dropout (BSD); ξ_{ij} is used to indicate whether the corresponding item similarity s_{ij} is in-block ($\xi_{ij} = 1$) or off-block ($\xi_{ij} = 0$). The larger the value of d_{ij}, the less likely s_{ij} is within a block and should be zero.

2.5.3 Block regularized factored similarity model

Given BSD, we formulate a block regularized factored similarity model (BFSM) by the following generative process:

1. For each item i, draw item factor $\boldsymbol{p}_i \sim \mathcal{N}(0, \beta^{-1}I)$;
2. For each item j, draw item factor $\boldsymbol{q}_j \sim \mathcal{N}(0, \beta^{-1}I)$;
3. For each item pair (i, j),
   (a) draw the selection variable $\xi_{ij} \sim Bernoulli(\sigma(-\lambda d_{ij}))$;
   (b) calculate item similarity $s_{ij} = \xi_{ij} \cdot f_\theta(\boldsymbol{p}_i, \boldsymbol{q}_j)$;
4. For each user-item pair (u, i), draw $r_{ui} \sim \mathcal{N}(\tilde{r}_{ui}, 1)$.

We briefly explain the generative procedure. We use Gaussian priors for $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$; $f_\theta(\cdot)$ is a function parameterized by θ. Depending on the definition of $f_\theta(\cdot)$, different ICF models can be formulated [93, 107, 248]. For example, FISM defines $f_\theta(\cdot)$ directly as the dot product: $f_\theta(\boldsymbol{p}_i, \boldsymbol{q}_j) = \boldsymbol{p}_i^T \boldsymbol{q}_j$. Following the generation procedure, we can write the block regularized factored similarity model (BFSM) as follows:

$$\underset{P,Q}{\arg\min} \quad \frac{1}{2} \sum_{u,i \in R} \|r_{ui} - \tilde{r}_{ui}\|_2^2 + \frac{\beta}{2} \left(\|P\|_F^2 + \|Q\|_F^2 \right), \tag{2.15}$$

where $\tilde{r}_{ui} = \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \xi_{ij} \cdot f_\theta(\boldsymbol{p}_i, \boldsymbol{q}_j)$. Eq. (2.15) can be optimized via mini-batch stochastic gradient descent (SGD) [93, 107, 248].

Complexity of BFSM. As we optimize F offline for BFSM, we only analyze the complexity of optimizing P and Q. BFSM is optimized via SGD. The complexity of evaluating a prediction $\tilde{r}_{ui}$ is $O(|\mathcal{R}_u^+|\, k)$, which is scalable.

2.6 Experimental Setup

In this section, we introduce the experimental setups.

2.6.1 Research questions

Our research questions are:

RQ1.1 To what extent can BSR improve the performance of ICF methods for top-N recommendation?
RQ1.2 How does BSR affect the performance of top-N recommendation?
RQ1.3 What is the impact of BSR on the training procedure of ICF methods?

2.6.2 Datasets

We evaluate the performance of BSR on five benchmark datasets.

– *Amazon*:[2] A dataset based on the Amazon product catalogue [157]; we select one of the categories, Sports & Outdoors, which contains transactions between different product items and users indicated with multivariate rating values.
– *BookX*:[3] A subset of the Book-Crossing dataset, which was collected by [277] from the Book-Crossing community.

[2] http://jmcauley.ucsd.edu/data/amazon/
[3] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Table 2.1: Descriptive statistics of the datasets.

Name	*#User*	*#Item*	*#Rating*	*Density*
Amazon	5,653	11,944	86,149	0.13%
BookX	5,671	5,367	86,354	0.28%
Yahoo	7,594	8,641	106,593	0.16%
MovieLens	6,040	3,706	1,000,209	4.47%
Pinterest	55,187	9,916	1,500,809	0.27%

#User, #Item and #Rating denotes the number of users, items and ratings, respectively. Density is calculated as #Rating/(#User$\times$#Item).

– *Yahoo*:[4] A small sample of the Yahoo!Movies community's preferences for various movies, rated on a scale from A+ to F.

For these three datasets, we filter out users with fewer than 10 ratings and items that are rated by fewer than 5 users. Following the common setting for implicit feedback, we binarize the ratings. We also adopt two datasets for a fair comparison against [93].

- *MovieLens*:[5] The MovieLens 1M Dataset released by the GroupLens research project;
- *Pinterest*: One of the largest social curation networks. The implicit feedback data is constructed by [80] for evaluating content-based image recommendation.

Table 2.1 lists descriptive statistics of the datasets.

2.6.3 Methods used for comparison

To assess the performance of BSR, we apply BSR to three state-of-the-art ICF methods, including one SM and two FSMs: SLIM [169], FISM [107] and NAIS [93], respectively denoted by SLIM-$\mathcal{B}$, FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$.

Baselines. We compare SLIM-$\mathcal{B}$, FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$ with the following baselines:[6]

– *Item-based k-nearest-neighbor* (item$k$NN) [63]: An early ICF method that heuristically computes item similarities. We choose cosine as the similarity function and apply shrinkage to the similarities.
– *Sparse linear method* (SLIM) [169]: A SM that learns a sparse item similarity matrix.
– *Factored item similarity model* (FISM) [107]: A FSM that factorizes item similarity matrix into two low-rank matrices.
– *Neural attentive item similarity model* (NAIS) [93]: A Neural-based FSM that utilizes an attention mechanism.
– *Bayesian personalized ranking* (BPR) [188]: A ranking/retrieval criteria-based method. We train a latent space model with the pair-wise loss function.

[4] `https://webscope.sandbox.yahoo.com/catalog.php`
[5] `https://grouplens.org/datasets/movielens/`
[6] We exclude LorSLIM [47], the low-rank sparse linear model, from our experimental comparisons. We failed to generate a set of reasonable recommendations using LorSLIM on all datasets and we were also unable to reproduce the results obtained using LorSLIM as reported in [47]. The source code of LorSLIM on MovieLens with 100k ratings (ML-100k) is evaluated with 336 items, rather than all 1,682 items. For a fair comparison, we evaluate SLIM-$\mathcal{B}$ in the same setting, which provides much better results than reported in their paper, i.e., HR@10 $= 0.574$, ARHR@10 $= 0.265$ against HR@10 $= 0.397$, ARHR@10 $= 0.207$. A similar issue exists with the method proposed in [110]. Therefore, we exclude the two methods from our experiments.

– *Pure singular-value-decomposition* (pureSVD) [57]: A latent space model designed for top-N recommendation.
– *Weighted regularized matrix factorization* (WRMF) [100]: A latent space model specially for implicit datasets.
– *Multinomial variational auto-encoder* (mVAE) [141]: A state-of-the-art non-linear method for top-N recommendation. It utilizes variational autoencoder and assume multinomial likelihood function.
– LocalSLIM: SLIM with item clustering. The implementation is similar to [51] but clustering items instead.

Implementation details. We use LibRec [88] to run the experiments for item$k$NN, SLIM, BPR and WRMF. We use the source code implementation in [93] for FISM and NAIS and that in [141] for mVAE. As reported in [93], NAIS-concat and NAIS-prod differ slightly in performance while NAIS-prod shows better convergence; we compare with NAIS-prod only in the experiments. We implement SLIM-$\mathcal{B}$ by alternating minimization of Eq. (2.8) and Eq. (2.13). We implement FISM-$\mathcal{B}$ by optimizing Eq. (2.15) through SGD. We implement NAIS-$\mathcal{B}$ on top of NAIS-prod. We optimize FISM, NAIS, FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$ with the same point-wise RMSE loss using a Adagrad learner with a learning rate of 0.01. We also implement PureSVD and localSLIM.

Parameters. The parameters of all methods are explored within the parameter space. We select parameters based on the best performance in terms of HR@10 on the validation set. We use all neighbors for item$k$NN, which generally leads to the best results [93]. We fix $\alpha = 0$ for FISM and NAIS as it has been empirically shown to lead to the best result [93]. The number of latent dimensions k of FISM, NAIS, BPR, PureSVD and WRMF are selected from $8, 16, 32, 64$. The regularization parameters are selected from $0.01, 0.1, 1, 10, 100$.

2.6.4 Evaluation methodology

We evaluate the methods using leave-one-out cross-validation (LOOCV): we hold out the latest interaction of each user as the test data and uses the remaining interactions as training set. The validation set consists of a randomly drawn interaction for each user from the training set. This evaluation method is widely utilized for top-N recommendations [17, 92, 188].

We use hit rate (HR) and average reciprocal hit-rank (ARHR) [63, 107] to evaluate the performance. ARHR is a weighted version of HR, which takes the ranking position of the test item i in the recommended list into account. Note that HR and ARHR can be regarded as Recall and mean reciprocal rank (MRR) when evaluating using LOOCV, respectively. We also use normalized discounted cumulative gain (NDCG) [93] as evaluation metric for a fair comparison with NAIS.

2.7 Experimental Results

We answer the research questions listed in § 2.6.1 based on the experimental results.

Table 2.2: Top-N recommendation from all items unrated by users.

Method	Amazon		BookX		MovieLens		Yahoo	
	HR@10	ARHR@10	HR@10	ARHR@10	HR@10	ARHR@10	HR@10	ARHR@10
item$k$NN [63]	0.0663	0.0251	0.0908	0.0409	0.1620	0.0662	0.3368	0.1611
BPR [188]	0.0603	0.0238	0.1047	0.0520	0.2353	0.0977	0.3460	0.1675
PureSVD [57]	0.0475	0.0171	0.0920	0.0504	0.2142	0.0920	0.2385	0.1029
WRMF [100]	0.0666	0.0267	0.1126	0.0554	0.2339	0.0967	0.3458	0.1574
localSLIM	0.0622	0.0040	0.1001	0.0072	0.1950	0.0120	0.3506	0.0248
mVAE [141]	0.0400	0.0144	0.0813	0.0388	0.2318	0.0926	0.3745	0.1762
SLIM [169]	0.0727	0.0300	0.1201	0.0666	0.2454	0.1073	0.4010	0.2086
FISM [107]	0.0586	0.0202	0.0577	0.0207	0.2167	0.0874	0.3198	0.1526
NAIS [93]	0.0547	0.0173	0.0820	0.0329	0.2316	0.0878	0.3304	0.1533
SLIM-$\mathcal{B}$	↑**0.0778****	↑**0.0329****	↑**0.1300****	↑**0.0678**	↑**0.2547****	↑**0.1104****	↑**0.4045**	↑**0.2089**
FISM-$\mathcal{B}$	↑0.0605	↑0.0209	0.0577	↑0.0228****	↓0.2150	↑0.0877	↑0.3204	↑0.1556
NAIS-$\mathcal{B}$	↑0.0548	↑0.0199****	↑0.0869	↑0.0369****	↓0.2303	↑0.0915*	↓0.3289	↓0.1521

The best result is shown in **boldface**, the second best is underlined and the third best is indicated with a dashed line. We indicate with↑if the method with BSR improves the performance, otherwise↓. We conducted two-sided tests for the null hypothesis that methods with/without BSR have identical average values. We attach asterisks if the improvement is statistically significant: * if $p < 0.05$ and ** if $p < 0.01$.

Table 2.3: Top-N recommendation from 100 candidate items. Embedding size of FSMs is fixed as 16.

Method	MovieLens		Pinterest	
	HR@10	NDCG@10	HR@10	NDCG@10
SLIM [169]	0.6864	0.4247	0.8620	0.5588
FISM [107]	0.6553	0.3851	0.8752	0.5522
NAIS [93]	0.6804	0.4055	0.8790	0.5604
NAIS$_{pre}$ [93]	**0.6969**	0.4194	0.8844	**0.5722**
SLIM-$\mathcal{B}$	↑0.6964**	↑**0.4296****	↑0.8650**	↑0.5604
FISM-$\mathcal{B}$	↑0.6712**	↑0.3983**	↑0.8799**	↑0.5560**
NAIS-$\mathcal{B}$	↑0.6939	↑0.4183	↑0.8834	↑0.5663
NAIS-$\mathcal{B}_{pre}$	↓0.6949	0.4194	↑**0.8845**	↓0.5701

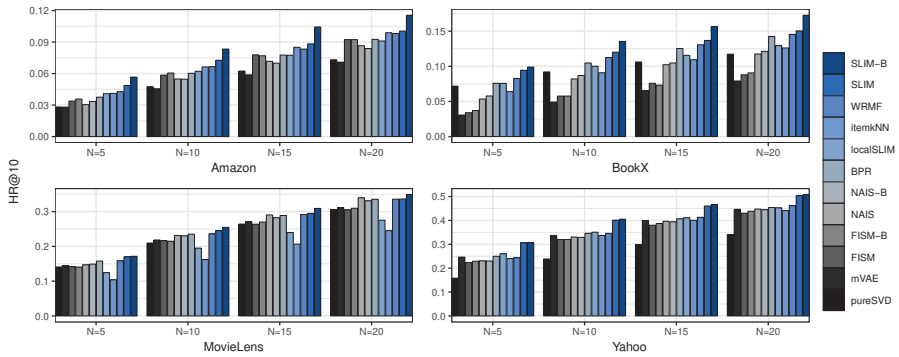


Figure 2.3: Top-N recommendation results for different values of N.

2.7.1 RQ1.1: Top-N recommendation performance

To answer RQ1.1, we compare SLIM-$\mathcal{B}$, FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$ with SLIM, FISM and NAIS, respectively. We also compare them with other baselines. We report results for $N = 10$ in Table 2.2 and 2.3.

Table 2.2 shows the results on the Amazon, BookX, MovieLens and Yahoo datasets by all methods,[7] where the recommended items to each user are constructed from *all the items* unrated by the user. Overall, SMs outperform other methods on all datasets and SLIM-$\mathcal{B}$ significantly improves upon SLIM.

We discuss the results per dataset: a) the Amazon dataset has the largest number of items, the smallest number of users, and the most sparse implicit feedback. Therefore, the overall accuracy for the Amazon dataset is low. Besides SLIM and SLIM-$\mathcal{B}$, WRMF performs best, as it is specifically designed for implicit datasets. However, the improvement of WRMF over item$k$NN is relatively modest. WRMF is outperformed by SLIM. Applying BSR improves all the three implemented methods, where the improvements in SLIM-$\mathcal{B}$ w.r.t. HR@10 and ARHR@10, and in NAIS-$\mathcal{B}$ w.r.t. ARHR@10 are significant. The effectiveness of capturing latent item groups is well confirmed in Amazon dataset. b) On the BookX dataset, results are similar to Amazon

[7] We do not show the performance on Pinterest here as methods implemented by Librec are not scalable.

as both datasets are implicit, while the overall results are better. SLIM-$\mathcal{B}$ improves over SLIM, especially w.r.t. HR@10 (by 8.24%), which is the most conspicuous among all datasets. The grouping effect of SLIM-$\mathcal{B}$ is evident. BSR also boosts FISM and NAIS. The improvement by SLIM-$\mathcal{B}$ is significant w.r.t. HR@10 while the improvements by FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$ are significant w.r.t. ARHR@10. Seems that BSR can help FSMs provide better ranking of items within recommendations. c) The overall performance on the MovieLens dataset is high since this dataset has the least sparse ratings. While SLIM-$\mathcal{B}$ marginally improves over SLIM, the improvement is significant. While BSR improves FISM and NAIS w.r.t. ARHR@10, it fails to improve w.r.t. HR@10. d) The superiority of SMs is clearly visible on the Yahoo dataset. SLIM outperforms mVAE substantially, and SLIM-$\mathcal{B}$ improves over SLIM. The grouping effect on the Yahoo dataset shows similarity with the MovieLens; for both datasets the best performance is achieved when $\lambda = 10$ for SLIM-$\mathcal{B}$ vs. with $\lambda = 100$ on Amazon and BookX. The learned item similarity matrix has a less rigorous c-block structure.

To illustrate the gains achieved by BSR over competing approaches, we show the HR score of all algorithms for different values of N (i.e., $5, 10, 15, 20$) on Amazon, BookX, MovieLens and Yahoo datasets in Figure 2.3, where similar results have been revealed as in Table 2.2.

For a fair comparison with NAIS, we follow exactly the same experimental setup as [93] to construct the candidate items for each user that has *100 items* (99 sampled unrated items together with the held-out test item). We then take the results from their paper for comparison. We report HR@10 and NDCG@10. To compare with NAIS with pretraining, denoted by NAIS$_{pre}$, we also add pretraining for NAIS-$\mathcal{B}$, denoted by NAIS-$\mathcal{B}_{pre}$. Table 2.3 shows the results. Overall, BSR improves the performance of SLIM, FISM and NAIS. However, with pretraining BSR fails to show the effectiveness of grouping. It seems that pretraining by FISM can bias the grouping effect. a) Similar to the results in Table 2.2, SLIM-$\mathcal{B}$ shows competitive results, though been outperformed by NAIS$_{pre}$ w.r.t. HR@10 on MovieLens. Except for NAIS-$\mathcal{B}_{pre}$, applying BSR improves the performance of ICF, where SLIM-$\mathcal{B}$ and FISM-$\mathcal{B}$ show significant improvement. b) On Pinterest, the first time FSMs overall performs better than SMs. FISM-$\mathcal{B}$ and NAIS-$\mathcal{B}$ outperform FISM and NAIS, respectively. The best and second best results are both reached by NAIS or NAIS-$\mathcal{B}$ with pretraining.

Item groups exist in many real-world applications and BSR is able to capture the grouping property to improve top-N recommendation. SMs can benefit more from BSR than FSMs, due to the theoretical guarantees. While BSR boosts the performance of FISM, it may fail to improve NAIS due to the influence of attention.

2.7.2 RQ1.2: Impact of block-aware similarity regularizations

§ 2.7.1 demonstrated the effectiveness of BSR for improving the performance of top-N recommendation. In this section, we analyze the impact of BSR in detail. We focus on the discussion of SLIM-$\mathcal{B}$ where BSR has theoretical guarantees. We grid-search the parameter λ that controls BSR. We also grid-search γ which is essential to overcome the negative effect of BSR. We vary c from $10, 50, 100$ to 500. We visualize the results with heat maps in Figure 2.4, where λ is shown on the x-axis and γ on the y-axis.

For the Amazon dataset (Figure 2.4, first row), generally larger numbers of item groups are preferable. The block-structure shows its effectiveness as we need a larger

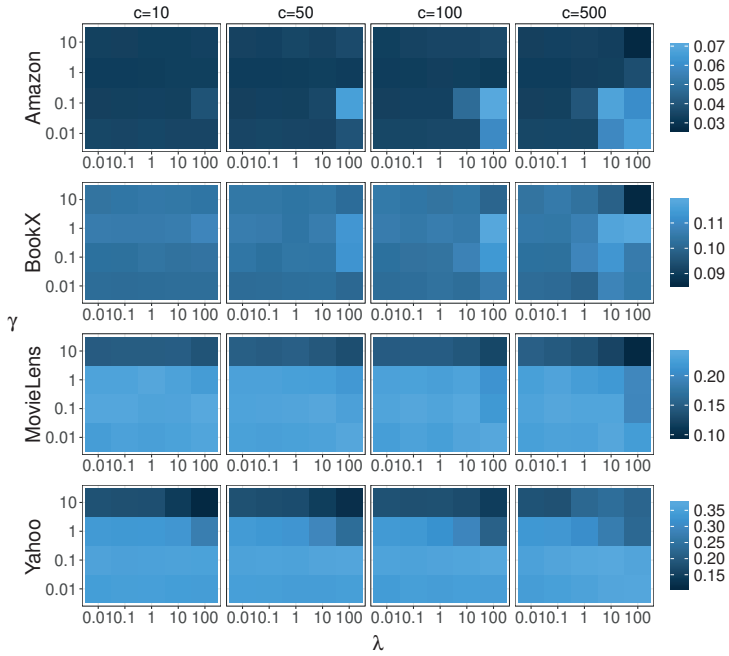


Figure 2.4: Impact of block-aware similarity regularizations. The color intensity corresponds to HR@10.

value for λ to achieve better performance. It is worth noting that similarity regularization is also important, as setting 0.1 is the best choice for γ. On the Amazon dataset, the performance of SLIM-$\mathcal{B}$ is sensitive to the parameters of BSR. Although a similar result is shown on the BookX dataset (Figure 2.4, second row), which also prefers larger number of latent item groups, SLIM-$\mathcal{B}$ is less sensitive to BSR on the BookX dataset. The similarity regularization is more effective on the BookX dataset as $\gamma = 1$ achieves the best performance. Similar heat map distributions are shown for the MovieLens and Yahoo datasets in third and fourth rows of Figure 2.4. On both datasets, the negative impact of block-diagonal regularization is not evident if $\gamma < 10$. A trend worth noting is that the two datasets prefer smaller item groups and less impact from block-diagonal regularizations.

Generally, block-diagonal regularization can impact the performance of top-N recommendation. However, solely applying the block-diagonal regularization has a negative effect to the performance (SLIM-$\mathcal{B}$ with $\gamma = 0.01$ generally performs worse). Similarity regularization provides a good remedy for the negative impact of block-diagonal regularization. The results also demonstrate the superiority of block-diagonal regularization over ℓ_1-norm regularization. This is because block-diagonal regularization is approaching ℓ_1-norm regularization when c is approaching n, whereas the best performance is generally achieved when $c \leq 100$.

2.7.3 RQ1.3: Training convergence and stability

We further analyze the impact of BSR on the training procedure to answer RQ3. We plot the test performance of HR@10 after each epoch by comparing SLIM, FISM

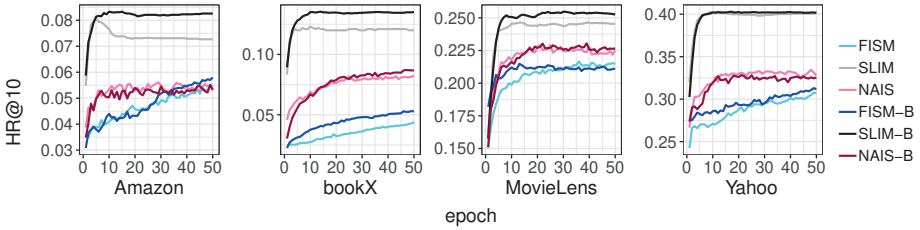


Figure 2.5: Test performance after each epoch.

and NAIS with/without BSR in Figure 2.5. We train all the compared models for 50 epochs. Clearly lines can be categorized into three groups. a) The first group, SLIM and SLIM-$\mathcal{B}$ (gray and black lines), performs the best and shows the best stability and convergence. Generally, SLIM grows and converges faster than SLIM-$\mathcal{B}$, but is outperformed by SLIM-$\mathcal{B}$ soon. On Amazon, while SLIM reaches the peak soon, it experiences a fall later, which is caused by overfitting. In comparison, SLIM-$\mathcal{B}$ stabilizes at the peak due to the effectiveness of BSR. b) The second group, NAIS and NAIS-$\mathcal{B}$ (pink and red lines), shows certain fluctuations. Besides the fluctuations, NAIS and NAIS-$\mathcal{B}$ show similar convergence and stability. Normally, NAIS performs better in the beginning and is outperformed by NAIS-$\mathcal{B}$ with more epochs (On Amazon and Yahoo, NAIS-$\mathcal{B}$ outperforms NAIS after 50 epochs). c) The last group, FISM and FISM-$\mathcal{B}$ (cyan and blue lines), shows the least performance. Different from the first two groups, except for MovieLens, FISM and FISM-$\mathcal{B}$ grow steadily, showing slow convergence rate and poor stability. On Amazon, FISM-$\mathcal{B}$ surpasses the second group and keeps growing after 50 epochs.

Hence, besides improving performance, BSR also helps with stability at the cost of a slight slow down of the convergence rate.

2.8 Summary

In this chapter, we have answered **RQ1** by studying item clustering for item-based collaborative filtering. we have proposed a block-aware similarity regularization (BSR) to capture the block-diagonal structure behind item similarities for item-based collaborative filtering (ICF) methods, so as to improve the top-N recommendation performance. We have applied BSR to similarity models (SMs), which has a theoretical guarantee of block-diagonality. We can theoretically ensure that the learned item similarities are sparse and capture transitive relations within blocks. Due to the scalability limitation of SMs, we have extended the BSR to a block-aware similarity dropout (BSD) and applied it to feature-based similarity models (FSMs). Experimental evaluations on a large number of dataset show the effectiveness of BSR for ICF methods.

In this chapter, we have provided a solution to utilize high-dimensional ratings for top-N recommendation. Next, we utilize additional information for top-N recommendation. We start with a single type of additional information (Chapter 3 and 4), after which we consider multiple types of additional information (Chapter 5–7).

3

Top-N Recommendation with High-dimensional Side Information

In Chapter 2, we have proposed a new regularization term for item-based collaborative filtering methods. The proposed method captures item clustering, which is able to provide effective recommendations with high-dimensional ratings. In this chapter, we propose to utilize high-dimensional side information for top-N recommendation. We provide a joint learning model that learns item similarities and reduces feature dimensions simultaneously. The proposed model provides an answer to the following research question asked in Chapter 1:

RQ2 Can we reduce the dimension of side information for effective top-N recommendation?

3.1 Introduction

Top-N recommendation has been widely adopted to recommend *ranked lists of items* so as to help users identify the items that best fit their personal tastes. Over the last decades, various efforts have been dedicated to provide top-N recommendations. Among them, the *item*-based scheme stands out for its solid performance. Representative methods include item-based k-nearest-neighbor (item$k$NN), sparse linear method (SLIM) [169], and so forth, which have been shown to outperform *user*-based scheme.

The recommendation accuracy of such item-based neighborhood methods relies largely on the item similarities computed or learned. Specifically, item similarities are usually made available based on user feedback (both explicit and implicit), e.g., purchases, ratings, reviews, clicks, and check-ins. Lately, there has been an increase in the amount of additional information associated with items, referred to as *side information* [170]. Typical examples include descriptions of movies in movie recommendation, resumes of applicants in job matching, content of emails in spam detection, reviews of items in online shopping, and so forth. Side information has generated the interest of many researchers and has led to the development of *hybrid* algorithms to enhance the performance of recommendations by taking advantage of such information.

Side information comes with a *high dimensionality*. For example, side information can be the text descriptions of items; when regarding each unique term in the corpus

This chapter was published as [43].

as one dimension, it is indisputably high-dimensional. Moreover, side information can also be in the form of images or videos where the dimensionality is evidently much higher. Nonetheless, existing methods overlook this fact when utilizing side information, and hence, they are facing problems of efficiency and accuracy due to the curse of high dimensionality. We address the issue in this chapter, and investigate how to leverage side information to boost the recommendation performance while limiting the impact from high dimensionality.

While side information is high-dimensional and sparse, it is reasonable to expect a low dimensionality of intrinsic features, and this suggests that we should incorporate dimensionality reduction for this task. Among the many available dimensionality reduction methods, locality preserving projection (LPP) [96] has been shown to produce a low-dimensional space that well preserves locality. As recommendation quality largely depends on item similarity, LPP is a natural candidate in this setting.

To summarize, we propose a top-N recommendation method to harness high-dimensional side information. By introducing a projection matrix, high-dimensional side information is reduced into a low-dimensional space. We present a joint learning model to simultaneously perform LPP and learn item similarity. We then conceive an alternative iterative optimization method to solve the model. Our experimental evaluation shows that the proposed method enjoys a performance gain of up to 21.2% on hit rate at 10 (HR@10) and 36.8% on average reciprocal hit-rank at 10 (ARHR@10) over state-of-the-art methods.

3.2 Related Work

We are aware of several recent methods that leverage side information for top-N recommendation. On top of SLIM [169], sparse linear method with side information (SSLIM) [170] utilizes a regularized optimization process to learn a sparse coefficient matrix. User-specific feature-based similarity model (UFSM) [71] combines item similarity model with factor models. Recently, Zhao et al. [270] have proposed a predictive collaborative filtering approach to utilize side information.

We also summarize recent methods using side information for rating prediction. Gantner et al. [77] proposed to map side information to latent item factors by learning the mapping function. Saveski et al. [198] proposed a local collective factorization method. Lu et al. [153] proposed an interactive model for matrix completion. Distinct from them, we integrate dimensionality reduction into top-N recommendation.

As to dimensionality reduction, this topic has been investigated extensively, for sparse feedback via various methods [192], including principal component analysis, singular value decomposition, non-negative matrix factorization and so on. However, high-dimensional side information has rarely been addressed in the setting, and this paper tries to fill in the gap.

3.3 The Proposed Approach

3.3.1 Notation

We first introduce the notations used throughout the paper. Let $\mathcal{U}$ and $\mathcal{I}$ be the sets of all users and all items, respectively, each of size m and n. The user feedback (both

explicit and implicit) shows the items that the users have purchased, viewed or rated, which is denoted by a matrix R of size $m \times n$. We treat feedback as binary, that is, if user u provided feedback for item i, then the (u, i)-entry of R (denoted by r_{ui}) is 1, otherwise it is 0. The item similarity matrix is represented by $S \in \mathbb{R}^{n \times n}$, where each value of entry s_{ij} is within $[0, 1]$. The feature matrix (side information associated with items) is denoted by $F \in \mathbb{R}^{n \times d}$, where d indicates the dimensionality of side information. The projection matrix is denoted by $W \in \mathbb{R}^{d \times k}$, which is used to map d-dimensional side information into a k-dimensional space where $k \ll d$.

3.3.2 Model description

This section describes the proposed model. We start with introducing the Baseline method without performing dimensionality reduction, then summarize LPP, and explain how to incorporate it in a recommender system. Finally, the proposed method is formed.

Recommendation with side information. Typically, top-N recommender systems perform matrix completion for R, the core of which is to learn item similarity, which is directly relevant to recommendation. Side information is utilized to enhance the learning of item similarity. While various forms of incorporating side information exist, we incorporate a regularization term on S along with feature matrix F and form the model as the following problem:

$$\begin{aligned} \arg\min_{S} \quad & \frac{1}{2}\|R - RS\|_F^2 + \frac{\alpha}{2}\sum_{i,j}^{n}\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 s_{ij} + \frac{\lambda}{2}\|S\|_F^2, \\ \text{such that} \quad & \boldsymbol{s}_j^T \mathbf{1} = 1, s_{jj} = 0, \forall j = 1, \cdots, n; \\ & 0 \leq s_{ij} \leq 1, \forall i, j = 1, \cdots, n, \end{aligned} \tag{3.1}$$

where $\boldsymbol{s}_j$ is the j-th column vector of S, representing how similar item j is to other items. The constraint $\boldsymbol{s}_j^T \mathbf{1} = 1$ is incorporated to avoid the case when the learned S is close to $\mathbf{0}$ especially when R is very sparse. The term $\frac{1}{2}\|R - RS\|_F^2$ in the objective function tries to reconstruct the feedback matrix by learning the coefficient matrix S, which was first introduced by SLIM [169] for top-N recommendation. As suggested there, the ℓ_2-norm is used to regularize S. While ℓ_1-norm is also suggested to encourage sparsity, it is omitted as it turns out to be constant here (due to $\boldsymbol{s}_j^T \mathbf{1} = 1$). α is a user-specified parameter to balance the two sources of information. We further justify the regularization to S by F in detail. Given the feature matrix F, $\boldsymbol{f}_i$ represents the feature vector for item i. A natural way to measure the item distance in terms of features is to compute the Euclidean distance between them, i.e., $\|\boldsymbol{f}_i - \boldsymbol{f}_j\|^2$. Although the item similarity is unknown, it is reasonable to assume that closer items (in terms of feature distance) are likely to have higher similarities, and thus, item similarity between item i and j can be regularized as $\|\boldsymbol{f}_i - \boldsymbol{f}_j\|^2 s_{ij}$.

Locality preserving projection. LPP is a linear approximation of the nonlinear Laplacian Eigenmap. The algorithmic procedure starts with constructing the adjacency graph from feature matrix F. The item similarity matrix S learned from Eq. (3.1) can

be used for this task. Then, we need to solve the generalized eigenvector problem:

$$F^T L F \boldsymbol{w} = \sigma F^T D F \boldsymbol{w}, \tag{3.2}$$

where D is a diagonal matrix, of which the i-th diagonal entry equals $\sum_{j=1}^{n} \frac{s_{ij}+s_{ji}}{2}$; L is a Laplacian matrix for S, i.e., $L = D - \frac{S+S^T}{2}$. The projection matrix W is formed as $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k]$, where eigenvector $\boldsymbol{w}_i$ corresponds to eigenvalue σ_i, which is in an ascending order as $\sigma_1 \leq \cdots \leq \sigma_d$. The linear combination FW denotes the projection of side information in a low-dimensional space.

The proposed model. Putting Eq. (3.1) and (3.2) together forms our proposed model as follows:

$$\begin{aligned} \underset{S, W^T W = I}{\arg\min} \quad & \frac{1}{2}\|R - RS\|_F^2 + \frac{\alpha}{2}\sum_{i,j}^{n}\left(\|W^T \boldsymbol{f}_i - W^T \boldsymbol{f}_j\|_2^2 s_{ij}\right) + \frac{\lambda}{2}\|S\|_F^2, \\ \text{such that} \quad & \boldsymbol{s}_j^T \mathbf{1} = 1, s_{jj} = 0, \forall j = 1, \ldots, n; \\ & 0 \leq s_{ij} \leq 1, \forall i, j = 1, \ldots, n. \end{aligned} \tag{3.3}$$

Rather than impose the constraint $W^T F^T D F W = I$ on W according to LPP, we directly assume $W^T W = I$ to learn a distinctive feature space. Besides, we regularize s_{ij} by $\|W^T \boldsymbol{f}_i - W^T \boldsymbol{f}_j\|_2^2$ instead of $\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2$ for two reasons: a) the model is formulated as a joint learning optimization problem so as to achieve dimensionality reduction and top-N recommendation simultaneously. We will show later in § 3.3.3 that optimizing W is under the framework of LPP; b) the training of the item similarity matrix S is enhanced in the projected low-dimensional feature space. We argue that incorporating LPP is able to not only preserve locality but also improve item similarity, which is explained below.

Denote the projection matrix as $W = \left[\boldsymbol{p}_1^T, \ldots, \boldsymbol{p}_d^T\right]^T$, where $\boldsymbol{p}_i$ is a k-dimensional row vector, representing the embedding of feature i. Though projection, each feature is represented by k distinctive aspects. We contend that the "synonyms" (different but semantically similar features) will have closer embeddings through LPP under the assumption that the synonyms are likely to appear in items with high similarities. Therefore, items containing synonyms will get closer in the projected space, which can further guide the learning of similarity towards more similar.

Once the solution (W^*, S^*) are obtained, we can recover the item-user recommendation score matrix $\tilde{R}$ by setting $\tilde{R} = RS^*$. We then rank the scores for unrated items of each user in a non-increasing order and recommend the first N items.

3.3.3 Solution

The optimization problem defined above is non-convex in terms of S, W together. Thus, it is unrealistic to expect an algorithm to find the global minimum. In what follows, we derive an alternative iterative algorithm to solve the problem.

Fix W update S. We first define the Lagrange function:

$$\begin{aligned} \mathcal{L}(\boldsymbol{s}_j, \boldsymbol{\varphi}_j, \theta_j, \xi_j) = & \frac{1}{2}\|\boldsymbol{r}_j - R\boldsymbol{s}_j\|_2^2 + \frac{\alpha}{2}\boldsymbol{q}_j^T \boldsymbol{s}_j + \theta_j \boldsymbol{s}_j^T \mathbf{1} + \\ & \frac{\lambda}{2}\boldsymbol{s}_j^T \boldsymbol{s}_j + \boldsymbol{\varphi}_j^T \boldsymbol{s}_j + \xi_j s_{jj}, \end{aligned} \tag{3.4}$$

where $\varphi_j, \theta_j, \xi_j, \forall j = 1, \ldots, n$ are the lagrangian multipliers, $q_{ij} = \|W^T \boldsymbol{f}_i - W^T \boldsymbol{f}_j\|_2^2$ and $\mathbf{1}$ is the vector with all elements equal 1. The partial derivation of $\mathcal{L}$ w.r.t $\boldsymbol{s}_j$ is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_j} = R^T R \boldsymbol{s}_j - R^T \boldsymbol{r}_j + \frac{\alpha}{2} \boldsymbol{q}_j + \theta_j \mathbf{1} + \lambda \boldsymbol{s}_j + \boldsymbol{\varphi}_j + \xi_j \boldsymbol{e}_j, \tag{3.5}$$

where $\boldsymbol{e}_j$ is the vector with only the j-th element equal 1 and others 0. A closed-form solution could be derived as follows:

$$s_{ij} = \begin{cases} \left[\left(R^T R + \lambda I \right)^{-1} \left(R^T \boldsymbol{r}_j - \frac{\alpha}{2} \boldsymbol{q}_j - \theta_j \mathbf{1} \right) \right]_{i+}, & \text{if } i \neq j \\ 0, & \text{if } i = j, \end{cases} \tag{3.6}$$

where $R^T R + \lambda I$ is positive definite if $\lambda > 0$ and $\theta_j = \boldsymbol{s}_j^T R^T \boldsymbol{r}_j - \boldsymbol{s}_j^T R^T R \boldsymbol{s}_j - \frac{\alpha}{2} \boldsymbol{s}_j^T \boldsymbol{q}_j - \lambda \boldsymbol{s}_j^T \boldsymbol{s}_j$; $[\cdot]_{i+}$ is the operator to take the i-th element of the vector if it is not less than 0, otherwise 0.

Fix S update W. To update W, we first introduce the following equation, which is based on the theory of spectral analysis:

$$\frac{1}{2} \sum_{i,j} \|W^T \boldsymbol{f}_i - W^T \boldsymbol{f}_j\|_2^2 s_{ij} = \text{Tr} \left(W^T F^T L F W \right). \tag{3.7}$$

Hence, the problem is equivalent to solving

$$\arg\min_{W^T W = I} \text{Tr} \left(W^T F^T L F W \right). \tag{3.8}$$

Applying the Karush-Kuhn-Tucker (KKT) first-order optimality conditions, we derive

$$F^T L F W = \sigma W, \tag{3.9}$$

and the solution is formed by the k eigenvectors of $F^T L F$ corresponding to the k smallest eigenvalues. Note that W is updated under the framework of LPP.

3.4 Experiment

3.4.1 Setup

To evaluate the performance of our method on the task of top-N recommendation with side information, we perform experiments on different real-world datasets. The statistics of the datasets are summarized in Table 3.1.

– *CUL*:[1] an online service that allows researchers to add scientific articles to their libraries. For each user, the articles added in his or her library are considered as preferred articles, from which titles and abstracts are collected and used as side information.

[1] CiteULike: `http://www.citeulike.org/`

Table 3.1: Statistics of the datasets used in this chapter.

Dataset	*#Users*	*#Items*	*#Feeds*	Density	*#Features*
Enron1	663	1,773	1,588	0.14%	25,133
Enron2	953	5,366	3,401	0.07%	32,063
Yahoo	7,594	8,641	19,434	0.03%	7,823
CUL	9,537	8,222	29,352	0.04%	6,860

– *Enron1* and *Enron2*:[2] the two largest mailbox extracted from Enron Email. The data is composed of email messages released during investigation of the Federal Energy Regulatory Commission against the Enron Corporation. By regarding the email content as side information, we predict the most likely recipients of new messages.
– Yahoo:[3] a small sample of the Yahoo! Movies community's preferences for various movies, rated on a scale from A+ to F, binarized to 0 or 1. The dataset also contains a large amount of side information about many movies.

To comprehensively understand the effectiveness of the methods, we adopt 5-time leave-one-out cross-validation (LOOCV). The evaluation of the model is conducted by comparing the recommendation list of each user with the item of that user in the test set. The recommendation quality is measured using hit rate (HR) and average reciprocal hit-rank (ARHR).[4] We evaluate the performance of our proposed method on top-N recommendation.

In this set of experiments, we refer to our method as *projection regularized item similarity model* (Prism). To evaluate its performance, Prism is first compared with SLIM to demonstrate the need to utilize side information when feedback is sparse. The performance of simple cosine-similarity (coSim) [71] is evaluated to show the quality of side information. To appreciate the effectiveness of dimensionality reduction, the performance of Baseline, formulated in Eq. (3.1), is also evaluated. We also compare Prism with state-of-the-art top-N recommendation methods with side information, including SSLIM [170], UFSM [71] and the method proposed in [270] (referred to as PCF). Parameters of all methods are carefully tuned through grid search.

3.4.2 Results and analysis

We vary the size of recommendation list, and find that Prism always achieves the best results. Table 3.2 shows the result of comparisons over four datasets with top-10 items recommended. By looking at the results achieved by SLIM and coSim, we characterize the datasets. Overall speaking, SLIM performs inferiorly to coSim on both Enron1 and Enron2, whereas the order is reversed on Yahoo and CUL. This shows that while all datasets are sparse with respect to user feedback information, the side information of Enron is of high quality and more relevant for recommendation. As the Enron datasets are of higher dimensionality, a significant performance gain is expected with Prism on Enron1 and Enron2. To verify, we scrutinize the results of Prism and Baselines, and find that the improvement of Prism over Baselines is much more evident on Enron1 and Enron2 than that on Yahoo and CUL. These results demonstrate the effectiveness

[2]Enron Mail Box: `https://www.cs.cmu.edu/~enron/`
[3]Yahoo! Movies: `https://webscope.sandbox.yahoo.com/`
[4]For each user, we recommend N items, where $N = 5, 10, 15, 20$. Due to space limitations, we only present the result with $N = 10$.

Table 3.2: Comparison of top-N recommendation algorithms.

Method	Enron1			Enron2		
	Parameters	HR@10	ARHR@10	Parameters	HR@10	ARHR@10
coSim	—	0.1408	0.0992	—	0.1460	0.1196
SLIM	$\beta = 0.6, \lambda = 0.2$	0.0865	0.0347	$\beta = 0.1, \lambda = 0.5$	0.1569	0.0668
SSLIM1	$\alpha = 0.9, \beta = 0.1, \lambda = 0.2$	0.2032	0.0966	$\alpha = 0.8, \beta = 0.1, \lambda = 0.5$	0.2204	0.1119
SSLIM2	$\alpha = \beta = 0.1, \lambda = 0.2$	0.0853	0.0327	$\alpha = 0.1, \beta = 0.1, \lambda = 0.5$	0.1547	0.0733
UFSM$_{rmse}$	$l = 1.\lambda = 0.1, \mu_1 = 0.01, \mu_2 = 10^{-5}$	0.1485	0.1059	$l = 4, \lambda = 0.1, \mu_1 = 0.1, \mu_2 = 10^{-5}$	0.1693	0.1273
UFSM$_{bpr}$	$l = 1.\lambda = 10^{-5}, \mu_1 = 0.01, \mu_2 = 10^{-4}$	0.1416	0.1040	$l = 3, \lambda = 10^{-5}, \mu_1 = 0.01, \mu_2 = 0.01$	0.1511	0.1142
PCF	$\beta = 0.5, \gamma = 10.0, \lambda = 500$	0.2013	0.1011	$\beta = 0.2, \gamma = 5, \lambda = 1000$	0.2318	0.1201
Baseline	$\alpha = 1.0, \lambda = 0.3$	0.0966	0.0435	$\alpha = 0.9, \lambda = 0.5$	0.1679	0.0850
Prism	$k = 100, \alpha = 2.0, \lambda = 0.2$	**0.2153**	**0.1091**	$k = 100, \alpha = 0.3, \lambda = 0.4$	**0.2810**	**0.1742**
Method	**Yahoo**			**CUL**		
	Parameters	HR@10	ARHR@10	Parameters	HR@10	ARHR@10
coSim	—	0.0241	0.0106	—	0.1238	0.0559
SLIM	$\beta = 0.9, \lambda = 0.5$	0.0558	0.0181	$\beta = 1.0, \lambda = 0.5$	0.1961	0.0758
SSLIM1	$\alpha = 0.7, \beta = 0.5, \lambda = 0.1$	0.0543	0.0193	$\alpha = 0.9, \beta = 1.0, \lambda = 0.5$	0.1916	0.0733
SSLIM2	$\alpha = \beta = 0.1, \lambda = 0.5$	0.0485	0.0181	$\alpha = 0.1, \beta = 0.1, \lambda = 0.5$	0.2223	0.0873
UFSM$_{rmse}$	$l = 6, \lambda = 0.1 = \mu_1 = 0.1, \mu_2 = 10^{-4}$	0.0408	0.0195	$l = 5, \lambda = 10^{-5}, \mu_1 = 10^{-4}, \mu_2 = 10^{-5}$	0.1821	0.0705
UFSM$_{bpr}$	$l = 5, \lambda = \mu_1 = \mu_2 = 1^{-5}$	0.0400	0.0192	$l = 5, \lambda = 10^{-5}, \mu_1 = 0.01, \mu_2 = 10^{-5}$	0.1942	0.0803
PCF	$\beta = 1.0, \gamma = 10, \lambda = 2000$	0.0556	0.0208	$\beta = 0.8, \gamma = 5, \lambda = 2000$	0.2167	0.0834
Baseline	$\alpha = 1.0, \lambda = 0.5$	0.0618	0.0230	$\alpha = 1.0, \lambda = 0.1$	0.2118	0.0864
Prism	$k = 300, \alpha = 0.9, \lambda = 0.1$	**0.0672**	**0.0232**	$k = 200, \alpha = 0.3, \lambda = 0.1$	**0.2247**	**0.0936**

of incorporating LPP for recommendation with high-dimensional side information.

As for the comparison with other methods, Prism achieves the best results over all tested datasets, especially on Enron2, which has the highest dimensionality of side information. The recommendation accuracy of Prism on this dataset enjoys a performance gain up to 21.2% on HR@10 and 36.8% on ARHR@10 over state-of-the-art methods. On the Yahoo dataset SSLIM and UFSM actually degrade the accuracy compared with SLIM. While PCF increases it, the increment is limited. This should be attributed to the poor quality of side information. By contrast, Prism improves it, exhibiting the robustness of Prism; that is, even on the dataset where side information is of limited correlation to recommendation, the preferable result could be expected. This robustness is also displayed on CUL, which takes good user feedback but poor side information. It seems that CUL is more suitable to the methods that loosely couple with side information like SSLIM2. In this case, Prism is still able to achieve quite competitive performance, as the relevance of side information is improved through dimensionality reduction and α is tuned small to emphasize more on feedback information. The performance on Enron1 is not that distinctive. As the side information is of high quality, the methods that tightly couple with side information stand out (SSLIM1 and PCF). On the other hand, as the feature dimensionality is lower than that on Enron2, dimensionality reduction is not equally effective.

3.5 Summary

In this chapter, we have answered **RQ2** by performing dimension reduction on side information. Specifically, we have shown the problems encountered when utilizing high-dimensional side information to enhance the performance of recommendation, which had not been well investigated by existing literature. We have proposed a novel method to address the challenge, namely projection regularized item similarity model (Prism). The method integrates locality preserving projection (LPP) and top-N recommendation into a joint learning algorithm. Under the novel framework, LPP not only resolved the issue brought by high dimensionality, but also improved the relevance of item similarity. We have conducted extensive experiments and the results demonstrated the superiority of Prism.

In this chapter, we further exploit high-dimensional side information for top-N recommendation, in addition to the high-dimensional ratings. Next, in Chapter 4, we consider a more challenging task: using high-dimensional and *noisy* side information for top-N recommendation, after which in the following chapters (Chpater 5–7), we make use of *heterogeneous* side information to better understand user's behavior and provide more accurate recommendations.

4

Collective Variational Auto-encoder for Top-N Recommendation

In the previous chapters, we have investigated how to utilize high-dimensional ratings (Chapter 2) and high-dimensional side information (Chapter 3) for top-N recommendation. In this chapter, we also leverage high-dimensional side information for top-N recommendation. We assume side information contains noise and apply denoising techniques. The proposed method answers the following research question asked in Chapter 1:

RQ3 How can we utilize high-dimensional side information with noise for top-N recommendation?

4.1 Introduction

Recommender systems have become increasingly indispensable. Applications include top-N recommendations, which are widely adopted to recommend users ranked lists of items. For e-commerce, typically only a few recommendations are shown to the user each time and recommender systems are often evaluated based on the performance of the top-N recommendations.

Collaborative filtering (CF)-based methods are a fundamental building block in many recommender systems. CF-based recommender systems predict what items a user will prefer by discovering and exploiting similarity patterns across users and items. The performance of CF-based methods often drops when ratings are very sparse. With the increased availability of *side information*, that is, additional information associated with items such as product reviews, movie plots, etc., there is great interest in taking advantage of such information so as to compensate for the sparsity of ratings.

Existing methods utilizing side information are linear models [170], which have a restricted model capacity. Recent work generalizes linear model by deep learning to explore non-linearities for large-scale recommendations [94, 203, 239, 273]. State-of-the-art performance is achieved by applying variational auto-encoders (VAEs) [116] for CF [125, 134, 141]. These deep models learn item representations from side information. Thus, the dimension of side information determines the input dimension of the network, which dominates the overall size of the model. This is problematic since

This chapter was published as [36].

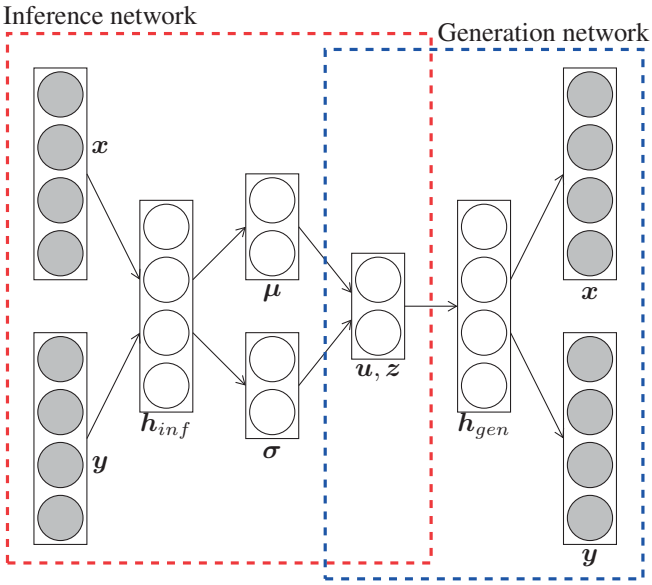


Figure 4.1: Collective variational auto-encoder.

side information is generally high-dimensional [43]. As we will see, existing deep models fail to beat linear models due to the high-dimensionality of side information and an insufficient number of samples.

To avoid the impact from high-dimensionality while exploiting the effectiveness of VAEs, we learn feature representations from side information. The dimensions of the side information correspond to the number of samples rather than the input dimension of the deep network. To instantiate this idea, we propose *collective variational auto-encoder* (cVAE), which learns to recover user ratings and side information simultaneously through VAE. While user ratings and side information are different sources of information, both are information associated with items. Thus, we take ratings from each user and each dimension of side information over all items as the input for VAE, so that samples from both sources of information have the same dimensionality (number of items). We can then feed ratings and side information into the same inference network and generation network. cVAE complements the sparse ratings with side information, as feeding side information into the same VAE increases the number of samples for training. The high-dimensionality of side information is not a problem for cVAE, as it increases the sample size rather than the network scale. To account for the heterogeneity of user rating and side information, the final layer of the generation network follows different distributions depending on the type of information. Training a VAE by feeding it side information as input acts like a pre-training step, which is a crucial step for developing a robust deep network. Our experiments show that the proposed model, cVAE, achieves state-of-the-art performance for top-N recommendation with side information.

The remainder of the chapter is organized as follows. We present preliminaries in § 4.2. We introduce the cVAE model and optimization in § 4.3. § 4.4 describes the experimental setup and results. We review related work in § 4.5 and conclude in § 4.6.

Table 4.1: Notation used in the chapter.

	Notation	Description
Constants	m	Number of users
	n	Number of items
	d	Dimension of side information
	k	Dimension of latent item representation
	N	Number of recommended items
Variables & parameters	$U \in \mathbb{R}^{m \times k}$	Matrix of latent user representation
	$V \in \mathbb{R}^{n \times k}$	Matrix of latent item representation
	$X \in \mathbb{R}^{d \times n}$	Matrix of side information
	$Y \in \mathbb{R}^{m \times n}$	Matrix of user rating
	$Z \in \mathbb{R}^{d \times k}$	Matrix of latent feature representation
	$\boldsymbol{h}_{inf}$	Hidden layer of inference network
	$\boldsymbol{h}_{gen}$	Hidden layer of generation network
	$\boldsymbol{\mu} \in \mathbb{R}^k$	The mean of latent input representation
	$\boldsymbol{\sigma} \in \mathbb{R}^k$	The variance of latent user or feature representation
Functions	$f_\phi(\cdot)$	Non-linear transformation of inference network
	$f_\theta(\cdot)$	Non-linear transformation of generation network
	$\mu(\cdot)$	The activation function to get $\boldsymbol{\mu}$
	$\sigma(\cdot)$	The activation function to get $\boldsymbol{\sigma}$
	$\varsigma(\cdot)$	The sigmoid function

4.2 Preliminaries

4.2.1 Notation

We introduce relevant notation in this section. We use m, n and d to denote the number of users, items and the dimension of side information, respectively. We study the problem of top-N recommendation with high-dimensional side information, where $d \gg n$. We write $X \in \mathbb{R}^{d \times n}$ for the matrix for side information and $Y \in \mathbb{R}^{m \times n}$ for user ratings. We summarize our notation in Table 4.1.

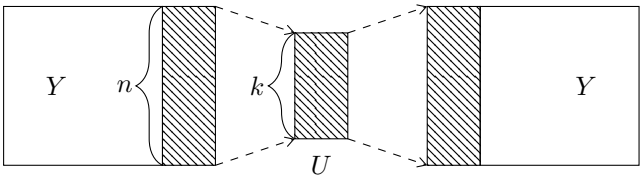
4.2.2 Linear models for top-N recommendation

Sparse linear method (SLIM) [169] achieves state-of-the-art performance for top-N recommendation. SLIM learns to reproduce the user rating matrix Y through:

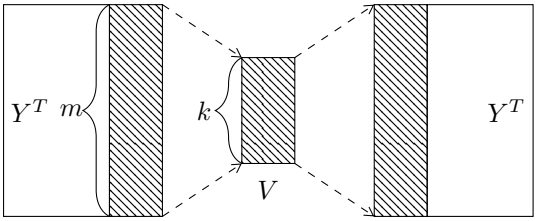
$$Y \sim YW.$$

Here, $W \in \mathbb{R}^{n \times n}$ is the coefficient matrix, which is analogous to the item similarity matrix. The performance of SLIM is heavily affected by the rating sparsity [107]. Side information has been utilized to overcome this issue [43, 170, 270]. As a typical example of a method that uses side information, collective sparse linear method (cSLIM) learns W from both user rating and side information. Specifically, X, Y are both reproduced through:

$$Y \sim YW, \quad X \sim XW.$$



(a) User-side autoencoder



(b) Item-side autoencoder

Figure 4.2: Auto-encoders for collaborative filtering.

cSLIM learns the coefficient matrix W collectively from both side information X and user rating Y, a strategy that can help to overcome rating sparsity by side information. However, cSLIM is restricted by the fact that it is a linear model, which has limited model capacity.

4.2.3 Autoencoders for collaborative filtering

Recently, autoencoders have been used to address CF problems [203, 216, 239, 276]. Autoencoders are neural networks popularized by Kramer [120]. They are unsupervised networks where the output of the network aims to be a reconstruction of the input.

In the context of CF, the autoencoder is fed with incomplete rows (resp. columns) of the user rating matrix Y. It then outputs a vector that predicts the missing entries. These approaches perform a non-linear low-rank approximation of Y in two different ways, using a user-side autoencoder (UAE) (Figure 4.2(a)) or item-side autoencoder (IAE) (Figure 4.2(b)), which recover Y respectively through:

$$U \sim f(Y), \quad Y \sim g(U),$$

and

$$V \sim f(Y^T), \quad Y^T \sim g(V),$$

where $U \in \mathbb{R}^{m \times k}$ is the user representation and $V \in \mathbb{R}^{n \times k}$ is the item representation. Moreover, $f(\cdot)$ and $g(\cdot)$ are the encode network and decode network, respectively. UAEs encode Y to learn a user latent representation U and then recover Y from U. In contrast, IAEs encode the transpose of Y to learn item latent representation V and then recover the transpose of Y from V. Note that UAEs work in a similar way as SLIM, as both can be viewed as reproducing Y through $Y \sim g(f(Y))$, which also captures item similarities.

When side information associated with items is available, the feature-side autoen-

coder (FAE) is utilized to learn item representations:

$$V' \sim f(X^T), \quad X^T \sim g(V'),$$

where $V' \in \mathbb{R}^{n \times k}$ is the item representation. Existing hybrid methods incorporate FAE with IAE as both learn item representations. However, this way of incorporating side information needs to estimate two separate VAEs, which is not an effective way to address rating sparsity. They are also vulnerable to the high dimensionality of side information.

4.3 Method

In this section, we propose a new way to incorporate side information with user ratings by combining the effectiveness of both cSLIM and autoencoders. We propose to reproduce X by a FAE and Y by a UAE. In this way, the input for autoencoders of both X and Y are of the same dimension, i.e., the number of items n. Thus, we can feed X and Y into the same autoencoder rather than two different autoencoders, which helps to overcome rating sparsity.

4.3.1 Collective variational auto-encoder

We propose a collective variational auto-encoder (cVAE) to generalize the linear models for top-N recommendation with side information to non-linear models, by taking advantage of variational auto-encoders (VAEs). Specifically, we propose to recover X, Y through

$$\begin{aligned} U &\sim f_\phi(Y), \quad Y \sim f_\theta(U), \\ Z &\sim f_\phi(X), \quad X \sim f_\theta(Z), \end{aligned}$$

where $f_\phi(\cdot)$ and $f_\theta(\cdot)$ correspond to the inference network and generation network parameterized by ϕ and θ, respectively. An overview of cVAE is depicted in Figure 4.1. Unlike previous work utilizing VAEs, the proposed model encodes and decodes user rating and side information through the same inference and generation networks. Our model can be viewed as a non-linear generalization of cSLIM, so as to learn item similarities collectively from user ratings and side information. While user ratings and side information are two different types of information, cSLIM fails to distinguish them. In contrast, cVAE assumes the output of the generation network to follow different distributions according to the type of input it has been fed.

Next, we describe the cVAE model in detail. Following common practice for VAEs, we first assume the latent variables $\boldsymbol{u}$ and $\boldsymbol{z}$ to follow a Gaussian distribution:

$$\boldsymbol{u} \sim \mathcal{N}(0, I), \quad \boldsymbol{z} \sim \mathcal{N}(0, I),$$

where $I \in \mathbb{R}^{k \times k}$ is an identity matrix. While X and Y are fed into the same network, we would like to distinguish them via different distributions. In this chapter, we assume that Y is binarized to capture implicit feedback, which is a common setting for top-N recommendation [169]. Thus we follow Lee et al. [125] and assume that the rating of user j over all items follows a Bernoulli distribution:

$$\boldsymbol{y}_j \mid \boldsymbol{u}_j \sim Bernoulli(\varsigma(f_\theta(\boldsymbol{u}_j))),$$

where $\varsigma(\cdot)$ is the sigmoid function. This defines the loss function when feeding user rating as input, i.e., the logistic log-likelihood for user j:

$$\log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j) = \sum_{i=1}^{n} y_{ji} \log \varsigma(f_{ji}) + (1 - y_{ji}) \log \left(1 - \varsigma(f_{ji})\right), \tag{4.1}$$

where f_{ji} is the i-th element of the vector $f_\theta(\boldsymbol{u}_j)$ and $f_\theta(\boldsymbol{u}_j)$ is normalized through a sigmoid function so that f_{ji} is within $[0, 1]$.

For side information, we study numerical features so that we assume the j-th dimension of side information from all items follows a Gaussian distribution:

$$\boldsymbol{x}_j \mid \boldsymbol{z}_j \sim \mathcal{N}(f_\theta(\boldsymbol{z}_j), I).$$

This defines the loss function when feeding side information as input, i.e., the Gaussian log-likelihood for dimension j:

$$\log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j) = \sum_{i=1}^{n} -\frac{1}{2}(x_{ji} - f_{ji})^2, \tag{4.2}$$

where f_{ji} is the i-th element of vector $f_\theta(\boldsymbol{z}_j)$. Note that although we assume $\boldsymbol{x}$ and $\boldsymbol{y}$ to be generated from $\boldsymbol{z}$ and $\boldsymbol{u}$ respectively, the generation has shared parameters θ.

The generation procedure is summarized as follows:

1. for each user $j = 1, \ldots, m$:
   (a) draw $\boldsymbol{u}_j \sim \mathcal{N}(0, I)$;
   (b) draw $\boldsymbol{y}_j \sim Bernoulli\left(\varsigma(f_\theta(\boldsymbol{u}_j))\right)$
2. for each dimension of side information $j = 1, \ldots, d$:
   (a) draw $\boldsymbol{z}_j \sim \mathcal{N}(0, I)$;
   (b) draw $\boldsymbol{x}_j \sim \mathcal{N}(f_\theta(\boldsymbol{z}_u), I)$.

Once the cVAE is trained, we can generate recommendations for each user j with items ranked in descending order of $f_\theta(\boldsymbol{u}_j)$. Here, $\boldsymbol{u}_j$ is calculated as $\boldsymbol{u}_j = \mu(f_\phi(\boldsymbol{y}_j))$, that is, we take the mean of $\boldsymbol{u}_j$ for prediction.

Next, we discuss how to perform inference for cVAE.

4.3.2 Variational inference

The log-likelihood of cVAE is intractable due to the non-linear transformations of the generation network. Thus, we resort to variational inference to approximate the distribution. Variational inference approximates the true intractable posterior with a simpler variational distribution $q(U, Z)$. We follow the mean-field assumption [244] by setting $q(U, Z)$ to be a fully factorized Gaussian distribution:

$$\begin{aligned} q(U, Z) &= \prod_{j=1}^{m} q(\boldsymbol{u}_j) \prod_{j=1}^{d} q(\boldsymbol{z}_j), \\ q(\boldsymbol{u}_j) &= \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j^2), \\ q(\boldsymbol{z}_j) &= \mathcal{N}(\boldsymbol{\mu}_{m+j}, \boldsymbol{\sigma}_{m+j}^2), \end{aligned}$$

While we can optimize $\{\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j\}$ by minimizing the kullback-leiber (KL) divergence $\mathbb{KL}(q_\phi \parallel p_\theta)$, the number of parameters to learn grows with the number of users and

dimensions of side information. This can become a bottleneck for real-world recommender systems with millions of users and high-dimensional side information. The VAE replaces individual variational parameters with a data-dependent function through an inference network parameterized by ϕ, i.e., f_ϕ, where $\boldsymbol{\mu}_j$ and $\boldsymbol{\sigma}_j$ are generated as:

$$\begin{aligned}
&\boldsymbol{\mu}_j = \mu(f_\phi(\boldsymbol{y}_j)), \quad \boldsymbol{\sigma}_j = \sigma(f_\phi(\boldsymbol{y}_j)), \quad \forall j = 1, \ldots, m \\
&\boldsymbol{\mu}_{m+j} = \mu(f_\phi(\boldsymbol{x}_j)), \quad \boldsymbol{\sigma}_{m+j} = \sigma(f_\phi(\boldsymbol{x}_j)), \quad \forall j = 1, \ldots, d.
\end{aligned}$$

Putting together $p_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and $p_\phi(\boldsymbol{u} \mid \boldsymbol{y})$ with $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ and $p_\theta(\boldsymbol{y} \mid \boldsymbol{u})$ forms the proposed cVAE (Figure 4.1).

Next we derive the evidence lower bound (ELBO):

$$\mathcal{L}(q) = \mathbb{E}_{q_\phi}\left[\log p_\theta(X, Y \mid U, Z)\right] - \mathbb{KL}\left(q_\phi \parallel p(U, Z)\right). \tag{4.3}$$

We use a Monte Carlo gradient estimator [174] to infer the expectation in Eq. (4.3). We draw L samples of $\boldsymbol{u}_j$ and $\boldsymbol{z}_j$ from q_ϕ and perform stochastic gradient ascent to optimize the ELBO. In order to take gradients with respect to ϕ through sampling, we follow the reparameterization trick [116] to sample $\boldsymbol{u}_j$ and $\boldsymbol{z}_j$ as:

$$\begin{aligned}
\boldsymbol{u}_j^{(l)} &= \mu(f_\phi(\boldsymbol{y}_j)) + \boldsymbol{\epsilon}_1^{(l)} \circ \sigma(f_\phi(\boldsymbol{y}_j)), \\
\boldsymbol{z}_j^{(l)} &= \mu(f_\phi(\boldsymbol{x}_j)) + \boldsymbol{\epsilon}_2^{(l)} \circ \sigma(f_\phi(\boldsymbol{x}_j)), \\
\boldsymbol{\epsilon}_1^{(l)} &\sim \mathcal{N}(0, I), \quad \boldsymbol{\epsilon}_2^{(l)} \sim \mathcal{N}(0, I).
\end{aligned}$$

As the $\mathbb{KL}$-divergence can be analytically derived [116], we can then rewrite $\mathcal{L}(q)$ as:

$$\begin{aligned}
\mathcal{L}(q) = \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{m}\log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j^{(l)}) + \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{d}\log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j^{(l)}) + \\
\sum_{j=1}^{d+m}\left(1 + 2\log(\boldsymbol{\sigma}_j) - \boldsymbol{\mu}_j^2 - \boldsymbol{\sigma}_j^2\right).
\end{aligned} \tag{4.4}$$

We then maximize ELBO given in Eq. (4.4) to learn θ and ϕ.

4.3.3 Implementation details

We discuss the implementation of cVAE in detail. As we feed the user rating matrix Y and the item side information X through the same input layer with n neurons, we need to ensure that the input from both types of information are of the same format. In this chapter, we assume that user ratings are binarized to capture implicit feedback and that side information is represented as a bag-of-words. We propose to train cVAEs through a two-phase algorithm. We first feed it side information to train, which works as pre-training. We then refine the VAE by feeding user ratings. We follow the typical setting by taking f_θ as a multi layer perceptron (MLP); f_ϕ is also taken to be a MLP of the identical network structure with f_θ. We also introduce two parameters, i.e., α and β, to extend the model and make it more suitable for the recommendation task.

Parameter α. Generally, item feature X and user rating Y can be very sparse, that is, the positive entry (x_{ji} or $y_{ji} = 1$) is much less than the negative entry (x_{ji}

or $y_{ji} = 0$). We introduce a parameter α to balance between positive samples and negative samples. Specifically, the loss functions in Eq. (4.1) and (4.2) become

$$\log p_\theta(\boldsymbol{y}_j \mid \boldsymbol{u}_j) = \sum_{i=1}^{n} \alpha \cdot y_{ji} \log \varsigma(f_{ji}) + (1 - y_{ji}) \log (1 - \varsigma(f_{ji}))$$

and

$$\log p_\theta(\boldsymbol{x}_j \mid \boldsymbol{z}_j) = \sum_{i=1}^{n} \frac{c_{ji}}{2} (x_{ji} - f_{ji})^2,$$

where

$$c_{ji} = \begin{cases} \alpha, \text{ if } x_{ji} > 0, \\ 1, \text{ otherwise.} \end{cases}$$

Parameter β. We can adopt different perspectives about the ELBO derived in Eq. (4.3) as: the first term can be interpreted as the reconstruction error, while the second $\mathbb{KL}$ term can be viewed as regularization. The ELBO is often over-regularized for recommendation tasks [141]. Therefore, a parameter β is introduced to control the strength of regularization, so that the ELBO becomes:

$$\mathcal{L}(q) = \mathbb{E}_{q_\phi} \left[\log p_\theta(X, Y \mid U, Z)\right] - \beta \cdot \mathbb{KL} \left(q_\phi \parallel p(U, Z)\right).$$

We propose to train the cVAE in two phases. We first pre-train the cVAE by feeding it side information only. We then refine the model by feeding it user ratings. While Liang et al. [141] suggested to set β small to avoid over-regularization, we opt for a larger value for β during refinement, for two reasons: a) the model is effectively pre-trained with side information; it would be reasonable to require the posterior to comply more with this prior; and b) refinement with user ratings can easily overfit due to the sparsity of ratings; it would be reasonable to regularize heavier so as to avoid overfitting.

4.4 Experiments

4.4.1 Experimental setup

Dataset. We conduct experiments on two datasets, Games and Sports, constructed from different categories of Amazon products [157]. For each category, the original dataset contains transactions between users and items, indicating implicit user feedback. The statistics of the datasets are presented in Table 4.2. We use the product reviews as item features. We extract unigram features from the review articles and remove stopwords. We represent each product item as a bag-of-word feature vector.

Table 4.2: Statistics of the datasets used.

Name	*#User*	*#Item*	*#Rating*	*#Feature*	*Feature density*
Games	5,195	7,163	96,316	20,609	3.49%
Sports	5,653	11,944	86,149	31,282	0.97%

Methods for comparison. We contrast the performance of cVAE with that of existing existing VAE-based methods for CF: collaborative variational auto-encoder (cfVAE) [134] and multinomial variational auto-encoder (mVAE) [141]. Note that the performance of cfVAE will be affected greatly by the high-dimensionality of side information. Besides, as cfVAE was designed originally for the rating prediction task, the recommendations provided by cfVAE are likely be less effective. While mVAE is effective for top-N recommendation, it suffers from rating sparsity as side information is not utilized.

We also compare with the state-of-the-art linear model for top-N recommendation with side information, i.e., cSLIM [170]. By comparing with cSLIM, we can evaluate the capacity of cVAE as it can be regarded as a deep extension of cSLIM. We also compare with fVAE, which is the pre-trained model of cVAE with side information only. cVAE extends fVAE with user ratings.

For all VAE-based methods, we follow [116] to set the batch size as 100 so that we can set $L = 1$. We choose a two-layer network architecture for the inference network and generation network. For cfVAE and mVAE, the scale is 200–100 for inference network and 100–200 for generation network. For fVAE and cVAE, the scale is 1000–100 and 100–1000, respectively. The reason that the network scales for cvVAE and mVAE are smaller is that 1. the input for cfVAE is high-dimensional with relatively fewer samples; and 2. the input for mVAE is sparse, which easily overfits with larger networks. In comparison, we can select more hidden neurons for fVAE as it takes each dimension of the features over all items as input, so that the input for the network has relatively fewer dimensions and the number of samples is sufficient. This is similar with cVAE, which uses side information to overcome rating sparsity.

Evaluation method. To evaluate the performance on the top-N recommendation task, we split the user rating matrix Y into Y_{train}, Y_{valid} and Y_{test}, respectively, for training the model, selecting parameters, and testing the recommendation accuracy. Specifically, for each user, we randomly hold 10% of the ratings in the validation set and 10% in the test set and put the other ratings in the training set. For each user, the unrated items are sorted by predicted score in decreasing order and the first N items are returned as the top-N recommendations for that user.

Given the list of top-N recommended items for user u, precision at N (Pre@N) and recall at N (Rec@N) are defined as

$$Pre@N = \frac{|\text{relevant items} \cap \text{recommended items}|}{N},$$
$$Rec@N = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|}.$$

average precision at N (AP@N) is a ranked precision metric that gives larger credit to correctly recommended items in the top-N ranks. AP@N is defined as the average of precision scores computed at all positions with an adopted item, namely

$$AP@N = \frac{\sum_{k=1}^{N} Pre@k \times rel(k)}{\min\{N, |\text{relevant items}|\}},$$

where Pre@k is the precision at cut-off k in the top-N recommended list. Here, $rel(k)$

Table 4.3: Parameter selection.

Method	Parameters	
	Games	Sports
cSLIM	$\alpha = 10^{-2}, \beta = 10^{-3}, \lambda = 1$	$\alpha = 0, \beta = 10, \lambda = 10$
cfVAE	$\lambda_u = 10, \lambda_v = 0.8$	$\lambda_u = 10, \lambda_v = 1$
mVAE	$\alpha = 4, \beta = 0.1$	$\alpha = 8, \beta = 0.1$
fVAE	$\alpha = 6, \beta = 0.1$	$\alpha = 5, \beta = 0.9$
cVAE	$\alpha = 2, \beta = 2$	$\alpha = 1, \beta = 2$

Table 4.4: Comparison of top-N recommendation performance.

Method	Rec@5	Rec@10	Rec@15	Rec@20	MAP@5	MAP@10	MAP@15	MAP@20
	Games							
cSLIM	0.0761	0.1162	0.1474	0.1734	0.0590	0.0337	0.0240	0.0188
cfVAE	0.0685	0.1065	0.1359	0.1608	0.0519	0.0298	0.0212	0.0165
mVAE	0.0137	0.0206	0.0270	0.0375	0.0106	0.0060	0.0043	0.0034
fVAE	0.0495	0.0796	0.1072	0.1276	0.0390	0.0230	0.0167	0.0131
cVAE	**0.0858***	**0.1376****	**0.1731****	**0.2081****	**0.0668***	**0.0394****	**0.0279****	**0.0218****
	Sports							
cSLIM	0.0419	0.0622	0.0776	0.0911	0.0263	0.0148	0.0104	0.0080
cfVAE	0.0315	0.0512	0.0639	0.0768	0.0206	0.0119	0.0084	0.0065
mVAE	0.0171	0.0249	0.0328	0.0390	0.0109	0.0062	0.0044	0.0034
fVAE	0.0284	0.0437	0.0602	0.0732	0.0190	0.0109	0.0078	0.0061
cVAE	**0.0441**	**0.0655**	**0.0857***	**0.1035***	**0.0268**	**0.0152**	**0.0107**	**0.0084**

We attach asterisks to the best score if the improvement over the second best score is statistically significant; we conducted two-sided tests for the null hypothesis that cVAE and the second best have identical average values (* $p < 0.1$, ** $p < 0.05$).

is an indicator function

$$rel(k) = \begin{cases} 1 & \text{if the item ranked at } k \text{ is relevant,} \\ 0 & \text{otherwise.} \end{cases}$$

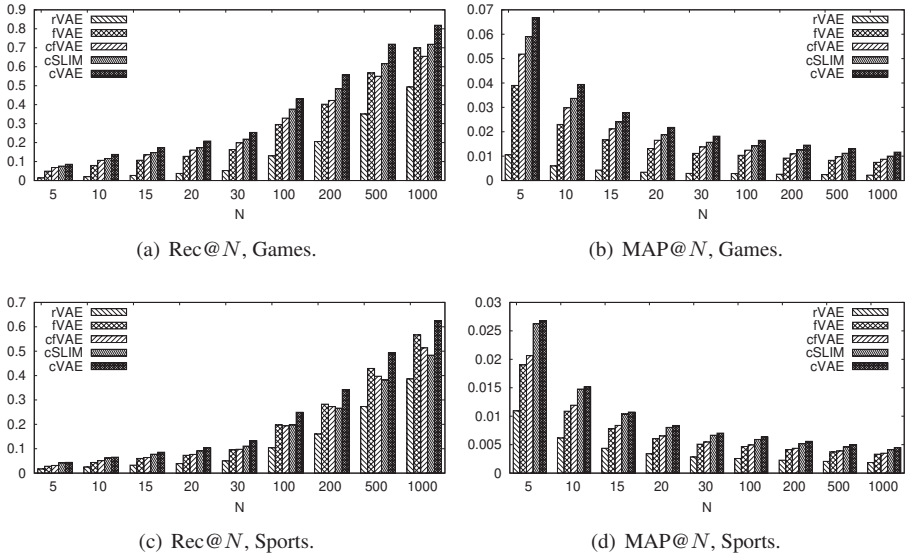
mean average precision at N (MAP@N) is defined as the mean of the AP scores for all users. As in [239], the list of recommended items is evaluated with Y_{test} using Rec@N and MAP@N.

4.4.2 Experimental results

Parameter selection. To compare the performance of alternative top-N recommendation methods, we first select parameters for all the methods through validation. Specifically, for cSLIM, we select α, β and λ from 0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10. For cfVAE, we select λ_u from $0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10$ and λ_v from $0, 0.1, 0.2, \ldots, 1$. For mVAE and fVAE, we select α from $1, 2, \ldots, 10$ and β from $0, 0.1, 0.2, \ldots, 1$. For cVAE, we select α from $1, 2, \ldots, 10$ and β from $0, 0.5, 1, \ldots, 3$. Note that we tune β with larger values to possibly regularize heavier during the refinement. The result of parameter selection is shown in Table 4.3.

Performance comparison. We present the results in terms of Rec@N and MAP@N in Table 4.4, where N is respectively set as $N = 5, 10, 15, 20$. We show the best score in boldface.

As shown in Table 4.4, cVAE outperforms other methods according to all metrics



(a) Rec@N, Games.

(b) MAP@N, Games.

(c) Rec@N, Sports.

(d) MAP@N, Sports.

Figure 4.3: Performance on the top-N recommendation task.

and on both datasets. The improvement is also significant in many cases. The significance of the improvements becomes more evident when N gets larger. The other three methods utilizing VAE are less effective with high-dimensional side information. They even fail to beat linear models. In contrast, cVAE improves over cSLIM by using VAEs for non-linear low-rank approximation. This demonstrates the effectiveness of our proposed cVAE model.

On the Games dataset, cVAE shows significant improvements over the state-of-the-art. Apart from cVAE, cfVAE provides the best recommendation among all VAE-based CF methods, though it fails to beat cSLIM. This is followed by fVAE, which utilizes side information only. mVAE performs the worst, due to the rating sparsity. On the Sports dataset, significant improvements can only be observed for Rec@15 and Rec@20. The results yield an interesting insight. If we look at the parameter selection for cSLIM, we can see that α is set to 0, which means cSLIM generates the best recommendations when no side information is utilized. This does not necessarily mean that the side information of Sports is useless for the recommendation task. Actually, fVAE provides acceptable recommendations by utilizing side information only. Therefore, the way of incorporating side information by cSLIM is not effective. In comparison, cVAE improves over cSLIM by utilizing side information.

Effect of the number of recommended items. Table 4.4 reveals a possible trend that the recommendation improvement becomes more evident when more items are recommended. We use Figure 4.3 to illustrate this, where N is increased from 5 to 1000. As depicted in Figure 4.3(a), the gaps between cVAE and other methods is getting larger with as N increases. It is interesting to note that fVAE surpasses cfVAE when $N = 500$ and $N = 1000$. This further demonstrates the effectiveness of a pre-training phase with side information proposed in this model. In Figure 4.3(c), both

fVAE and cfVAE outperform cSLIM when $N \geq 200$, and fVAE outperforms cfVAE when $N \geq 100$. This shows that deep models are superior to linear models when more items are recommended. In comparison, the improvement achieved by cVAE is more evident when $N \geq 100$, and the gap between cVAE and the second best method is always substantial. In contrast, the performance w.r.t. MAP@N does not show big differences when N grows. Note that on the Games dataset (Figure 4.3(b)), cVAE performs much better than cSLIM when N is small. The improvement becomes less evident when N grows.

4.5 Related Work

Related work concerns linear models for top-N recommendation with side information and deep models for collaborative filtering.

4.5.1 Top-N recommendation with side information

Various methods have been developed to incorporate side information in recommender systems. Most of these methods have been developed in the context of the rating prediction problem, whereas the top-N recommendation problem has received less attention. In the rest of this section we only review methods addressing top-N recommendation problems. Ning et al. [170] propose several methods to incorporate side information with SLIM [169]. Among these methods, cSLIM generally achieves the best performance as it can well compensate for sparse ratings with side information. Zhao et al. [270] propose a joint model to combine self-recovery for user rating and predication from side information. Side information is also utilized to address cold-start top-N recommendation. Elbadrawy et al. [71] learn feature weights for calculating item similarities. Sharma et al. [207] further improve over [71] by studying feature interactions. While these methods deliver state-of-the-art performance for top-N recommendation, they are all linear models, which have a restricted model capacity.

4.5.2 Deep learning for hybrid recommendation

Several authors have attempted to combine deep learning with collaborative filtering. Wu et al. [239] utilize a denoising autoencoder to encode ratings and recover the score prediction. Zhuang et al. [276] propose a dual-autoencoder to learn representations for both users and items. He et al. [94] generalize matrix factorization for collaborative filtering by a neural network. These methods utilize user ratings only, that is, side information is not utilized. Wang et al. [230] propose stacked denoising autoencoders to learn item representations from side information and form a collaborative deep learning method. Later, Li et al. [133] reduce the computational cost of training by replacing stacked denoising autoencoders by a marginalized denoising autoencoder. Rather than manually corrupting the input, variational autoencoders were later utilized for representation learning [134]. These models achieve state-of-the-art performance among hybrid recommender systems, but they are less effective when side information is high-dimensional. For more discussions on deep learning based recommender systems, see [263].

4.6 Summary

In this chapter, we have answered **RQ3** by providing a new network structure on top of VAEs. Specifically, we have proposed a new way to feed side information to a neural network so as to overcome the high-dimensionality. We have proposed collective variational auto-encoder (cVAE), which can be regarded as a non-linear generalization of cSLIM. cVAE overcomes rating sparsity by feeding both ratings and side information into the same inference network and generation network. To cater for the heterogeneity of information, i.e., rating and side information, we have assumed them to follow different distributions, which is reflected in the use of different loss functions. As for the implementation, we have introduced a parameter α to balance the positive samples and negative samples. We also have introduced β as a parameter for regularization, which controls how much the latent variable should comply with the prior distribution. We have conducted experiments using two Amazon datasets. The results show the superiority of cVAE over other methods in scenarios with high-dimensional side information.

In this chapter, we leverage high-dimensional and noisy side information for top-N recommendation, which is an even more challenging task than those studied in previous chapters. Next, we design models to utilize *heterogeneous* information for top-N recommendation, i.e., how to effectively integrate different types of information (Chapter 5–7).

Part II

Heterogeneous Information

5 Local Variational Feature-based Similarity Model

In the previous chapters, we have studied how to utilize high-dimensional information for top-N recommendations (Chapter 2–4). Next, we study how to leverage heterogeneous information for top-N recommendations. In this chapter, we examine the problem of recommending top-N *new* items for users. To do so, we design a model to seamlessly integrate ratings with side information, which answers the following research question asked in Chapter 1:

RQ4 How can we effectively fuse item features with ratings for the recommendation of top-N new items?

5.1 Introduction

Top-N recommendation systems expose users to a limited number of items that reflect the most relevant items a user has not yet rated. This helps users cope with large volumes of information. Existing methods for this task broadly fall into two categories: latent space methods and neighborhood-based methods. Latent space methods [57] learn a low-rank factorization of the user-item matrix into user and item factor matrices, representing both the users and the items in a common latent space. Neighborhood-based methods [63] (user-based or item-based) focus on identifying similar users/items, where item-based neighborhood methods demonstrate better performance than user-based ones [51, 52]. Item-based neighborhood methods can be further categorized into two classes: memory-based [63, 197] and model-based [107, 169]. Memory-based methods compute similarities between items based on statistical measures, such as Pearson coefficient and cosine similarity. However, recommendations based on such heuristic-based approaches are usually inferior. Compared to memory-based methods, model-based methods, often known as similarity models, achieve state-of-the-art performance on the top-N recommendation task by learning similarities from data [107, 169].

It remains a challenging task to recommend top-N *cold-start* items, that is, recommending N items to users from a set of *new* items. The problem of recommending top-N new items is significant because new items are continuously observed: new

This chapter was published as [42].

products are introduced, new books and articles are written, and news stories break. Conventional similarity models cannot generate a recommendation in a cold-start setting [1, 14, 15, 77, 198]. The cold-start problem strongly impacts recommendation performance and the user experience, hence it attracts much attention from the research community [50, 142, 225]. Feature-based similarity models (FSMs) address the problem by extending similarity models to utilize auxiliary information associated with items, i.e., item features, where item similarity is calculated using item features. FSMs have demonstrated their effectiveness for recommending top-N new items [71, 207].

Existing FSMs have the following limitations:

– *They estimate global similarity functions only*. Existing FSMs exploit information across all users to estimate the similarity function, thus assuming that items have the same similarities for all users. In many real-world applications, item similarities should be better identified within subsets of users [51, 208], especially when a large number of users are involved. In fact, there could be a pair of items that are extremely similar for a specific subset of users, while they have low similarity for another subset of users. Existing FSMs fail to capture item similarities w.r.t. a specific aspect that is only of interest to a subset of like-minded users.
– *The estimated similarity functions are linear*. Linear similarity functions fail to capture the complex structure underlying item features. Item similarities measured by linear functions can also be inaccurate, especially when item features are sparse.

To overcome these limitations of existing FSMs, we propose to model local aspects of items that are of interest for a subset of users and extend the linear similarity function to a non-linear one. Specifically, we first identify user subsets via clustering, where users within the subset share similar preferences. For each user subset, we estimate a local similarity function. Motivated by the success of deep learning in the context of collaborative filtering [133, 230], we also estimate a global similarity function that encodes item features into deep representations to measure item similarity in a latent space. Local similarity functions capture specific aspects of items and the global similarity function encodes more abstract properties of items. The combination of local and global similarity functions captures feature-based item similarities from different perspectives.

One challenging task is how to combine deep learning with item collaborative filtering and user clustering: (1) deep learning requires the inputs to be i.i.d. [230]; therefore, it is difficult for deep models to capture implicit relationships among items, which is crucial for item collaborative filtering; and (2) deep learning is rarely applied to clustering problems; typically, deep learning-based methods are used for dimensionality reduction, followed by classical clustering techniques applied to the resulting low-dimensional space [242].

We address the challenge of combining deep learning with item collaborative filtering and user clustering by introducing a Bayesian generative model [116, 227]. We propose a *local variational feature-based similarity model* (LVSM) that integrates deep learning with user clustering and collaborative filtering for top-N cold-start item recommendation. Inference for LVSM is challenging due to the complex entanglement of variables and the non-linear structure within the deep network. Therefore, we conduct variational inference. Existing deep learning for collaborative filtering methods

introduce offset variables on top of latent item representations, which can facilitate variational inference [134, 230]. However, for new items, the offset cannot be inferred due to the absence of ratings. In order to recommend new items, they simply ignore the offset, which brings bias between the rated items and new items. Unlike these methods, LVSM assumes that the generation of user ratings depends directly on the latent item representations. However, this also brings an extra difficulty for inference. We derive the evidence lower bound (ELBO) with approximations, based on which ELBO can be efficiently optimized through a variational EM procedure.

The contributions of this chapter can be summarized as follows:

1. We propose a deep generative model, LVSM, to address the item cold-start top-N recommendation problem. The model can capture local aspects of items and measure global item similarity based on deep representations extracted from item features.
2. To address the difficulty of optimizing LVSM, we perform variational inference and derive the ELBO. Given this approximation, LVSM can be optimized efficiently.
3. We conduct comprehensive experiments to demonstrate the effectiveness of LVSM, yielding important insights into how it generates robust recommendations with a large fraction of cold-start items and sparse item features.

The remainder of the chapter is organized as follows. We introduce preliminaries in § 5.2. We review related work in § 5.3. We propose our model, LVSM, in § 5.4 and then conduct variational inference in § 5.5. § 5.6 and § 5.7 describe our experimental setup and results. We summarize the chapter in § 5.8.

5.2 Problem Definition

In this work, we consider the cold-start top-N recommendation problem, i.e., the problem of recommending items that have neither been seen nor rated by users. The problem is defined as follows: given a set of new items (rating information for these items from users is entirely missing) and their features (characteristics such as genre, product categories, keywords, etc.), recommend each user with the top-N items selected from the new items.

To recommend new items, standard cold-start recommender systems work as follows [71, 207]: (1) for a given user, predict her preference scores for all new items; the preference scores are predicted using some models; (2) for this user, the new items are sorted using their predicted scores in non-increasing order; the N items at the top of the sorted list are recommended to her; and (3) repeat this process for each user in the system.

Next, we introduce the relevant notation. We write m, n, d for the number of users, items and item features, respectively. We refer to Y as the preference matrix; y_{ui} represents the rating of user u to item i. In many scenarios, user ratings are in the form of implicit feedback, such as purchase history, watching habits, browsing activity, etc. Following the common setting for implicit feedback [65], we assume that user ratings are binarized [100, 175, 228]. We refer to the item feature matrix as X. Then, $\boldsymbol{x}_i$ represents the feature vector for item i and x_{ij} represents the j-th feature of item i. We assume numerical values for item features; in this way, we are able to handle various multimedia features [134]. The notation used to describe LVSM as well

as other models is summarized in Table 5.1.

Table 5.1: Notation used in this chapter.

	Notation	Description
Sets and numbers	$\mathcal{U}$	Set of users
	$\mathcal{I}$	Set of items
	$\mathcal{R}_u^+$	Set of items rated by user i
	$\mathcal{R}_{u-i}^+$	Set of items rated by user u excluding item i
	m	Number of users, i.e., $\lvert\mathcal{U}\rvert$
	n	Number of items, i.e., $\lvert\mathcal{I}\rvert$
	d	Number of item features
	c	Number of user groups
	n_u	Number of items rated by user i, i.e., $\left\lvert\mathcal{R}_u^+\right\rvert$
	n_{u-i}	Number of items rated by user i excluding item j, i.e., $\left\lvert\mathcal{R}_{u-i}^+\right\rvert$
Variables	$Y \in \mathbb{R}^{m\times n}$	User rating matrix
	$X \in \mathbb{R}^{n\times d}$	Item feature matrix
	$V \in \mathbb{R}^{n\times h}$	Latent item representation matrix
	$\boldsymbol{v}_i \in \mathbb{R}^h$	Latent representation of item i
	$\boldsymbol{x}_i \in \mathbb{R}^d$	Feature vector of item i
	$\boldsymbol{h}_i^{inf}$	Hidden variables of item i in the inference network
	$\boldsymbol{h}_i^{gen}$	Hidden variables of item i in the generation network
	y_{ui}	Rating of user u for item i
	s_{ij}	Similarity between item i and item j
	z_u	Indicator of the group for user u
Parameters	Θ	Parameters of the generative model
	Φ	Parameters of the inference model
	θ	Parameters of generation network
	ρ	Parameters of the inference network
	Ω	Parameters of feature weights
	$\boldsymbol{\omega}_k \in \mathbb{R}^d$	Parameters of feature weights for k-th user group
	$\pi \in \mathbb{R}^{m\times c}$	Variational parameters of Z

5.3 Related Work

The idea of estimating multiple local models together with a global model has previously been found to be effective for many recommendation tasks, including rating prediction in both general [119] and cold-start settings [208] and top-N recommendation [51, 52]. The broader message of this chapter is that we extend the effectiveness of the idea to top-N recommendation in a cold-start setting.

Local variational feature-based similarity model (LVSM) is specifically designed for recommending top-N new items. Besides reviewing models specifically designed for this problem, we review related works concerning a broader scope, e.g., methods designed for item cold-start recommendation. This is because many item cold-start recommendation methods can also provide a top-N recommendation from new items, even though they have originally been designed for rating prediction over new items.

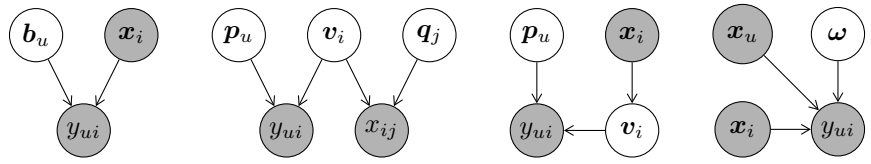
Naively, new items may be recommended to users based on their popularity [177]

or based on a random selection [149]. The accuracy of these methods is low as they cannot provide personalized recommendations. Alternative methods have been proposed to warm-up cold-start items by forcing several representative users to rate them [55, 149]. In recent years, there has been an increase in interest in utilizing other rich sources associated with items along with the rating matrix to increase the accuracy of the recommendation [7, 76, 235, 256], and in dealing with cold-start challenges. Although many other hybrid methods [160, 183, 210] also utilize item features, they are specifically designed to address the data sparsity problem and fail to cope with cold-start item problems, which is the main focus of this chapter.

Next, we discuss work that utilizes item features, namely so-called feature-based methods.

5.3.1 Feature-based methods

Based on how the rating of user u for item i, i.e., y_{ui}, is generated, different models have been proposed. Here, we review four common methods, respectively *user modeling* (UM), *latent factor model* (LFM), *item feature mapping* (IFM) and *feature-based similarity model* (FSM). We describe each category of models and depict them as probabilistic graphical models in Figure 5.1.



(a) User modeling. (b) Latent factor model. (c) Item feature mapping. (d) Feature-based similarity model.

Figure 5.1: Overview of existing feature-based methods represented as probabilistic graphical models. $\boldsymbol{b}_u$: a parameter associated with user u. $\boldsymbol{p}_u, \boldsymbol{v}_i, \boldsymbol{q}_j$: latent factors associated with user u, item i and feature j. $\boldsymbol{x}_i, \boldsymbol{x}_j$: feature vector for item i, j. $\boldsymbol{x}_u$: feature vector for user u, defined as $\boldsymbol{x}_u = \sum_{j \in \mathcal{R}^+_{u-i}} \boldsymbol{x}_j$. $\boldsymbol{\omega}$: parameters for similarity function.

User modeling (UM). One of the earliest approaches for identifying which of the new items may be relevant to a user is user modeling [13, 61, 78, 265]. These methods learn to generate personalized recommendations by formulating the task as a classification or regression problem. While they provide personalized recommendations, they are generally regarded as content-based filtering methods, which fail to take advantage of collaborative filtering. Later, factorization machines [34, 37, 185] have been proposed to capture feature interactions. Factorization machines can utilize item features and can be categorized as UM in the scenario of item cold-start recommendation.

Latent factor model (LFM). LFMs provide a better way to utilize item features that does take recent advances in matrix factorization methods into account [1, 172, 184, 198, 209, 275]. Rating and item feature matrices are simultaneously decomposed, sharing latent item factors. However, LFM requires a large parameter space, especially when item features are high-dimensional. LFM also shares item factors across different contexts [102]. This is problematic as item factors that are cold-start in the context of

user ratings will be learned mainly based on data from the context of item features that are not cold-start, and therefore the item factors are not properly learned in an item cold-start setting.

Item feature mapping (IFM). An alternative type of feature-based model is IFM. Several authors [77, 229, 262] take item features to form a regression model. Unlike UM, they first learn a mapping function to project item features into a common latent space as user factor. Wang et al. [229] propose an IFM based on topic modeling. A recent trend is to extract deep latent item factors for collaborative filtering [82, 226, 234]. Auto-encoders have recently been studied to learn item representations from content [134, 230]. Item representations are used as regularizations for item factors.

Feature-based similarity model (FSM). FSMs have been shown to achieve state-of-the-art performance for recommending top-N new items [4, 22, 63, 71, 107, 169, 207]. FSMs learn similarity functions, measuring item similarities based on item features. The similarity functions are estimated across all users, exploiting the effectiveness of item collaborative filtering. Existing FSMs estimate linear or bilinear similarity functions [71, 207]. As LVSM, our proposed method, follows the general framework of FSMs, we now discuss FSMs in more detail.

5.3.2 Feature-based similarity models

FSMs attempt to predict a rating score y_{ui} of user i for a new item j by defining:

$$\tilde{y}_{ui} = \sum_{j \in \mathcal{R}^+_{u-i}} \text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{5.1}$$

where $\mathcal{R}^+_{u-i}$ is the set of items rated by user i excluding item j; $\text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a similarity function that measures the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. When $\text{sim}(\cdot)$ is linear or bilinear, Eq. (5.1) can be rewritten as:

$$y_{ui} = \text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i). \tag{5.2}$$

Here, $\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i)$ measures the similarity of $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$, where $\boldsymbol{x}_u = \sum_{j \in \mathcal{R}^+_{u-i}} \boldsymbol{x}_j$. There are several definitions for the similarity function $\text{sim}(\cdot)$. One of the most intuitive ones is to calculate the dot product [63]:

$$\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i) = \boldsymbol{x}_u^T \boldsymbol{x}_i. \tag{5.3}$$

The similarity function defined in Eq. (5.3) has several drawbacks:

1. *learning free*: the similarity function is predefined; it does not utilize historical preferences in order to estimate a similarity function that better predicts the observed preferences;
2. *equal weights*: the features are treated equally when measuring item similarity; and
3. *non-collaborative*: the rating score that is computed for a new item w.r.t. user u relies entirely on the set of items previously liked by u, and as such it does not use information from other users.

To overcome these drawbacks, personalized feature weighting (PFW) [22] has been proposed; it introduces personalized weights $\boldsymbol{\omega}_u$ for item features:

$$\text{sim}(u, i) = \boldsymbol{\omega}_u^T \left(\boldsymbol{x}_u \circ \boldsymbol{x}_i \right), \tag{5.4}$$

where $\circ$ is the element-wise product between vectors. PFW introduces learning parameters to the model and weighs features to provide personalized recommendations. However, PFW also fails to take advantage of collaborative filtering as $\boldsymbol{\omega}_u$ is optimized separately for each user. Later, user-specific feature-based similarity model (UFSM) [71] has been introduced, which defines $\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i)$ as:

$$\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i) = \sum_{k=1}^{c} \pi_{uk} \boldsymbol{\omega}_k^T \left(\boldsymbol{x}_u \circ \boldsymbol{x}_i\right). \tag{5.5}$$

Eq. (5.5) defines c global similarity functions $(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c)$ and user-specific contributions of each global similarity function $(\pi_{u1}, \ldots, \pi_{uc})$. user-specific feature-based similarity model (UFSM) exploits item collaborative filtering by estimating $\{\boldsymbol{\omega}_k\}$ across all users. However, UFSM fails to take into consideration interactions among features. UFSM considers item features independently. Hence, the similarity measured this way could be inaccurate especially when features are high-dimensional and sparse, where two items might share few common features. To capture feature interactions, a feature-based factorized bilinear similarity model (FBSM) [207] has been proposed, where $\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i)$ is defined as:

$$\text{sim}(\boldsymbol{x}_u, \boldsymbol{x}_i) = \boldsymbol{x}_u^T D \boldsymbol{x}_i + \boldsymbol{x}_u^T F F^T \boldsymbol{x}_i, \tag{5.6}$$

where D and FF^T approximate the diagonal and off-diagonal of the feature interaction matrix, respectively. While UFSM and FBSM demonstrate superior performance for item cold-start top-N recommendations, the linearity of both models has restricted their expressiveness. Both methods estimate similarity functions from information across all users, rather than subsets of like-minded users, thus failing to capture local aspects.

5.3.3 Local collaborative filtering

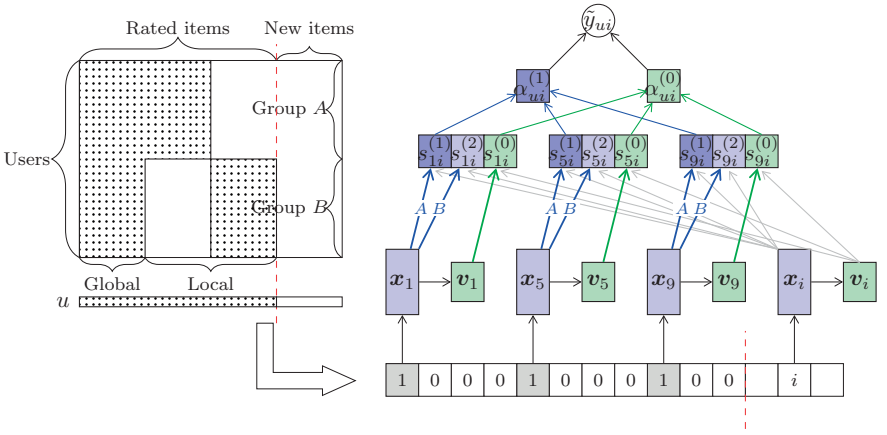
Clustering has been widely studied for collaborative filtering [21, 81, 124, 232, 246, 251, 268]. Previous methods cluster users or items based on user ratings into subgroups and then train a local model separately for each cluster. The results from all subgroups are aggregated to produce recommendations.

Christakopoulou et al. [52] propose local latent factor models, where the assignments of users to subsets are constantly updated. Wang et al. [232] introduce a probabilistic model to cluster items as topics. Wu et al. [238] propose a mixture model to infer memberships of users or items to subgroups. Lee et al. [123] describe an iterative way to estimate latent factors, where, first, latent factors representing the anchor points are estimated and, then, based on the similarities of the observed entries to the anchor points, the latent factors are re-estimated. Christakopoulou et al. [51] explore subsets of users to learn user-specific local item similarity models, which are combined with a global similarity model.

Unlike these methods, LVSM addresses the problem of recommending new items by combining user clustering with deep learning.

5.3.4 Review-based recommendation

User reviews are an important source of information for recommendation; they can help to address the rating sparsity for collaborative filtering methods [16, 32, 48, 87,



(a) Rating matrix. (b) Network structure of LVSM.

Figure 5.2: An illustrative example of LVSM.

272]. Existing review-based recommenders show their effectiveness by applying sentiment analysis [180], topic modeling [157, 218] or aspect extraction [16, 35, 266] to user reviews. By concatenating all the reviews belonging to an item as item features, these methods can also help to tackle the item cold-start problem [48, 272]. Unlike these methods, we propose to utilize generic item features. We only assume to have similarity information instead of the semantic information behind item features. Techniques applied to user reviews cannot be applied in our setting.

5.4 Local Variational Feature-based Similarity Models

5.4.1 Overview

In this chapter, we study the problem of recommending top-N new items to users. The solution provided in our work falls into the cold-start recommendation framework introduced in § 5.2 and we contribute a more effective model for predicting scores for new items. Specifically, we propose a Bayesian generative model, an addition to the family of feature-based similarity models (FSMs), namely the *local variational feature-based similarity model* (LVSM). LVSM extends linear similarity functions to non-linear ones by learning a global similarity function via a variational auto-encoder (VAE) [116]. LVSM also identifies user groups and learns the corresponding local similarity functions.

Figure 5.2 gives an illustrative example to describe how LVSM works. Figure 5.2(a) depicts a rating matrix, where rows are users and columns are items. The doted areas indicate the rated items by users. New items are on the right of the red dashed line. For the rated items on the left of the red dashed line, some have been rated by all users (global), some have been rated only by users in group A or group B (local). Given a user u and her history of rated items, LVSM calculates local and global similarities between the new item i and user rated items $1, 5, 9$ (Figure 5.2(b)). Here, $s_{ji}^{(1)}, s_{ji}^{(2)}$ are the local similarities between i and j based on users from group A and B; $s_{ji}^{(0)}$ is the

global similarity. We assume user u is from group A. Thus we form the prediction as $\tilde{y}_{ui} = \alpha_{ui}^{(1)} + \alpha_{ui}^{(0)} = \sum_{j\in\{1,5,9\}}(s_{ji}^{(1)} + s_{ji}^{(0)})$.

5.4.2 Model description

Modeling ratings. We start by modeling ratings. As we assume that user ratings are binarized, we define the rating y_{ui} to follow a Bernoulli distribution:

$$y_{ui} \sim Bernoulli\left(\sigma(\tilde{y}_{ui})\right), \tag{5.7}$$

where $\sigma(\cdot)$ is the sigmoid function and $\tilde{y}_{ui}$ is the predicted score. We propose to compute $\tilde{y}_{ui}$ by:

$$\tilde{y}_{ui} = \alpha_{ui}^{(0)} + \alpha_{ui}^{(z_u)},$$

where $z_u \in \{1, \ldots, c\}$ is a variable that indicates which group user u belongs to. Furthermore, $\alpha_{ui}^{(0)}$ and $\alpha_{ui}^{(z_u)}$ are the scores calculated based on the global similarity function and the z_u-th local similarity function. Following FSM, we assume that $\alpha_{ui}^{(k)}$ is calculated by aggregating item similarities:

$$\alpha_{ui}^{(k)} = \sum_{j\in\mathcal{R}_{u-i}^{+}} s_{ji}^{(k)}, \quad \forall k \in \{0, \ldots, c\}, \tag{5.8}$$

where $\mathcal{R}_{u-i}^{+}$ is the set of items that are rated by user u excluding item i, and s_{ij} is the similarity between item i and j. The motivation for excluding item i is based on the estimation constraint [107] that known rating information for a particular user-item pair y_{ui} is not used when the rating for that item is being estimated. Therefore, $\tilde{y}_{ui}$ can be computed by:

$$\tilde{y}_{ui} = \sum_{j\in\mathcal{R}_{u-i}^{+}} s_{ji}^{(0)} + s_{ji}^{(z_u)}. \tag{5.9}$$

We combine $s_{ji}^{(0)}$ and $s_{ji}^{(z_u)}$, linearly and equally, following [207], where the local and global similarity functions capture the diagonal and off-diagonal feature interactions (Eq. (5.6)). The linear combination is especially useful for inference; we can derive the expectation of $\alpha_{ui}^{(k)}$ (Eq. (5.26)).

Modeling global similarities. Inspired by Eq. (5.6), we define the global similarity function to capture feature interactions. Recently, several publications have explored deep neural networks (DNNs) to learn non-linear feature interactions [46, 94, 95, 206, 264]. However, capturing feature interactions by these methods is not suitable for the global similarity function as they do not have a Bayesian nature, which complicates combinations with item-based CF. Instead, we utilize a variational auto-encoder (VAE) [116]. Then, the global similarity function is defined as the inner product of latent item representations learned by the variational auto-encoder (VAE):

$$s_{ij}^{(0)} = \text{sim}_0(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_\rho(\boldsymbol{x}_i)^T f_\rho(\boldsymbol{x}_j) = \boldsymbol{v}_i^T \boldsymbol{v}_j, \tag{5.10}$$

where $\boldsymbol{v}_i, \boldsymbol{v}_j$ are the latent representations of item i, j, respectively; $f_\rho(\cdot)$ stands for the inference network of VAE, which is parameterized by ρ. As suggested by the VAE, we

use a unit Gaussian prior for $\boldsymbol{v}_i$:

$$\boldsymbol{v}_i \sim \mathcal{N}(0, I). \tag{5.11}$$

Note that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are used directly to calculate the similarity, rather than introducing offset variables like [134, 230]. This is because the offset cannot be inferred for new items. However, this complicates inference as the DNN is directly coupled with the model. Fortunately, we can derive an efficient inference thanks to the linear combination in Eq. (5.9).

Modeling local similarities. We define the local similarity function with respect to the k-th user group by:

$$s_{ij}^{(k)} = \text{sim}_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\omega}_k^T \left(\boldsymbol{x}_i \circ \boldsymbol{x}_j\right), \tag{5.12}$$

where $\circ$ is the element-wise product between vectors; $\boldsymbol{\omega}_k$ is the feature weight vector for the k-th user group, where we use a Gaussian prior:

$$\boldsymbol{\omega}_k \sim \mathcal{N}(0, \lambda_\omega^{-1} I), \tag{5.13}$$

Given user u, the local similarity will be calculated based on which group u belongs to, denoted by z_u. As z_u is discrete, we use a multinomial distribution for z_u. As we presume no information about which group users belong to, we assume equal probabilities:

$$z_u \sim \mathit{Multi}(1/c). \tag{5.14}$$

Modeling item features. The item feature $\boldsymbol{x}_i$ is generated from its latent representation $\boldsymbol{v}_i$ through a DNN. Let W_l and $\boldsymbol{b}_l$ the parameters associated with the l-layer of the DNN. Following [134, 230], we model W_l and $\boldsymbol{b}_l$ with a Gaussian distribution:

$$W_l \sim \mathcal{N}(0, \lambda_W^{-1} I), \quad \boldsymbol{b}_l \sim \mathcal{N}(0, \lambda_b^{-1} I). \tag{5.15}$$

The output of each layer $\boldsymbol{h}_l$ also follows a Gaussian distribution:

$$\boldsymbol{h}_l \sim \mathcal{N}(\phi(\boldsymbol{h}_{l-1}^T W_l + \boldsymbol{b}_l), I). \tag{5.16}$$

The feature $\boldsymbol{x}_i$ is generated from the last layer output $\boldsymbol{h}_L$. Depending on what type of data the item feature is, $\boldsymbol{x}_i$ can be assumed to be generated from a multivariate Bernoulli distribution if it is binary, or it can be generated from a Gaussian distribution if it is a real number:

$$\boldsymbol{x}_i \sim \begin{cases} \mathit{Bernoulli}(\sigma(\boldsymbol{h}_L)), & \text{if } \boldsymbol{x}_i \text{ is binary,} \\ \mathcal{N}(\boldsymbol{h}_L, \lambda_h^{-1} I), & \text{if } \boldsymbol{x}_i \text{ is real.} \end{cases} \tag{5.17}$$

The overall generation procedure is as follow:

1. For each layer $l = 1, \ldots, L$,
   (a) draw the parameter $W_l \sim \mathcal{N}(0, \lambda_W^{-1} I)$;
   (b) draw the bias $\boldsymbol{b}_l \sim \mathcal{N}(0, \lambda_b^{-1} I)$.
2. For each item $i \in \mathcal{I}$,

(a) draw item representation $\boldsymbol{v}_i \sim \mathcal{N}(0, I)$;
(b) draw hidden layer $\boldsymbol{h}_1 \sim \mathcal{N}(\phi(\boldsymbol{v}_i^T W_l + \boldsymbol{b}_l), \lambda_h^{-1} I)$, where $\phi(\cdot)$ is the activation function;
(c) for each layer $l = 2, \ldots, L$, draw hidden layer $\boldsymbol{h}_l \sim \mathcal{N}(\phi(\boldsymbol{h}_{i-1}^T W_l + \boldsymbol{b}_l), \lambda_h^{-1} I)$;
(d) draw item feature $\boldsymbol{x}_i \sim Bernoulli(\sigma(\boldsymbol{h}_L))$ if $\boldsymbol{x}_i$ is binary; $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{h}_L, \lambda_x^{-1} I)$ if $\boldsymbol{x}_i$ is real, where $\sigma(\cdot)$ is the sigmoid function.

3. For each user $u \in \mathcal{U}$, draw $z_u \sim Multi(1/c)$.
4. For each user group $k = 1, \ldots, c$, draw $\boldsymbol{\omega}_k \sim \mathcal{N}(0, \lambda_\omega^{-1} I)$.
5. For each user-item pair $(u, i), u \in Y$, draw $y_{ui} \sim Bernoulli\,(\sigma(\tilde{y}_{ui}))$, where $\tilde{y}_{ui}$ is calculated based on Eq. (5.9).

Once the model is optimized, we can predict the score of a new item i for user u throughout the inference of user rating $\tilde{y}_{ui}$.

5.5 Model Optimization

In this section, we describe an optimization method for LVSM, i.e., how to optimize the parameters $\Omega = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c\}$ for the similarity functions and the parameter θ for the generation network. Let $\Theta = \{\Omega, \theta\}$. We perform maximum a posteriori (MAP) estimation to infer LVSM by optimizing the following posterior:

$$\begin{aligned} p(\Theta \mid X, Y) &\simeq p(\Theta, X, Y) \\ &= p(X, Y \mid \Theta) p(\Theta) \\ &= p(\Theta) \int_V \sum_Z p(V, Z, X, Y \mid \Theta) dV. \end{aligned} \tag{5.18}$$

The posterior in Eq. (5.18) is intractable for exact inference, as the marginalization of latent variables is extremely difficult, due to the complex entanglement of variables and the non-linear structure of the deep network. Therefore, we turn to approximate inference algorithms. Based on the idea of VAE, we perform variational inference for LVSM. We first write the log-joint likelihood of LVSM:

$$\begin{aligned} \log p(V, Z, X, Y \mid \Theta) = &\log p(Y \mid V, X, Z, \Omega) + \\ &\log p(X \mid V, \theta) + \\ &\log p(V) + \\ &\log p(Z). \end{aligned} \tag{5.19}$$

We model the variational distribution of latent variables as $q(V, Z \mid \Phi)$, where Φ is the set of variational parameters. The evidence lower bound (ELBO) [254] is given as:

$$\mathcal{L}(\Theta, \Phi; q) = \mathbb{E}_q \left[\log p(V, Z, X, Y \mid \Theta) - q(V, Z \mid \Phi)\right] + \log p(\Theta).$$

Given the ELBO, we can thus find approximate empirical Bayes estimates for LVSM via an alternating variational expectation maximization (EM) procedure that maximizes a lower bound w.r.t. the variational parameters Φ, and then, for fixed values of the variational parameters, maximizes the lower bound w.r.t. the model parameters Θ. We summarize the variational EM algorithm in Algorithm 1.

Algorithm 1: Variational EM Algorithm

$t \leftarrow 0$;
$\Theta^{(0)} \leftarrow$ randomly initialize parameters;
while not converge **do**
    $\Phi^{(t)} \leftarrow \arg\max_{\Phi} \mathcal{L}(\Theta^{(t)}, \Phi; q)$, see § 5.5.2 ; `/* E-step */`
    $\Theta^{(t+1)} \leftarrow \arg\max_{\Theta} \mathcal{L}(\Theta, \Phi^{(t)}; q)$, see § 5.5.3 ; `/* M-step */`
    $t \leftarrow t + 1$;

5.5.1 Variational inference

We discuss in detail how to derive the ELBO. For the variational distributions, we assume

$$\boldsymbol{z}_u \sim \mathit{Multi}(\boldsymbol{\pi}_u), \quad \boldsymbol{v}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\varsigma}_i^2).$$

Based on the mean-field assumption, we fully factorize $q(V, Y, Z \mid \Phi)$:

$$q(V, Y, Z \mid \Phi) = \prod_{u=1}^{m} q(\boldsymbol{z}_u \mid \boldsymbol{\pi}_u) \prod_{i=1}^{n} q(\boldsymbol{v}_i \mid \boldsymbol{\mu}_i, \boldsymbol{\varsigma}_i^2),$$

where $\pi = \{\boldsymbol{\pi}_u\}$, $\rho = \{\boldsymbol{\mu}_i, \boldsymbol{\varsigma}_i^2\}$ are the free variational parameters. The number of parameters to optimize grows with the number of users and items, which becomes a bottleneck for real-world applications with millions of users and items. To address this issue, we utilize a variational auto-encoder (VAE) [116] to replace individual parameters $\{\boldsymbol{\mu}_i, \boldsymbol{\varsigma}_i\}$ with a data-dependent function through an inference network parameterized by ρ, i.e., $f_\rho(\boldsymbol{x}_i)$, where ρ is independent of samples and thus the scale of ρ is free from n; ρ is the parameters of inference network, which is designed to have an identical neural network structure with the generation network parameterized by θ.

Therefore, the ELBO is given as:

$$\begin{aligned}\mathcal{L}(q; \Theta, \Phi) = &\sum_{u=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{c} \pi_{uk} \mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right] + \\ &\sum_{i=1}^{n} \mathbb{E}_{q_\rho}\left[\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta)\right] - \mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right) + \\ &\sum_{u=1}^{m}\sum_{k=1}^{c} \pi_{uk}\left(\log p(z_u) - \log \pi_{uk}\right) + \log p(\Theta),\end{aligned} \tag{5.20}$$

where q_ρ is an abbreviation for $q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho)$.

We start from deriving $\mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right]$, which is in the first line of Eq. (5.20):

$$\begin{aligned}\mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right] &= \mathbb{E}_{q_\rho}\left[y_{ui} \log \sigma(\tilde{y}_{ui}^{(k)}) + (1 - y_{ui}) \log(1 - \sigma(\tilde{y}_{ui}^{(k)}))\right] \\ &= y_{ui} \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\} + 1)\right],\end{aligned} \tag{5.21}$$

where $\tilde{y}_{ui}^{(k)}$ is calculated by Eq. (5.9) with $z_u = k$. It is not easy to infer $\mathbb{E}_{q_\rho}[\log(\exp\{\tilde{y}_{ui}^{(k)}\} + 1)]$. Therefore, we approximate it as follows:

$$\begin{aligned}\mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\}+1)\right] &\approx \mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\})\right] \\ &= \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \log\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\} \\ &\approx \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right).\end{aligned} \tag{5.22}$$

We then derive the expectation $\mathbb{E}_{q_\rho}[\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta)]$, which is the first term in the second line of Eq. (5.20). It is problematic to derive $\mathbb{E}_{q_\rho}[\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta)]$ due to the non-linear transformation within the inference network parameterized by ρ. While we can obtain an unbiased estimate of it by sampling $\boldsymbol{v}_i \sim q_\rho$ and perform stochastic gradient ascent to optimize it, the challenge is that we cannot trivially take gradients with respect to ρ through this sampling process. Therefore, we apply the re-parameterization trick [116], which works as follows in this setting: we first draw a sample $\boldsymbol{\epsilon}^{(l)}$, which is independent from φ and $\boldsymbol{x}_i$, and then re-parameterize $\boldsymbol{v}_i$ as follows:

$$\begin{aligned}&\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, I), \\ &\boldsymbol{v}_i^{(l)} = \boldsymbol{\mu}_i + \boldsymbol{\epsilon}^{(l)} \circ \boldsymbol{\varsigma}_i^2.\end{aligned} \tag{5.23}$$

The $\mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right)$, which is the second term in the second line of Eq. (5.20), has an analytical solution:

$$\mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right) = \frac{1}{2}\left(2\log(\boldsymbol{\varsigma}_i) - \boldsymbol{\mu}_i^2 - \boldsymbol{\varsigma}_i^2\right). \tag{5.24}$$

Putting Eq. (5.21), (5.22), (5.23) and (5.24) together, we can rewrite the ELBO in Eq. (5.20) as:

$$\begin{aligned}\mathcal{L}(q;\Theta,\Phi) \simeq &\sum_{u=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{c}\pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right)\right] + \\ &\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n}\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i^{(l)}, \theta) - \frac{1}{2}\sum_{i=1}^{n}\left(2\log(\boldsymbol{\varsigma}_i) - \boldsymbol{\mu}_i^2 - \boldsymbol{\varsigma}_i^2\right) + \\ &\log p(\Theta) - \sum_{u=1}^{m}\sum_{k=1}^{c}\pi_{uk}\log\pi_{uk},\end{aligned} \tag{5.25}$$

where

$$\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \boldsymbol{\omega}_k^T(\boldsymbol{x}_u \circ \boldsymbol{x}_i) + \mathbb{E}_{q_\rho}\left[\boldsymbol{v}_i\right]^T \sum\nolimits_{k\in\mathcal{R}_{u-i}^+}\mathbb{E}_{q_\rho}\left[\boldsymbol{v}_j\right]. \tag{5.26}$$

The reason that we can write $\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]$ as Eq. (5.26) is that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are i.i.d. samples generated form the inference network. It is worth noting that $\mathbb{E}_{q_\rho}\left[\boldsymbol{v}_i\right] = \boldsymbol{\mu}_i, \mathbb{E}_{q_\rho}\left[\boldsymbol{v}_j\right] = \boldsymbol{\mu}_j$.

Based on the variational inference in this section, we can detail the variational E-step and M-step, respectively in § 5.5.2 and § 5.5.3.

5.5.2 Variational E-step

We update the variational parameter $\pi = \{\boldsymbol{\pi}_u\}$ in the E-step. We isolate the optimization problem for $\boldsymbol{\pi}_u$ as:

$$\min_{\boldsymbol{\pi}_u} \quad \sum_{k=1}^{c} \pi_{uk}\gamma_{uk} - \pi_{uk}\log\pi_{uk},$$

where $\sum_{k=1}^{c}\pi_{uk} = 1$ and

$$\gamma_{uk} = \sum_{i=1}^{n} \pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\} + 1\right)\right]. \tag{5.27}$$

By introducing the Lagrange multiplier λ_u, we have:

$$\mathcal{L} = \sum_{k=1}^{c}(\pi_{uk}\gamma_{uk} - \pi_{uk}\log\pi_{uk}) - \lambda_u(\sum_{k=1}^{c}\pi_{uk} - 1).$$

The derivative of $\mathcal{L}$ over π_{uk} is

$$\nabla_{\pi_{uk}}\mathcal{L} = \gamma_{uk} - \log\pi_{uk} - \lambda_u - 1.$$

Applying the Karush-Kuhn-Tucker (KKT) first-order optimality conditions, we have:

$$\pi_{uk} = \frac{\exp(\gamma_{uk} - 1)}{\sum_{k=1}^{c}\exp(\gamma_{uk} - 1)},$$

which provides the desired closed-form.

5.5.3 Variational M-step

We update the parameters of VAE (ρ and θ) and the parameters of local similarity functions ($\Omega = \{\boldsymbol{\omega}_k\}$) in the M-step. We propose to optimize these parameters through stochastic gradient ascent. At each time we select a user u and an item i. We write $\mathcal{L}_{ui}$ to denote the loss of ELBO regarding u, i:

$$\begin{aligned}\mathcal{L}_{ui} = &\sum_{k=1}^{c}\pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\} + 1\right)\right] + \\ &\sum_{j\in\mathcal{R}_u^+}\left[\mathbb{E}_{q_\varphi}\left[\log p(\boldsymbol{x}_j \mid \boldsymbol{v}_j, \theta)\right] - \frac{1}{2}\left(2\log(\boldsymbol{\varsigma}_j) - \boldsymbol{\mu}_j^2 - \boldsymbol{\varsigma}_j^2\right)\right].\end{aligned} \tag{5.28}$$

The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\mu}_j, \forall j \in \mathcal{R}_u^+$ is

$$\begin{aligned}\frac{\partial\mathcal{L}_{ui}}{\partial\boldsymbol{\mu}_j} = &\sum_{k=1}^{c}\pi_{uk}\left[y_{ui} - \sigma(\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}])\right]\frac{\partial\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]}{\partial\boldsymbol{\mu}_j} + \\ &\frac{1}{L}\sum_{l=1}^{L}\frac{\partial\log p(\boldsymbol{x}_j \mid f_\theta(\boldsymbol{v}_j^{(l)}))}{\partial\boldsymbol{v}_j^{(l)}} + \boldsymbol{\mu}_j,\end{aligned} \tag{5.29}$$

where

$$\frac{\partial \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]}{\partial \boldsymbol{\mu}_i} = \sum_{j \in \mathcal{R}_{u-i}^+} \boldsymbol{\mu}_j,$$

$$\frac{\partial \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]}{\partial \boldsymbol{\mu}_j} = \boldsymbol{\mu}_i, \quad j \in \mathcal{R}_{u-i}^+.$$

The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\varsigma}_j, \forall j \in \mathcal{R}_u^+$ is:

$$\frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\varsigma}_j} = \frac{1}{L} \sum_{l=1}^{L} \frac{\partial \log p(\boldsymbol{x}_j \mid f_\theta(\boldsymbol{v}_j^{(l)}))}{\partial \boldsymbol{v}_j^{(l)}} \circ \boldsymbol{\epsilon}^{(l)} - \frac{1}{\boldsymbol{\varsigma}_i} + \boldsymbol{\varsigma}_i. \tag{5.30}$$

The parameters of the generation network (θ) and the inference network (ρ) can be updated through backpropagation, once $\boldsymbol{\mu}_j$ and $\boldsymbol{\varsigma}_j$ have been updated. The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\omega}_k$ is:

$$\frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\omega}_k} = -\pi_{uk} \left[y_{ui} - \sigma(\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]) \right] (\boldsymbol{x}_u \circ \boldsymbol{x}_i). \tag{5.31}$$

Algorithm 2 summarizes the variational M-step.

Algorithm 2: Variational M-step

while not converged **do**
  for user $u \in \mathcal{U}$ **do**
    $i \leftarrow$ randomly select item from $\mathcal{R}_u^+ \cup \mathcal{R}_u^-$;
    for item $j \in \mathcal{R}_u^+$ **do**
      $\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j \leftarrow$ generate through inference network;
      for $l = 1$ **to** L **do**
        $\boldsymbol{v}_j^{(l)} \leftarrow$ sample according to Eq. (5.23);
      $\frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\mu}_j} \leftarrow$ calculate according to Eq. (5.29);
      $\frac{\partial \mathcal{L}_{ij}}{\partial \boldsymbol{\varsigma}_j} \leftarrow$ calculate according to Eq. (5.30);
      $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \eta \frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\mu}_j}$;
      $\boldsymbol{\sigma}_j \leftarrow \boldsymbol{\sigma}_j + \eta \frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\sigma}_j}$;
    for $k = 1$ **to** c **do**
      $\frac{\partial \mathcal{L}_{ij}}{\partial \boldsymbol{\omega}_k} \leftarrow$ calculate according to Eq. (5.31);
      $\boldsymbol{\omega}_k \leftarrow \boldsymbol{\omega}_k + \eta \left(\frac{\partial \mathcal{L}_{ui}}{\partial \boldsymbol{\omega}_k} - \lambda_\omega \boldsymbol{\omega}_k \right)$;

5.5.4 Computational analysis

We analyze the complexity of optimizing LVSM. In the variational E-step (§ 5.5.2), we calculate π_u based on Eq. (5.27), before which we calculate α_u based on Eq. (5.26). Since the calculation of π_{u_1} and π_{u_2} are independent, $\forall u_1 \neq u_2 \in \mathcal{U}$, we can optimize

the variational parameter π_u in parallel. Therefore, we only analyze the complexity of computing π_u for one user, which is $O(n(d + h|\mathcal{R}^+_{u-i}|))$, where h is the size of item representations.

As the variational M-step (§ 5.5.3) is optimized via stochastic optimization, we analyze the complexity of evaluating a single sample, a user-item pair (u, i), i.e., the complexity of calculating Eq. (5.28), which is $O(c(d+h|\mathcal{R}^+_{u-i}|)+2|\mathcal{R}^+_u|\sum_{l=1}^{L}(h_{l-1}h_l))$. Here, h_l denotes the size of l-th layer, where $h_0 = d$ and $h_L = h$, and the term $2\sum_{l=1}^{L}(h_{l-1}d_l)$ is included for the complexity of VAE, which has an inference network and a generation network with an identical network structure.

5.6 Experimental Setup

5.6.1 Research questions

We seek to answer the following research questions:

RQ4.1 How does LVSM perform on the item cold-start top-N recommendation task?
RQ4.2 Does modeling local and global similarities help to improve performance?
RQ4.3 What is the impact of feature sparsity on the recommendation performance?
RQ4.4 What is the effect of the fraction of cold-start items on the recommendation performance?
RQ4.5 How well can LVSM perform on large-scale datasets?

5.6.2 Datasets

To answer our research questions we conduct experiments on five datasets, respectively Beauty, Games, Sports, Kindle, CiteULike article (CUL-a) and CiteULike tag (CUL-t). Beauty, Games, Sports and Kindle are constructed from different categories of Amazon products [157]. For each category, the original dataset contains transactions between users and items, indicating implicit user feedback. We convert the multivariate rating values to 1s and filtered out less popular product items and users that appeared less than three transactions to construct the implicit rating matrix. For each dataset, we use the product reviews as the feature of the product items. We extract unigram features from the review articles and remove stopwords. For Beauty, Games, Sports, we select the most frequent 8,000 features as the item features and represent each product item as a bag-of-words feature vector, where feature value is binarized. We retain the original features of Kindle as we evaluate scalability of LVSM on Kindle.

CUL-a and CUL-t are datasets of user libraries of articles with different scales and degrees of sparsity obtained from CiteULike.[1] The first dataset, CUL-a, has been collected by Wang et al. [229]. The second dataset, CUL-t, has been independently collected by Wang et al. [231] and is even larger and sparser. Each article in the two datasets has a title and abstract. The content information of the articles is the concatenation of the titles and abstracts. We follow the same procedure as in [229] to preprocess the text content information. After removing stop words, the vocabulary for each dataset is selected according to the tf-idf value of each word.

The statistics of the datasets are presented in Table 5.2. $\#User$, $\#Item$ and $\#Feature$ denote the number of users, items and features, respectively. The rating

[1] http://www.citeulike.org

density is calculated as:

$$Rating\ density = \frac{\#Rating}{\#User \times \#Item},$$

where $\#Rating$ is the number of interactions between user and item. It is common that values of item features are missing, especially when an item feature is high-dimensional, e.g., the bag-of-words representation from a text. We write 0 for missing feature values. Therefore, we can also measure the density of features:

$$Feature\ sparsity = \frac{\#Nonzero}{\#Item \times \#Feature},$$

where $\#Nonzero$ is the number of non-zero feature values.

Table 5.2: Statistics of the datasets used.

Dataset	*#User*	*#Item*	*#Feature*	*Rating density*	*Feature density*	*Feature type*
Beauty	5,083	11,909	8,000	0.150%	0.528%	binary
Games	6,255	10,672	8,000	0.180%	0.324%	binary
Sports	6,174	13,257	8,000	0.116%	0.665%	binary
Kindle	26,555	22,203	11,308	0.085%	2.011%	binary
CUL-a	5,480	11,564	7,988	0.276%	0.838%	tf-idf
CUL-t	7,947	7,582	7,715	0.162%	0.224%	tf-idf

Note that we binarize feature values for the Amazon datasets and calculate tf-idf for CiteULike. The reason is that user reviews are generally noisy in terms of how they describe items, which might harm the performance of recommendation. We binarize item features for the Amazon datasets to overcome noise. On the other hand, the features for CiteULike are extracted from scientific papers, which we assume to be qualified and relevant as item features. Tf-idf values could well benefit the recommendations. It is also worth noting that tf-idf values are only suitable for text features, where binary values are applicable for other features, e.g., item attributes, image pixels, etc. Experimenting with binary values can demonstrate the generality of the proposed model.

5.6.3 Evaluation protocol

We follow the evaluation methodology of [71, 207] to evaluate the performance of item cold-start top-N recommendation. Specifically, we split the user rating matrix Y into Y_{train}, Y_{valid} and Y_{test}, respectively, for training, validating and testing. We assume that each subset of ratings contains non-overlapping items (columns) of Y, so that we can evaluate the performance of recommending new items as users in Y_{train} do not have any preferences for items in Y_{valid} and Y_{test}. In this chapter, we randomly select 80%, 10% and 10% items for Y_{train}, Y_{valid} and Y_{test}, respectively. For each user, the cold-start items are sorted in decreasing order and the first N items are returned as the top-N recommendations for that user. The list of recommended items is validated with Y_{valid} and evaluated with Y_{test} using two metrics: recall at N (Rec@N) and discounted cumulative gain at N (DCG@N). Given the list of top-N recommended items for user u, Rec@N measures how many of the items liked by u appeared in that

list, whereas the DCG@N measures how high the relevant items were placed in the list. Rec@N and DCG@N are defined as follows:

$$\text{Rec@}N = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|},$$
$$\text{DCG@}N = imp_1 + \sum_{i=2}^{N} \frac{imp_i}{\log_2(i)},$$

where the importance score imp_i of the items with rank i in the top-N list is:

$$imp_i = \begin{cases} \frac{1}{N}, & \text{if the item at rank } i \text{ is relevant,} \\ 0, & \text{otherwise.} \end{cases}$$

The main difference between Rec@N and DCG@N is that DCG@N is sensitive to the rank of the items in the top-N list. Both Rec@N and the DCG@N are computed for each user and then averaged over all users.

5.6.4 Methods used for comparison

We evaluate LVSM by comparing it against eight other feature-based models for recommending new items. We use the categories of models introduced in § 5.3.1 to characterize the methods we consider.

– *Feature-based singular value decomposition* (SVDFeature) [34]: A feature-based matrix factorization method. For item cold-start recommendation, the rating score y_{ui} is estimated as:

$$\tilde{y}_{ui} = b_u + \boldsymbol{b}_i^T \boldsymbol{x}_i + \boldsymbol{p}_u^T \sum_{j=1}^{d} x_{ij} \boldsymbol{w}_j = \sum_{j=0}^{d} b'_j x_{ij}, \tag{5.32}$$

where $b'_0 = b_u, b'_t = \boldsymbol{p}_u^T \boldsymbol{w}_j, j \geq 1$. Thus, feature-based singular value decomposition (SVDFeature) can be categorized as user modeling (UM). We utilize the ranking method of SVDFeature for our experiments.

– *Simple cosine-similarity* (coSim) [22]: A memory-based neighborhood method that estimates item similarities by cosine similarity based on item features. The preference score y_{ui} is estimated as:

$$\tilde{y}_{ui} = \sum_{j \in \mathcal{R}_u^+} \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{|\boldsymbol{x}_i| \, |\boldsymbol{x}_j|}. \tag{5.33}$$

– *Personalized feature weighting* (PFW) [22]: A non-collaborative technique that learns user models independently. A feature weighting vector $\boldsymbol{\omega}_u$ of length d is estimated for each user u to reflect the importance of the different item features for each user. The preference score y_{ui} of user u for item i is estimated as:

$$\tilde{y}_{ui} = \sum_{j \in \mathcal{R}_u^+} \boldsymbol{\omega}_u^T (\boldsymbol{x}_i \circ \boldsymbol{x}_j).$$

Table 5.3: Methods used for comparison.

Method	Multi-model	Collaborative	Neural	Category
SVDFeature [34]	–	✓	–	UM
LCE [198]	–	✓	–	LFM
NSPR [69]	–	✓	✓	IFM
cfVAE [134]	–	✓	✓	IFM
coSim [22]	–	–	–	
PFW [22]	–	–	–	
FBSM [207]	–	✓	–	FSM
UFSM [71]	✓	✓	–	
LVSM	✓	✓	✓	

– *Local collective embeddings* (LCE) [198]: A typical latent factor model (LFM) that collectively factorizes a user rating matrix and an item feature matrix. For a new item i with feature $\boldsymbol{x}_i$, the item factor $\boldsymbol{v}_i$ is first inferred by solving $\boldsymbol{x}_i = \boldsymbol{v}_i W^T$, where $W \in \mathbb{R}^{d \times h}$ is the feature factor.
– *Neural semantic personalized ranking* (NSPR) [69]: A typical item feature mapping (IFM) that maps item feature via a DNN. During recommendation, the expectation of item representation $\boldsymbol{v}_i$ is inferred from item feature $\boldsymbol{x}_i$ via the DNN.
– *Collaborative variational auto-encoder* (cfVAE) [134]: A state-of-the-art IFM. The difference between neural semantic personalized ranking (NSPR) and collaborative variational auto-encoder (cfVAE) is that cfVAE learns item representations from item features via a VAE.
– *User-specific feature-based similarity model* (UFSM) [71]: A feature-based similarity model (FSM) that models global aspects by learning multiple global similarity functions; the user-specific similarity function is calculated by aggregating global similarities with personalized weights.
– *Feature-based factorized bilinear similarity model* (FBSM) [207]: A FSM that models the interaction between features. The interaction matrix is further factorized to reduce the complexity. The similarity function is defined by Eq. (5.6).

We summarize these methods in Table 5.3. Note that while all methods can generate personalized recommendations, simple cosine-similarity (coSim) and PFW fail to take advantage of collaborative filtering. For PFW, UFSM and FBSM, we train both a point-wise loss function and a pair-wise loss function, where the model with the pair-wise loss function is subscripted with $_{pair}$, i.e., PFW$_{pair}$, UFSM$_{pair}$ and FBSM$_{pair}$. We train NSPR with two types of pairwise probability, respectively, logistic probability and probit probability, as proposed in [69]; we refer to the respective models as NSPR-L and NSPR-P.

5.6.5 Parameter settings

For LVSM we fix $\lambda_W = \lambda_b = \lambda_h = \lambda_\omega = 0.1$ for the prior of parameters $p(\Theta)$. We choose a two-layer multi layer perceptron (MLP) network architecture (50–10 for the inference network and 10–50 for the generation newtork) with a ReLU activation function [83]. We select a smaller network scale for CUL-t (10–5 for the inference network and 5–10 for the generation newtork) as the item feature of the dataset is extremely sparse so that algorithms easily overfit. For the number of local aspects, we

Table 5.4: Parameter settings.

Dataset	NSPR-L [69]		NSPR-P [69]		LCE [198]				cfVAE [134]	
	λ_v	λ_u	λ_v	λ_u	α	β	λ	k	λ_v	λ_u
Beauty	0.1	1	0.5	0.01	0.9	1	0.1	200	0.5	1
Games	0.9	10	0.9	1	0.5	10	1	200	0.5	1
Sports	0.9	0.1	0.5	1	0.5	1	1	500	0.9	1
CUL-a	0.5	0.1	0.5	0.01	0.9	1	0.1	500	0.5	1
CUL-t	0.9	0.01	0.1	0.01	0.9	1	1	500	0.9	1

Dataset	UFSM [71]				UFSM$_{pair}$ [71]				SVDFeature [34]		
	μ_1	μ_2	λ	c	μ_1	μ_2	λ	c	α	β	k
Beauty	0.01	1	0.1	3	1	1	0.1	2	0.01	1	50
Games	1	0.01	1	1	0.1	1	0.1	2	0.1	0.1	50
Sports	1	1	0.01	4	0.1	0.01	1	6	0.01	0.01	50
CUL-a	1	0.01	1	6	0.1	1	0.01	6	0.01	0.01	500
CUL-t	1	0.01	1	5	0.1	0.01	0.1	5	0.01	0.01	500

Dataset	FBSM [207]			FBSM$_{pair}$ [69]			PFW [22]	PFW$_{pair}$ [22]
	β	λ	k	β	λ	k	μ	μ
Beauty	0.1	0.01	1	0.1	0.01	5	10	1
Games	0.01	1	10	0.1	0.01	5	10	0.01
Sports	0.01	0.1	5	1	0.01	10	0.1	10
CUL-a	0.01	1	1	0.1	0.1	1	0.1	0.01
CUL-t	0.01	1	1	0.1	0.1	1	1	1

try $c = 1, 2, 3$, respectively, and denote the corresponding model as LVSM$_1$, LVSM$_2$ and LVSM$_3$.

We select the same network structure for cfVAE as it also utilizes a VAE. We select a larger scale for the network of VAE for cfVAE as it is used for learning item factors (100–50 for the inference network and 50–100 for the generation network). We also try to find a smaller network scale for cfVAE on CUL-t, but find out that it is not possible to improve the performance. Therefore, we keep the same network scale for cfVAE over all datasets. Similarly, we also select a two-layer perceptron (100–50) for NSPR.

For the methods used for comparison, we select the hyper-parameters by Rec@10 on the validation set Y_{valid}. A detailed list of parameter settings is included in Table 5.4; there, we tune the ℓ_2-norm regularization parameter μ for PFW, which is selected from $\{0.01, 0.1, 1, 10\}$. We tune μ_1, μ_2, λ and l for UFSM, where μ_1, μ_2, λ are selected from $\{0.01, 0.1, 1, 10\}$ and l from $\{1, 2, 3, 4, 5, 6\}$. We tune β, λ and k for FBSM, where β, λ are selected from $\{0.01, 0.1, 1, 10\}$ and k from $\{1, 5, 10, 20\}$. We tune λ_1 and λ_2 for SVDFeature, which, respectively, stand for the regularization parameter of the user factor and the item factor; both λ_1 and λ_2 are selected from $\{0.01, 0.1, 1, 10\}$. We also tune the latent dimension k for SVDFeature, which is selected from $\{50, 100, 200, 500\}$. We tune α, β, λ for local collective embeddings (LCE), where α balances the importance of user rating and item feature, which is within $[0, 1]$; we select α from $\{0.1, 0.2, \ldots, 0.9\}$ and β, λ from $\{0.01, 0.1, 1, 10\}$. We also tune the latent dimension k for LCE, which is selected from $\{50, 100, 200, 500\}$. We tune λ_v and λ_u for cfVAE; λ_v controls the contribution of latent item representation to item factor, which we select from $\{0.1, 0.2, \ldots, 0.9\}$; λ_u is the regularization for user factor, which we selected from $\{0.01, 0.1, 1, 10\}$. Similarly, we also tune λ_v and λ_u for cfVAE, which are selected from $\{0.1, 0.2, \ldots, 0.9\}$ and $\{0.01, 0.1, 1, 10\}$ respectively.

Table 5.5: Performance of recommending top-N new items.

Beauty	Rec@5	Rec@10	Rec@15	Rec@20	DCG@5	DCG@10	DCG@15	DCG@20
coSim [22]	0.1045	0.1490	0.1885	0.2245	0.0380	0.0225	0.0168	0.0137
SVDFeature [34]	0.0035	0.0261	0.0273	0.0297	0.0013	0.0026	0.0018	0.0014
NSPR-L [69]	0.0378	0.0535	0.0707	0.0814	0.0145	0.0086	0.0065	0.0052
NSPR-P [69]	0.0083	0.0275	0.0415	0.0596	0.0026	0.0029	0.0026	0.0024
cfVAE [34]	0.0932	0.1318	0.1584	0.1816	0.0346	0.0206	0.0151	0.0121
LCE [198]	0.0996	0.1408	0.1801	0.2065	0.0367	0.0220	0.0163	0.0130
PFW [22]	0.1009	0.1445	0.1848	0.2179	0.0361	0.0216	0.0162	0.0132
PFW$_{pair}$ [22]	0.1005	0.1501	0.1893	0.2233	0.0362	0.0219	0.0164	0.0133
UFSM [71]	0.1065	0.1486	0.1870	0.2220	0.0380	0.0223	0.0166	0.0135
UFSM$_{pair}$ [71]	0.1000	0.1478	0.1864	0.2220	0.0360	0.0218	0.0162	0.0132
FBSM [207]	0.0501	0.0836	0.1180	0.1443	0.0177	0.0117	0.0092	0.0079
FBSM$_{pair}$ [207]	0.0503	0.0849	0.1201	0.1488	0.0188	0.0120	0.0096	0.0079
LVSM$_1$	0.1061	0.1544	0.1899	0.2242	0.0396	0.0239	0.0176	**0.0143****
LVSM$_2$	0.1077	0.1523	0.1904	0.2232	0.0395	0.0235	0.0175	0.0141
LVSM$_3$	**0.1100**	**0.1579***	**0.1940**	**0.2276**	**0.0401***	**0.0242****	**0.0177****	**0.0143****
Games	Rec@5	Rec@10	Rec@15	Rec@20	DCG@5	DCG@10	DCG@15	DCG@20
coSim [22]	0.0653	0.1050	0.1401	0.1645	0.0216	0.0139	0.0107	0.0088
SVDFeature [34]	0.0025	0.0054	0.0086	0.0121	0.0008	0.0007	0.0005	0.0005
NSPR-L [69]	0.0080	0.0187	0.0236	0.0296	0.0027	0.0022	0.0017	0.0014
NSPR-P [69]	0.0130	0.0177	0.0232	0.0335	0.0044	0.0026	0.0020	0.0018
cfVAE [34]	0.0538	0.0854	0.1149	0.1381	0.0175	0.0113	0.0088	0.0072
LCE [198]	0.0475	0.0768	0.1028	0.1274	0.0161	0.0106	0.0082	0.0068
PFW [22]	0.0611	0.1015	0.1282	0.1535	0.0202	0.0132	0.0100	0.0083
PFW$_{pair}$ [22]	0.0576	0.0968	0.1261	0.1524	0.0194	0.0128	0.0098	0.0081
UFSM [71]	0.0596	0.0974	0.1291	0.1553	0.0196	0.0126	0.0098	0.0081
UFSM$_{pair}$ [71]	0.0621	0.0987	0.1311	0.1585	0.0210	0.0134	0.0103	0.0085
FBSM [207]	**0.0739**	0.1007	0.1414	0.1699	0.0236	0.0141	0.0110	0.0091

FBSM$_{pair}$ [207]	**0.0739**	0.1007	0.1402	0.1699	0.0232	0.0139	0.0108	0.0090
LVSM$_1$	0.0732	0.1084	0.1420	0.1704	0.0242	0.0148	0.0112	0.0092
LVSM$_2$	**0.0739**	0.1078	0.1387	0.1698	0.0238	0.0146	0.0111	0.0092
LVSM$_3$	0.0736	**0.1112***	**0.1441**	**0.1757***	**0.0243**	**0.0151****	**0.0114****	**0.0094****
Sports	Rec@5	Rec@10	Rec@15	Rec@20	DCG@5	DCG@10	DCG@15	DCG@20
coSim [22]	0.0599	0.0924	0.1185	0.1402	0.0167	0.0106	0.0079	0.0065
SVDFeature [34]	0.0042	0.0078	0.0194	0.0230	0.0011	0.0007	0.0009	0.0007
NSPR-L [69]	0.0128	0.0224	0.0289	0.0380	0.0040	0.0026	0.0020	0.0017
NSPR-P [69]	0.0141	0.0223	0.0318	0.0380	0.0043	0.0027	0.0022	0.0017
cfVAE [34]	**0.0648**	0.0875	0.1004	0.1124	**0.0202**	**0.0116**	0.0082	0.0064
LCE [198]	0.0539	0.0866	0.1092	0.1294	0.0165	0.0104	0.0077	0.0063
PFW [22]	0.0554	0.0871	0.1121	0.1362	0.0162	0.0102	0.0077	0.0063
PFW$_{pair}$ [22]	0.0555	0.0881	0.1125	0.1368	0.0163	0.0103	0.0077	0.0064
UFSM [71]	0.0575	0.0917	0.1178	0.1387	0.0160	0.0103	0.0077	0.0063
UFSM$_{pair}$ [71]	0.0561	0.0919	0.1168	0.1376	0.0162	0.0105	0.0079	0.0064
FBSM [207]	0.0313	0.0550	0.0678	0.0850	0.0078	0.0055	0.0041	0.0035
FBSM$_{pair}$ [207]	0.0313	0.0550	0.0708	0.0880	0.0078	0.0055	0.0042	0.0036
LVSM$_1$	0.0618	0.0972	**0.1265****	**0.1480***	0.0184	0.0115	**0.0087**	**0.0070***
LVSM$_2$	0.0607	0.0936	0.1244	0.1467	0.0184	0.0114	0.0086	0.0069
LVSM$_3$	0.0640	**0.0981***	0.1206	0.1443	0.0187	**0.0116**	0.0086	**0.0070***
CUL-a	Rec@5	Rec@10	Rec@15	Rec@20	DCG@5	DCG@10	DCG@15	DCG@20
coSim [22]	0.1977	0.2819	0.3394	0.3840	0.0973	0.0590	0.0435	0.0348
SVDFeature [34]	0.0017	0.0065	0.0109	0.0149	0.0013	0.0013	0.0011	0.0010
NSPR-L [69]	0.0020	0.0065	0.0102	0.0120	0.0022	0.0018	0.0015	0.0012
NSPR-P [69]	0.0034	0.0087	0.0135	0.0197	0.0021	0.0017	0.0015	0.0014
cfVAE [34]	0.0348	0.0594	0.0823	0.1020	0.0255	0.0166	0.0128	0.0106
LCE [198]	0.1639	0.2572	0.3203	0.3657	0.0871	0.0556	0.0420	0.0337
PFW [22]	0.1798	0.2574	0.3146	0.3602	0.0890	0.0544	0.0403	0.0323

PFW$_{pair}$ [22]	0.1812	0.2609	0.3148	0.3558	0.0900	0.0552	0.0406	0.0325
UFSM [71]	0.1905	0.2738	0.3291	0.3724	0.0935	0.0577	0.0424	0.0339
UFSM$_{pair}$ [71]	0.1940	0.2770	0.3330	0.3766	0.0941	0.0574	0.0424	0.0340
FBSM [207]	0.0053	0.0093	0.0154	0.0212	0.0042	0.0026	0.0021	0.0018
FBSM$_{pair}$ [207]	0.0043	0.0093	0.0147	0.0170	0.0028	0.0022	0.0018	0.0014
LVSM$_1$	0.2108	0.3019	0.3633	0.4089	0.1083	0.0664	0.0487	0.0388
LVSM$_2$	**0.2226****	**0.3172****	0.3772	0.4225	0.1116	0.0683	0.0501	0.0400
LVSM$_3$	0.2173	0.3108	**0.3800****	**0.4226****	**0.1121****	**0.0684****	**0.0504****	**0.0401****
CUL-t	Rec@5	Rec@10	Rec@15	Rec@20	DCG@5	DCG@10	DCG@15	DCG@20
coSim [22]	**0.1972**	0.2728	0.3316	0.3704	**0.0629**	**0.0379**	0.0278	0.0221
SVDFeature [34]	0.0093	0.0148	0.0190	0.0304	0.0032	0.0021	0.0016	0.0015
NSPR-L [69]	0.0051	0.0097	0.0140	0.0181	0.0019	0.0014	0.0013	0.0012
NSPR-P [69]	0.0053	0.0087	0.0135	0.0176	0.0028	0.0020	0.0016	0.0013
cfVAE [34]	0.0368	0.0650	0.0927	0.1137	0.0143	0.0096	0.0078	0.0065
LCE [198]	0.1513	0.2298	0.2870	0.3253	0.0502	0.0315	0.0236	0.0189
PFW [22]	0.1777	0.2557	0.3089	0.3517	0.0562	0.0343	0.0252	0.0203
PFW$_{pair}$ [22]	0.1821	0.2559	0.3076	0.3502	0.0570	0.0344	0.0253	0.0203
UFSM [71]	0.1865	0.2671	0.3213	0.3640	0.0599	0.0364	0.0268	0.0214
UFSM$_{pair}$ [71]	0.1920	0.2740	0.3264	0.3691	0.0612	0.0370	0.0270	0.0216
FBSM [207]	0.0033	0.0141	0.0191	0.0245	0.0011	0.0013	0.0010	0.0010
FBSM$_{pair}$ [207]	0.0029	0.0135	0.0229	0.0271	0.0010	0.0011	0.0011	0.0009
LVSM$_1$	0.1935	0.2760	0.3335	0.3762	0.0616	0.0375	0.0276	0.0219
LVSM$_2$	0.1918	0.2769	0.3318	0.3693	0.0591	0.0363	0.0266	0.0211
LVSM$_3$	0.1923	**0.2805**	**0.3379**	**0.3834**	0.0615	**0.0379**	**0.0280**	**0.0223**

We show the best score in **boldface** and the second best is underlined. We conducted two-sided tests for the null hypothesis that the best and the second best have identical average values. We attach asterisks to the best score if the improvement over the second best is statistically significant; we use a single asterisk * if $p < 0.05$ and two asterisks ** if $p < 0.01$. Note that we do not take LVSM for both best and second best, that is, if LVSM$_1$ performs the best in one metric, we do not take LVSM$_2$ as the second best in the same metric even if it is. This is because we want to show whether the improvement of LVSM is significant over other baselines.

5.6.6 Experiments

To answer our research questions, we conduct different set of experiments:
- To answer RQ4.1, we generate the top-N recommendations of new items by comparing all baselines with LVSM on the Beauty, Games, Sports, CUL-a and CUL-t datasets (§ 5.7.1).
- To answer RQ4.2, we run incremental experiments on the Beauty, Games, CUL-a and CUL-t datasets to evaluate the modeling of local and global similarity functions of FSMs. We also show how user sub-groups learned by LVSM look like through a qualitative example on Games (§ 5.7.2).
- To answer RQ4.3, we manually sparsify the Beauty dataset and evaluate the performance on the Beauty dataset with different feature densities (§ 5.7.3).
- To answer RQ4.4, we vary the number of new items and run experiments on the Sports dataset (§ 5.7.4).
- To answer RQ4.5, we compare both efficiency and accuracy of LVSM with other FSMs on the Kindle dataset (§ 5.7.5).

5.7 Results and Analysis

In this section we report on the results of our experiments and answer our research questions.

5.7.1 Performance comparison

To address RQ4.1, we present an overall comparison of the top-N recommenders that we consider. We report the recommendation results in terms of Rec@N and DCG@N in Table 5.5, where respectively 5, 10, 15, 20 items are recommended.

We organize the discussion of the results by dataset. We first look at the Beauty dataset. We note that LVSM_3 dominates the performance on all metrics. The second best results are achieved by coSim, PFW and UFSM. The improvement of LVSM_3 over the second best is significant in terms of Rec@10, DCG@5, DCG@10, DCG@15 and DCG@20. This demonstrates the superiority of FSMs for item cold-start top-N recommendation. This also shows the power of LVSM as it significantly improves over the state-of-the-art FSMs.

Next, we look at the Games dataset, which shows similar results. The difference is that FBSM performs the second best for this task and FBSM achieves a comparable performance as LVSM_2 on Rec@5. The Games dataset has the sparsest item feature but least sparse ratings of all Amazon datasets. This characteristic of the Games dataset benefits FBSM as it models the interaction among features to overcome feature sparsity while the least sparse rating helps it to learn feature interactions. LVSM can also benefit from the characteristics of the Games dataset as the modeling of global item similarities captures the feature interaction in a more advanced way. The improvement of LVSM over FBSM is significant on Rec@10, Rec@20, DCG@10, DCG@15 and DCG@20.

We turn to the Sports dataset. LVSM generally has an advantage, but is outperformed by cfVAE in terms of Rec@5 and DCG@5. As shown in Table 5.2, the ratings of the Sports dataset are the sparsest among all the datasets that we consider. LVSM and cfVAE show their advantage of utilizing a VAE by benefiting from the automatic denoising property of the VAE. While cfVAE shows promising results when $N = 5$,

Table 5.6: Different ways of modeling item similarity functions.

Method	LVSM	LSM	UFSM
Global similarity	1	0	c
Lobal similarity	c	c	0

the effectiveness of cfVAE drops as N increases. Also, a comparison between cfVAE and NSPR reveals that a VAE is a better tool for extracting information from raw features than DNN. The superiority of LVSM is well demonstrated on Sports when N is getting larger. The improvement of LVSM over the second best approach is significant in terms of Rec@10, Rec@15, Rec@20 and DCG@20. With insufficient label information, methods that better extract information from item feature will show their advancement on Sports.

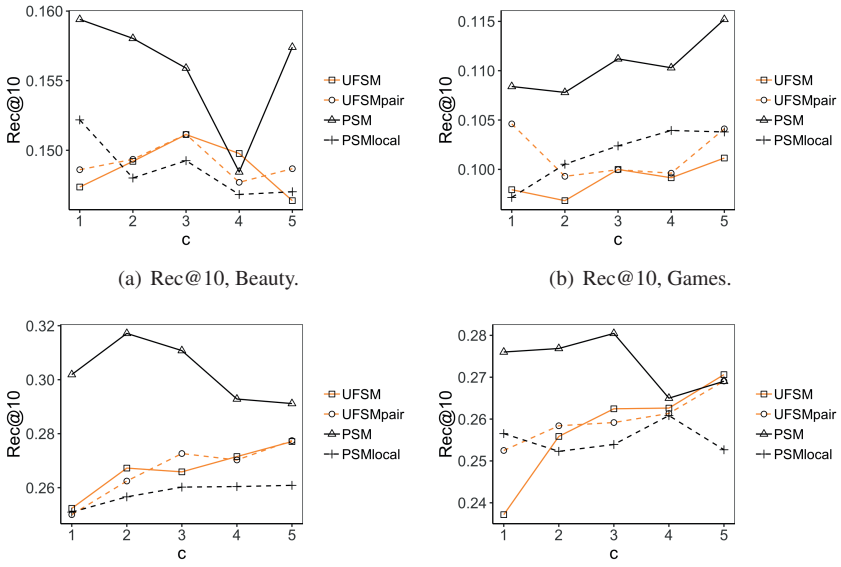
Next, we consider CUL-a. As CiteULike has better formatted features than the Amazon datasets to measure item similarity, we can expect a better performance achieved by FSMs. Surprisingly, although FSMs perform better than other methods, it actually fails to beat the non-collaborative filtering method coSim. A possible explanation is that the features of CUL-a are well-qualified to capture item similarities, where existing FSMs reached a bottleneck to further improve the performance, due to the sparsity of ratings. However, LVSM has the ability to improve the performance over coSim by a large margin. LVSM_2 is significantly better than coSim in terms of Rec@5 and Rec@10, and LVSM_3 is significantly better than coSim in terms of every DCG metric.

Finally, we look at CUL-t. While the performance of LVSM is very promising on CUL-a, it cannot significantly improve the performance on CUL-t. LVSM is outperformed by coSim in terms of Recall@5 and DCG@5, and achieves a tie with coSim in terms of DCG@10. The improvements of LVSM over coSim in terms of Rec@10, Rec@15, Rec@20, DCG@10, DCG@15 and DCG@20 is not significant. CUL-t has the sparsest features among all datasets. All methods except coSim all include learning, which is heavily impacted by the sparsity of features. Although the performance of LVSM is not exceptional, it actually shows a good denoising ability as the performance generally is at least as good as that of coSim. In comparison, other methods, especially FBSM, perform much worse.

To summarize, LVSM has generally shown its superiority over other methods on all datasets. Except on CUL-a, the improvement of LVSM over the second best method is usually significant. On the other hand, FSMs shows a better performance than other types of method (in other categories), for the task of item cold-start top-N recommendation. cfVAE enjoys the benefits of VAE for denoising with sparse features, compared with NSPR. However, as it belongs to IFM, which is not designed for top-N recommendation task, it fails to perform well, especially on CUL-a and CUL-t. LVSM takes advantage of both FSM and VAE to yield overall better performance.

5.7.2 Effect of modeling global and local similarities

We seek to answer RQ4.2, whether modeling global and local item similarities helps to improve performance. We form another baseline local feature-based similarity model (LSM) from LVSM, which calculates local similarities only. We also compare LVSM with UFSM, which calculates global similarities only. We summarize LVSM, LSM



(a) Rec@10, Beauty.

(b) Rec@10, Games.

(c) Rec@10, CUL-a.

(d) Rec@10, CUL-t.

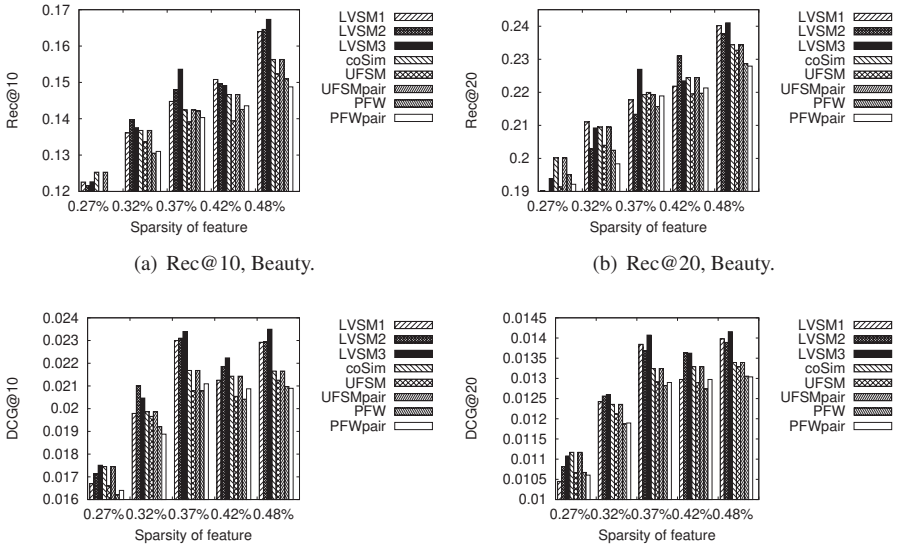
Figure 5.3: Effect of the number of similarity functions.

and UFSM in Table 5.6. We vary the number of user groups c and plot the Rec@10 scores obtained by LVSM, LSM, UFSM and UFSM$_{pair}$; see Figure 5.3.
Figure 5.3(a) displays the results on the Beauty dataset. UFSM, UFSM$_{pair}$ reach their peak performance when learning 3 global similarity functions. LVSM and LSM generally decrease their performance when modeling more local similarity functions. LSM is outperformed by UFSM, UFSM$_{pair}$ when $c \geq 2$ and LVSM is outperformed by UFSM$_{pair}$ when $c = 4$. In short, modeling global similarity functions only achieves the best performance, which shows that there may not exist user subgroups on the Beauty dataset. LVSM also shows better modeling capacity of global item similarities than UFSM.

Figure 5.3(b) shows a converse result. LVSM and LSM increase their performance by modeling local similarity functions. When $c = 5$, LVSM achieves its best performance, although the figure of LSM drops slightly. LSM outperforms UFSM and UFSM$_{pair}$ when $c \geq 2$ and LVSM further evidently improves over LSM. In short, learning local item similarity functions well captures user subsets in the Games dataset. The advantage of LVSM over UFSM is better illustrated as UFSM fails to model local similarities.

Figure 5.3(c) further demonstrates the suitability of learning item similarities with LVSM. Although LSM is outperformed by UFSM and UFSM$_{pair}$, LVSM outperforms UFSM and UFSM$_{pair}$, and the best performance is achieved when $c = 2$. We can conclude from Figure 5.3(c) that while solely modeling of local similarities is suboptimal, the integration of modeling local and global similarities well captures the essence of the application on CUL-a.

The results in Figure 5.3(d) are similar. LVSM shows better results when $c \leq 3$.



(a) Rec@10, Beauty.

(b) Rec@20, Beauty.

(c) DCG@10, Beauty.

(d) DCG@20, Beauty.

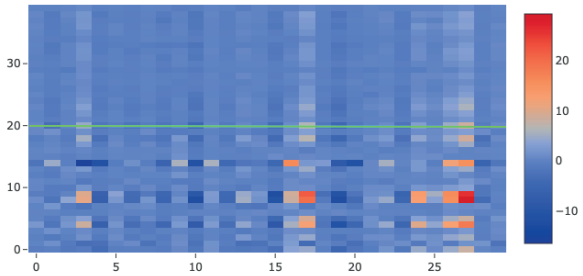
Figure 5.4: Effect of feature sparsity.

UFSM and UFSM$_{pair}$ catch up when c is increased. Note that while LSM achieves the best performance at $c = 4$, that setting is where LVSM actually generates its worst recommendation performance. This further confirms the effectiveness of jointly modeling local and global similarities, which is the advantage of bayesian graphical modeling.

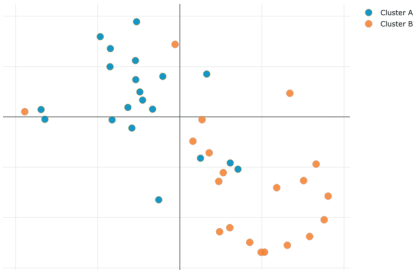
The estimation of local models is essential to the performance of LVSM. As each local model corresponds to a user group, it will be interesting to see what the learned user group looks like. Therefore, we provide a qualitative example in Figure 5.5, using the Games dataset. For the sake of obtaining a clear visualization, we consider two user groups only. We visualize the predicted scores $\tilde{y}_{ui}$ of users over new items. This is because users in the same group have similar behaviors, whereas users from different groups have different behaviors. We randomly select 30 items from all new items. For each group, we randomly select 20 users.

We visualize the predicted ratings for the 20 items in Figure 5.5(a). Clearly, users from different groups show different behaviors, illustrated by the different ratings given to the same items. Users from group A generally give lower ratings to items, compared with users from group B. Besides, similarities are clearly visible for users in group A, whereas behaviors of users from group B show some difference. We can also see the commonality in behaviors from both groups, which reflects the effect of the global similarity function.

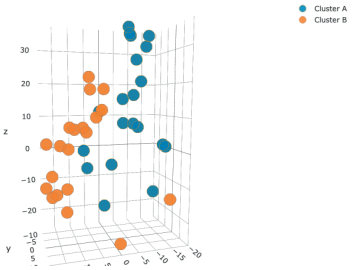
We also visualize the spatial proximity by conducting t-distributed stochastic neighbor embedding (t-SNE) [155] on the predicted rating scores. t-SNE identifies 2 and 3 main components, depicted in Figure 5.5(b) and Figure 5.5(c), respectively. Users from the two groups can be clustered with clear and different centroids.



(a) Example of predicted scores. Above the green line are the predicted ratings for users in group A and the ratings below the green line are for group B.



(b) t-SNE with 2 components.



(c) t-SNE with 3 components.

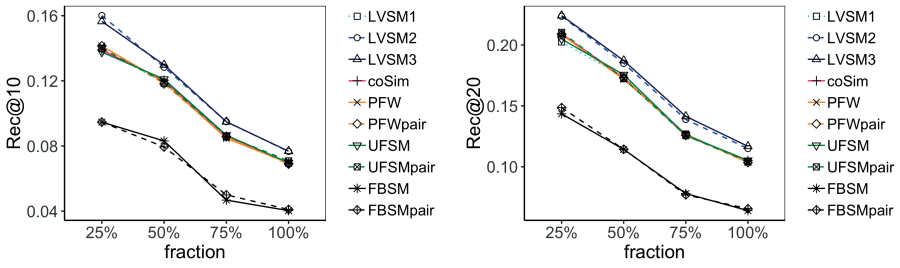
Figure 5.5: Qualitative example of user groups.

5.7.3 Effect of feature sparsity

We proceed to answer RQ4.3. We evaluate the effect of feature sparsity on the performance of recommenders. We manually sparsify item features, by randomly selecting dimensions in the feature to be excluded, where the feature density is roughly controlled as $0.27\%, 0.32\%, 0.37\%, 0.42\%, 0.48\%$. As we have already demonstrated the superiority of models in the FSM category over models in the UM, LFM and IFM categories, we only care about the impact of feature sparsity on FSMs, i.e., coSim, PFW, UFSM and LVSM. Note that we also exclude FBSM for comparison as it generates very poor recommendations when item features are even sparser. For illustration, we depict the results of a comparison in terms of Rec@10, Rec@20, DCG@10 and DCG@20 on the Beauty dataset in Figure 5.4.

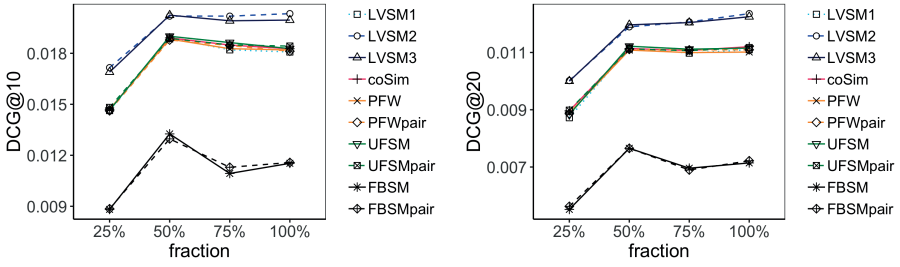
As shown in Figure 5.4(a), coSim and PFW outperform LVSM when *feature density* $= 0.27\%$. This is understandable: when item feature is extremely sparse, we have less information from data so that the simple models generally perform better, e.g., coSim. LVSM performs better when the item feature sparsity is 0.32%, 0.37%, 0.42%, 0.48%, respectively, where other models also surpass coSim and PFW. This shows that LVSM can overcome feature sparsity to a certain degree. When it is extremely sparse, we should turn to simpler models.

Similar results are shown for Rec@20 in Figure 5.4(b), where the superiority of LVSM is shown when item features are not extremely sparse. It is also interesting to see that while LVSM_2 generally shows less effective results than LVSM_1 and LVSM_3, LVSM_2 outperforms LVSM_1 and LVSM_3 when *feature sparsity* $= 0.42\%$. It seems



(a) Rec@10, Sports.

(b) Rec@20, Sports.



(c) DCG@10, Sports.

(d) DCG@20, Sports.

Figure 5.6: Effect of item cold-start.

that feature sparsity can also affect the number of local similarity functions in the data.

Next we look at the results in terms of DCG. While LVSM loses out to simple models when $\textit{feature sparsity} = 0.27\%$ in terms of Rec@10 and Rec@20; it wins back in terms of DCG@10, as shown in Figure 5.4(c). LVSM outperforms other methods with other degrees of feature sparsity. Interestingly, LVSM_1 and LVSM_2 perform even better when $\textit{feature sparsity} = 0.37\%$ than that when $\textit{feature sparsity} = 0.48\%$ and when $\textit{feature sparsity} = 0.42\%$, the performance actually degrades. We think that the dataset formed by controlling sparsity at 0.42% ignores some important features. The 0.37%-sparsity dataset might preserve these important feature and ignore some noisy features, which works similarly as feature selection.

While Figure 5.4(d) shows similar results for DCG@20; LVSM is again outperformed by simple models when $\textit{feature sparsity} = 0.27\%$. Similar to Figure (5.4(b)), LVSM_2 also outperforms LVSM_1 and LVSM_3 when $\textit{feature sparsity} = 0.42\%$, which further demonstrate the impact of feature sparsity on the number of local similarity functions to model.

5.7.4 Effect of item cold-start

Next, we turn to RQ4.4. We evaluate the effect of the fraction of cold-start items on the performance of top-N recommenders. As before, we only consider the effect on FSMs. We set different fractions of items to be cold-start items: we split Y into Y_{train}, Y_{test}, where Y_{train} contains $5/7$ items and Y_{test} contains $2/7$ items. By training the different methods on Y_{train} given the tuned parameters (Table 5.4), we test the performance of the trained model over different test set, with respectively 25%, 50%, 75% and

Table 5.7: Performance of recommending top-N new items on the Kindle dataset.

Method	Rec@10	DCG@10	E-step (secs.)	M-step/ Train (secs.)	Evaluation (secs.)	#Params
coSim	0.0305	0.0047	–	–	51.3917	–
PFW	0.0339	0.0060	–	2386.0205	48.6667	298,915,672
UFSM	0.0387	0.0053	–	101.9565	48.4056	**188,710**
FBSM	0.0323	0.0047	–	115.8321	76.6828	576,708
LVSM_1	0.0762**	0.0120**	703.7169	176.7171	46.6535	1,181,470
LVSM_2	0.0758**	0.0120**	889.5421	168.3204	46.9567	1,219,212
LVSM_3	0.0853**	0.0135**	1071.7346	165.2351	46.4882	1,256,954

100% columns of Y_{test}, e.g., $1/14, 1/7, 3/14$ and $2/7$ items. We report the result of $Rec@10, Rec@20, DCG@10, DCG@20$, respectively in Figure 5.6(a)–5.6(d).

A general trend revealed by Figure 5.6 is that the performance in terms of Recall decreases with the growth of the number of cold-start items. If we increase the number of cold-start items, the number of relevant items also increases, causing further difficulty for recommenders to identify all the relevant items. Inversely, DCG shows an increasing trend; when the number of relevant items increases, it is more likely that the relevant items appear in the recommendation list.

As shown by Figure 5.6(a) and 5.6(b), based on their performance in terms of Recall, the methods are generally categorized into three clusters. The first cluster consists of LVSM_2 and LVSM_3. The second cluster contains LVSM_1, coSim, UFSM, UFSM_{pair}, PFW and PFW_{pair}, which are inferior to the first cluster, but also provide good recommendations. The third cluster includes only FBSM and FBSM_{pair}, which is far behind the performance of the second cluster.

Unlike the performance in terms of Recall, over DCG the methods naturally cluster into two clusters, as shown by Figure 5.6(c) and 5.6(d). Besides LVSM_2 and LVSM_3 in the first cluster, other methods are all contained in the second cluster.

In short, LVSM beats other methods on all occasions of cold-start items. The superiority of LVSM is demonstrated by jointly modeling local and global similarity functions of items (LVSM_1 models only global similarities).

5.7.5 Performance on a large-scale dataset

And, finally, we turn to RQ4.5. To show the scalability of LVSM, we run experiments on the Kindle dataset. As training on the Kindle dataset is time-consuming, we do not grid-search the best parameters. Instead, we take the parameters tuned for Sports dataset as parameters for Kindle as rating sparsity of Sports is similar to Kindle. We exclude the comparison with other baselines since FSMs already show superior performance (§ 5.7.1). For the efficiency of training, we train PFW, UFSM and FBSM with a point-wise loss function. As the derived loss function for LVSM is point-wise, the comparison will be fair.

We report Rec@10 and DCG@10 in Table 5.7. As shown in the table, LVSM_1, LVSM_2 and LVSM_3 significantly and substantially outperform other FSMs. LVSM_1 and LVSM_2 show similar performances while LVSM_3 further improves over LVSM_1 and LVSM_2.[2] The effectiveness of LVSM is further confirmed on large-scale datasets,

[2]We can expect even better performances by estimating more local models. We leave further investigations for future work.

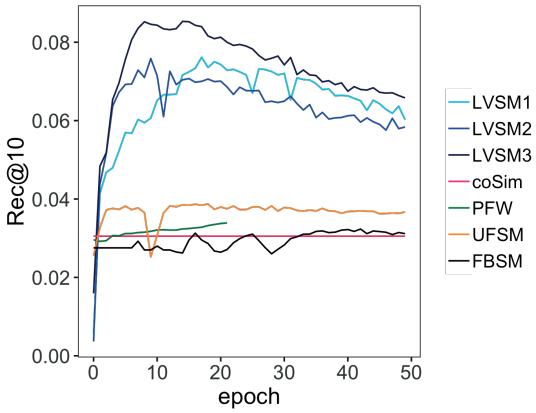


Figure 5.7: Comparison of training performance after each epoch.

showing a bigger improvement than on other datasets. The potential reason is that users' behaviors are highly diversified on the large-scale dataset, where estimating local models benefit more from the diversity. The large-scale dataset also contains more noise in item features, where the de-noising property of VAE is especially useful.

Besides, we also report the number of parameters and the time for training and evaluation for each epoch in Table 5.7. LVSM is efficient to train during M-step, which is comparable to UFSM and FBSM. E-step is a bit time-consuming. While the training time for M-step remains the same, it grows linearly with c for E-step. LVSM is also the most efficient method for evaluation. The last column shows the number of parameters of each model. coSim is a heuristic method that requires no parameters. PFW requires the most parameters as it estimates a personalized model for each user. UFSM has the least parameters, followed by FBSM. While LVSM has more parameters than UFSM and FBSM, it has far fewer parameters than PFW. Most parameters of LVSM are coming from VAE. Although the number of parameters increases linearly with c, the increase is marginally. It is noteworthy that the number of parameters of LVSM is roughly two times the number of FBSM. This is because VAE has two identical networks (inference network and generation network) and the first-layer contains most of the parameters.

Figure 5.7 depicts the performance of Rec@10 of the compared methods after each epoch. LVSM shows rapid growth in the beginning, reaching the peak around 10 to 20 epochs. After that, LVSM suffers a slight and steady drop. While LVSM_2 (blue line) performs better than LVSM_1 (cyan line) in the beginning, it is outperformed by LVSM_1 after around 15 epochs. This indicates that LVSM may be overfitting on the large-scale and sparse dataset and that it is necessary to apply an early stopping strategy both for efficiency and efficacy. In comparison, other FSMs converge shortly and stabilized quickly. However, they show limited potential to provide better recommendations. Although PFW shows a steady growth, we only report it for 20 epochs (green line). This is because we train each method for at most 24 hours and training PFW is very time-consuming.

In short, LVSM shows good performance also on large-scale datasets, both in terms of effectiveness and efficiency.

5.8 Summary

In this chapter, We have answered **RQ4** by studying a feature-based similarity model to recommend top-N new items. We have revisited the task of recommending top-N new items. We have proposed a local variational feature-based similarity model (LVSM) to address this problem by exploiting high-dimensional and sparse item features. Our method is a Bayesian generative model that jointly unifies item representation learning, user clustering, and item collaborative filtering. LVSM can learn deep representations from item features to facilitate similarity measurement. LVSM captures local aspects of items and clusters users into subsets, where a separate similarity function is learned for each subset. To achieve efficiency, we have conducted variational inference based on a variational auto-encoder, and optimized the model through the variational expectation maximization (EM)-algorithm.

Through a broad set of experiments, we have evaluated the efficacy of LVSM. LVSM outperforms state-of-the-art feature-based methods for recommending top-N new items. It provides robust recommendations independent of the quality of item features. It also generates good performance in extreme cases, e.g., with a large fraction of new items or with extremely sparse features. Besides, we have also found that (1) feature-based similarity models (FSMs) generally show good performance, especially when item features are of high quality as in the CiteULike article (CUL-a) and CiteULike tag (CUL-t) datasets; (2) the integration of local and global similarity functions measures item similarities more comprehensively than global similarity by itself and provides better top-N recommendation for new items than by only modeling the global similarity; and (3) the item representations learned by variational auto-encoder (VAE) denoise the original features and encode more information than learned via pure deep neural networks (DNNs).

Different from previous chapters, we study the problem of recommending top-N new items in this chapter by integrating item features and ratings. Next, we continue to utilize heterogeneous information for recommendations. We first combine user behavior and content information for scientific paper reranking (Chapter 6). We then provide a generic method for recommendation to overcome the high-dimensionality when integrating various information (Chapter 7).

6 Personalised Reranking of Paper Recommendations

In the previous chapters, we have studied how to utilize high-dimensional information for top-N recommendations (Chapter 2–4). We have also studied how to recommend top-N new items by a feature-based similarity model (Chapter 5). In this chapter, we combine user behavior and content information to rerank scientific papers. We propose a hybrid model that includes several content-based measures and a behavior-based method to address the challenge of recommending papers to cold-start users, which answers the following research question asked in Chapter 1:

RQ5 Can we address the challenge of recommending papers to cold-start users by effectively utilizing available heterogeneous information?

6.1 Introduction

Along with the digitization of academic resources and the increasing popularity of academic information platforms, online access to academic papers has become a widely used service. Various online academic service providers have given users access to papers through their search engines, such as Google Scholar [86], Aminer [219] and ScienceDirect [201], where users can enter queries to seek relevant papers in their database. In this scenario, users need to have an idea of what they are looking for, and the information needs can be formalized as queries. The search system takes a query as input, and returns a ranking of relevant papers for users to examine and interact with.

While such academic search engines can often fulfill user requests by catering to specific information needs represented as queries, there are cases when users' information needs are not explicitly specified. For instance, users may want to learn about new developments in their domain by looking at emerging papers that are relevant. In this case the user may not have an idea of what queries to enter on the search engine. This is a situation where paper recommender systems can step in and recommend relevant papers without the need for a user query.

Paper recommender systems have a role that is complementary to the search engine. The possible recommendation scenarios fall into three categories based on the recommendation timing:

This chapter was published as [139].

1. displaying paper recommendations before users start a new search session, based on their paper library or previously accessed papers [see, e.g., Google Scholar, 86];
2. during a search session, displaying related recommendations beside the content that the user is currently browsing [see, e.g., ScienceDirect, 201]; and
3. after a search session, sending emails of paper recommendations in the form of a newsletter [see, e.g., ScienceDirect, 201].

The first and third scenario fill the gap between user search sessions, while the second scenario is related to within-session recommendations.

In this study we focus on the third scenario. We look at the ScienceDirect paper recommender which sends a weekly email of paper recommendations to users. First, we provide a recommendation example from the system in Figure 6.1 to show how it works.[1] The recommender of ScienceDirect generates a ranked list of 5 paper recommendations based on the user's browsed papers. The email newsletter displays the title, venue (journal), authors, and publication date of each recommended paper. On clicking a recommendation, the user is linked to the paper on ScienceDirect. The system then logs on which recommendation(s) the user clicks. As a short summary, this system aims to recommend interesting papers to users based on their browsing history. A good recommendation list will place more relevant papers higher in the list.

Since the ScienceDirect paper recommender was released, an increasing number of users have signed up. It is especially challenging to make recommendations for these new users due to the lack of historical interactions with the recommender system. In this chapter, we address the challenge and try to come up with better recommendations for these new users. Specifically, we study the task of reranking the paper candidates generated by the current production system. Ranking is a very common module of the workflow in production recommendation systems, which usually include at least a candidate-generation phase and a ranking module [56, 60, 195]. The output of the system is generated by a multi-step process. We address this reranking task so that our model can easily be integrated into paper recommender systems (e.g., the ScienceDirect recommender). A direct application is to use our model to rerank the recommended papers generated by the ScienceDirect recommender system.

Over 14 million papers are indexed on ScienceDirect [202]. Picking the few papers that may appeal to the user is not a trivial task. Collaborative filtering techniques are often used in recommender systems to generate a candidate pool of papers based on user-paper interactions. Even though there was initially no data on user interaction with the recommender system, there was still a wealth of data on user interactions with papers on ScienceDirect. Apart from this behavioral aspect, paper metadata may also assist the recommendation task by providing similarity measures that are based on paper contents, e.g., to recommend semantically similar papers, or papers that are authored by the same or similar authors.

In this chapter, we propose a hybrid model that combines content and behavior to rerank the candidate paper recommendations generated by the ScienceDirect recommender. First, we propose several content-based measures that are derived from various paper aspects, such as word space similarity, and author similarity from an

[1] https://www.elsevier.com/connect/suffering-from-information-overload-personalized-recommendations-can-help


ScienceDirect

Hello Daniel, here are personalized recommendations based on your latest signed in ScienceDirect activity.

Constructing a Natural Language Inference dataset using generative neural networks
Original research article
Computer Speech & Language, Volume 46, November 2017, Pages 94-112
Janez Starc, Dunja Mladenić

Beyond interviewer effects in the standardized measurement of ego-centric networks
Original research article
Social Networks, Volume 50, July 2017, Pages 70-82
Andreas Herz, Sören Petermann

Restricted Boltzmann machines for vector representation of speech in speaker recognition
Open access - Original research article
Computer Speech & Language, Volume 47, January 2018, Pages 16-29
Omid Ghahabi, Javier Hernando

Key players in conservation diffusion: Using social network analysis to identify critical injection points
Original research article
Biological Conservation, Volume 210, Part A, June 2017, Pages 222-232
Emmanuel K. Mbaru, Michele L. Barnes

More structural holes, more risk? Network structure and risk perception among marijuana growers
Original research article
Social Networks, Volume 51, October 2017, Pages 127-134
Aili Malm, Martin Bouchard, Tom Decorte, ... Marije Wouters

Figure 6.1: An excerpt from a sample recommendation email sent to a ScienceDirect user based on his recent activity. The email contains 5 papers linked to ScienceDirect.

embedding space. Next, we use joint matrix factorization to learn a mapping from a user's browsed articles on the search engine to a user's clicks on the recommendations, to alleviate the sparsity of the recommendation click data. We use a pairwise learning model to rerank the candidate paper recommendation, which eventually leads to better results in offline evaluations based on real email click data.

The contribution of this chapter mainly lies in: (1) "task transfer" for the academic setting: data collected for one task (search) is used to help optimize performance on another (recommendation), and (2) how to combine content and user behavior to generate high-quality academic recommendations. The framework captures user interests on different paper aspects, as well as alleviating the sparsity problem in click data. The recommendation framework for ScienceDirect has implications for academic recommendation settings that share similar inputs. The framework relies on two kinds of input: paper properties and user interactions. The paper properties that we have utilized can be found on many popular academic search engines such as Google Scholar [86], Semantic Scholar [204] and CiteseerX [127]; the user interactions, i.e., how users interact with the search engine and the recommender respectively, are also available. For instance, in Google Scholar's search interface, there is a snippet showing recommended papers below the search bar for users to click on, even when users have not entered a search query; Semantic Scholar also has its proprietary email alert service that can send relevant paper recommendations.

The chapter is structured as follows. We describe the models that we propose in § 6.2, the experimental setup in § 6.3, and the results and analysis in § 6.4. We present related work in § 6.5 and conclude in § 6.6.

6.2 Models

In this section we introduce the models for the paper recommendation task. First, we introduce the production baseline, because it provides the candidates for our proposed reranking model. Then we introduce our *hybrid reranking model* (HRM) that considers both behavior and content, and reranks the candidates. Here, "hybrid" refers to using both content and behavior.

6.2.1 Production baseline

The production system takes the paper browsing history of a user as input, and produces a ranked list of 5 paper recommendations. While we are not able to elaborate on the exact details of the production system, we can describe the core part of the algorithm: the 5 candidates are generated and ranked by an algorithm that uses an item-item neighborhood-based collaborative filtering method [145, 197], based on usage similarity from ScienceDirect browsing logs. We refer to this paper-paper similarity as *browsing similarity* in the remainder of the chapter.

In this study, we apply the reranking model to the top five candidates from the production system, and compare the model's ranking to the production baseline. The top five candidates were chosen because for these recommendations, there is email click feedback that enables offline evaluation; if successful, the model could be applied to a longer list of candidates.

6.2.2 Proposed model

In this section we introduce the *hybrid reranking model* (HRM) by first providing an overview of the model architecture and then delving into the details. The model scores paper recommendation candidates generated by the production system, using both content and behavior components which will be explained shortly. The candidates are then reranked by the score.

Model architecture overview. An overview of the model is shown in Figure 6.2. A 2-layer feedforward neural network is used as the scoring function, where the input layer takes features from each candidate paper, and the output layer contains one node that yields the score.

We explain what the input feature representations are in Figure 6.2 from left to right: S_{recent} and $S_{history}$ are the similarity between recommendation candidates and users' browsed papers. They contain the average similarity scores of each paper aspect by comparing the candidate paper against the recent papers and historic papers, respectively; the attention features on different fields of papers and on recent/historical papers are derived from a user's browsed papers.

The browsing similarity features used by the production system are based on ScienceDirect browsing data: we use the mean and maximum similarity scores of each paper recommendation candidate compared against papers in the browsing history. The behavior features are the predicted click scores from the behavior model. Together these features determine the inputs for HRM.

Training is done by optimizing a pairwise hinge loss from the preferences of clicked

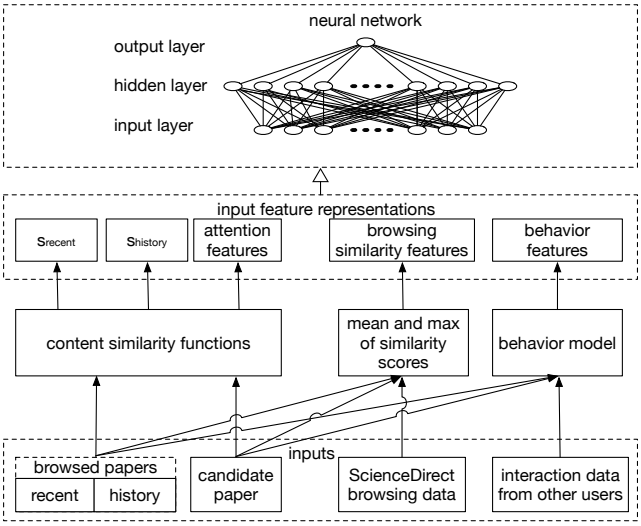


Figure 6.2: Architecture of the HRM that shows how a candidate paper recommendation for a user is scored.

papers $\mathcal{R}_u^+$ over the non-clicked papers $\mathcal{R}_u^-$ for each user u, shown below:

$$L(\mathcal{R}_u^+, \mathcal{R}_u^-) = \sum_{p_i \in \mathcal{R}_u^+} \sum_{p_j \in \mathcal{R}_u^-} \left[1 - f(x_{p_i}) + f(x_{p_j})\right]_+, \tag{6.1}$$

where $f(\cdot)$ denotes the scoring function (neural network), and x_{p_i} and x_{p_j} denote the feature representations for clicked paper p_i and non-clicked paper p_j , respectively.

We apply rectified linear unit (ReLU) activations on the hidden layer for efficient learning. We apply linear activations on the output layer, because this ensures an unbounded value for the pairwise loss function and also performs best in our experiments. We use the Adam optimizer [115] and mini batches during training.

Next, we introduce how we consider paper metadata to measure different types of similarity. Below, we provide formal representations of various paper aspects, of users, and then the similarity functions for them.

Before continuing, let us briefly introduce the main notation that we will be using in the remainder of the chapter; see Table 6.1.

Paper representations. Each paper p is represented as a collection of different aspects, grouped as follows: (1) *metadata from papers*: author A, venue V, freshness F, word space W, entity space E; (2) *metadata from user interactions*: impact I and popularity P. These aspects are available for all papers and users in our scenario and considered to be potentially useful for the recommender system. The reason for considering authors and venues is that users may be interested in papers from the same or similar authors, and those published in the same or similar venues. The word space and entity space measure content similarity and are thus also potentially useful. Besides, academic searchers may seek papers with high impact when they are learning about a domain, or popular papers (e.g., those with many downloads) that their community is discussing, or newly published papers that track the latest developments, hence the

Table 6.1: Notations used in the chapter.

	Notation	Description
Content-based mesures	p, p_i, p_j	Papers
	$A(p)$	Set of authors of paper p
	$V(p)$	Venue where paper p is published
	$F(p)$	Freshness score of paper p
	$I(p)$	Impact score of paper p
	$P(p)$	Popularity score of paper p
	$W(p)$	Word space representation of paper p
	$E(p)$	Entity space representation of paper p
	$SimX$	Similarity of field X for 7 choices of X (V, A, F, I, P, W, E)
	α_{field_i}	Attention feature for $field_i$ from 7 fields (V, A, F, I, P, W, E)
	α_{recent}	Attention score on the user's recent papers
	$\alpha_{history}$	Attention score on the user's historic papers
Behavior-based model	m	Number of users
	n	Number of papers
	k	Number of latent dimensions
	$B \in \mathbb{R}^{m \times n}$	Paper browsing history matrix
	$R \in \mathbb{R}^{m \times n}$	Paper click history matrix
	$S \in \mathbb{R}^{n \times n}$	Paper-paper browsing similarity matrix
	$D \in \mathbb{R}^{n \times n}$	A diagonal matrix whose entries are the row sums of S
	$M \in \mathbb{R}^{n \times n}$	Matrix to map browse to click
	$Q \in \mathbb{R}^{n \times k}$	Paper factor matrix
	$s_{ij} \in \mathbb{R}^{n \times n}$	Browsing similarity between paper p_i and p_j
	$\boldsymbol{q_i}$	Latent factor for paper p_i
	b_i	Bias of paper p_i
	r_{ui}	Predicted score of click of user u to paper p_i
	$\mathcal{B}_u^+$	Set of papers browsed by user u
	$\mathcal{R}_u^+$	Set of papers clicked by user u
	$\mathcal{R}_u^-$	Set of papers shown to but not clicked by user u
	$\mathcal{R}_u$	Set of papers in the candidate, i.e., $\mathcal{R}_u = \mathcal{R}_u^+ \cup \mathcal{R}_u^-$

inclusion of impact, popularity and freshness.

Formally, we can think of every paper p as a tuple $p = \langle A(p), V(p), F(p), W(p), E(p), I(p), P(p) \rangle$, where each aspect is defined as follows:

Authors:

$$A(p) = [a_1, a_2, \ldots, a_n], \tag{6.2}$$

where a_i is an author of the paper p.

Venue:

$$V(p) = v_i, \tag{6.3}$$

meaning that paper p is published in venue (journal) v_i.

Freshness score: We also model how "fresh" a paper p is, defined as:

$$F(p) = \frac{1}{e^{t_{current} - t_{publish(p)}}}, \tag{6.4}$$

where $t_{current}$ is the current time, and $t_{publish}(p)$ is the time of the paper p being published online. $F(p) \in (0, 1]$. The more recently a paper has been published, the higher the freshness score is.

Impact score: We use citations as a measure of impact for papers, which is defined as:

$$I(p) = \frac{\log(c(p) + 1)}{\log(c_{max} + 1)}, \tag{6.5}$$

where $c(p)$ is the citation count of paper p, and c_{max} is the maximum number of citations in the dataset.

Popularity score: The popularity of a paper p reflects how often users interact with the paper. We use the number of downloads to represent popularity:

$$P(p) = \frac{\log(d(p) + 1)}{\log(d_{max} + 1)}, \tag{6.6}$$

where $d(p)$ is the number of downloads of paper p, and d_{max} is the maximum number of downloads of a paper in the dataset.

Word space: To represent a paper p in a word space $W(p)$ we use tf-idf vectors, with values for words and bigrams in the article title, abstract and keywords. We remove English stop words, very common words and very rare words before calculating the tf-idf values. In the end, each paper is represented as a sparse vector of size of 2^{21}, with hashing to determine token indices in the vector.

Entity space: While word space measures such as tf-idf similarities can be used to directly compare the contents of papers, an entity space representation $E(p)$ is able to provide us with additional information that incorporates both structure and semantics through graph embeddings [25, 144].

We first build a knowledge graph by using important aspects of a paper including keyword, author and venue. The graph contains 4 node types, paper, author, keyword and venue nodes, and 3 relations (predicates) between a paper and an aspect as listed below: (1) *hasAuthor*: the paper has this author; (2) *hasKeyword*: the paper contains this keyword; and (3) *publishedInVenue*: the paper is published in this venue (journal).

Next, to compute the entity space we use the TransE model [25] to derive embeddings based on knowledge graphs. As input the model takes the triplets in the graph; these have the form (h, r, t), with a head entity h, a relation (predicate) r, and a tail entity t. The objective of the model is to learn embeddings so that $h + r$ lies in the proximate neighborhood of t if such a triplet (h, r, t) exists in the training set, and $h + r$ will be far away from t if the triplet is not valid. The model learns embeddings by minimizing a pairwise hinge loss:

$$\sum_{(h,r,t)\in T} \sum_{(h',r,t')\notin T} [1 + \|h + r - t\|_2 - \|h' + r - t'\|_2]_+, \tag{6.7}$$

where T denotes the training set of triples. Negative triplets (h', r, t') are sampled by replacing either the head or the tail entity with another random entity. After training, the cosine distance of the node embeddings reflects their proximity in the knowledge graph.

Due to the relatively high computational costs of working with knowledge graphs [25], we derive the embeddings on a subgraph instead of on the complete graph. We choose a reasonable size for the subgraph so that it is computationally feasible and also alleviates the sparsity problem in the node connections. The subgraph is comprised of the union of the browsed papers and recommended papers from 65,994 users, a superset that is about 15 times the size of the set of users that we will study in our experiments. In total we have 609,716 paper nodes, 1,650,470 author nodes, 3,961 venue nodes and 808,845 keyword nodes, plus 6,103,728 relation edges.

The graph is then used as input for the TransE model to derive embeddings of the nodes in the graph. In the end, we obtain embeddings for papers, authors and venues. These embeddings will be used later in content similarity measures.

User representations. The user representations are straightforward: each user u is represented as a collection of papers in their browsing history:

$$u = [P_{recent}, P_{history}] \tag{6.8}$$

$$P_{recent} = [p_1, p_2, \ldots, p_k] \tag{6.9}$$

$$P_{history} = [p_{k+1}, p_{k+2}, \ldots, p_n]. \tag{6.10}$$

We segment a user's browsed papers into two sets, the recent ones, P_{recent}, and the historical ones, $P_{history}$. We write p_i to refer to the i-th paper in each of the segmentations, in the order of occurrence in the user's timeline starting from the most recent one. In academic search, users' topic interests may shift over time [137]. We make this segmentation so that it may help us compare the user's recent interests against their historical interests, and see whether and to which extent there is a deviation.

In case of a large deviation, P_{recent} should provide more support to generate paper recommendations. Specifically, the clicked papers in the most recent session are put into P_{recent} if it contains at least clicks on 2 different papers, and the rest into $P_{history}$. Otherwise, we select the most recent θ papers from u into P_{recent} and put the rest into $P_{history}$. Papers in P_{recent} and $P_{history}$ are deduplicated.

Content similarities. Based on the user and paper representations, in this section we describe similarity functions to measure different types of content similarity. Specifically, the content component measures the similarity between candidate recommendations and users' browsed papers using information from the paper metadata. The output consists of similarity scores to feed into the reranking model.

Field-level similarities and attention features: First, we introduce similarity measures for individual fields, which are used to compare paper similarities in each field. When comparing two papers p_i and p_j, the similarity of each field is defined as follows.

For the word space and entity space, we use the cosine similarity of the vectors that represent each paper. The cosine similarity between two vectors v and v' is defined as:

$$\cos(v, v') = \frac{v \cdot v'}{\|v\| \cdot \|v'\|}, \tag{6.11}$$

where the similarity value $\cos(v, v')$ ranges between -1 and 1.

Then, the similarities for word and entity space are:

$$SimW(p_i, p_j) = \cos(W_{p_i}, W_{p_j}) \tag{6.12}$$
$$SimE(p_i, p_j) = \cos(E_{p_i}, E_{p_j}), \tag{6.13}$$

where W_{p_i} is the tf-idf vector and E_{p_i} is the paper entity vector for paper p_i obtained from the output of the TransE model [25].

Similarly, a venue entity vector $E_{v_{p_i}}$ for paper p_i and an author entity vector E_{a_m} for author a_m of p_i are obtained from the output of the TransE model [25]. We apply a "soft match" approach when comparing venue and author similarities. Compared to an "exact match" approach where the similarity ends up being either 1 (same) or 0 (different), the "soft match" approach outputs a continuous similarity score. For instance, "Accident Analysis & Prevention" and "Safety Science" being two different journals (with no overlapping terms in the journal title), they would have a similarity score of 0 in the "exact match" approach. However, in the embedding space they would have a similarity score of 0.48, representing a more precise estimate of the inherent similarity.

Then, venue and author based similarity measures, $SimV(\cdot,\cdot)$ and $SimA(\cdot,\cdot)$ are defined as follows:

$$SimV(p_i, p_j) = \cos(E_{v_{p_i}}, E_{v_{p_j}}) \tag{6.14}$$

$$SimA(p_i, p_j) = \begin{cases} \dfrac{\sum\limits_{a_m \in A_{p_i}} \max\limits_{a_n \in A_{p_j}} \cos(E_{a_m}, E_{a_n})}{|A_{p_i}|}, & \text{if } |A_{p_i}| \leq |A_{p_j}| \\ \dfrac{\sum\limits_{a_n \in A_{p_j}} \max\limits_{a_m \in A_{p_i}} \cos(E_{a_n}, E_{a_m})}{|A_{p_j}|}, & \text{otherwise,} \end{cases} \tag{6.15}$$

where v_{p_i} is the venue of paper p_i, $E_{v_{p_i}}$ is the corresponding paper entity vector; A_{p_i} is the set of authors of paper p_i, and E_{a_m} is the entity vector for author a_m. Note that in the author similarity function, we examine each author from the smaller author set and find the most similar one in the other set and then calculate the average of the similarities. This ensures that $SimA(p_i, p_j)$ is symmetrical.

For freshness, impact and popularity, these three measures are single value features. We use $L1$ distance with an adjusted weighting to obtain their similarities:

$$SimF(p_i, p_j) = (1 - \|F(p_i) - F(p_j)\|_1) \times \max(F(p_i), F(p_j)) \tag{6.16}$$
$$SimI(p_i, p_j) = (1 - \|I(p_i) - I(p_j)\|_1) \times \max(I(p_i), I(p_j)) \tag{6.17}$$
$$SimP(p_i, p_j) = (1 - \|P(p_i) - P(p_j)\|_1) \times \max(P(p_i), P(p_j)). \tag{6.18}$$

We define the weighting in order to capture the similarities only when two papers both have a high value in this field. In cases where both have a low value, the similarity value will be "down-weighted," representing a weaker level of evidence for similarity. For instance, given 2 paper pairs with low impact values (0.1, 0.2) and high impact values (0.8, 0.9), the similarity score would be 0.09 and 0.81 respectively. Although

the absolute difference of impact is the same for both pairs (0.1), the pair with relatively high values has a much larger similarity score.

Field level attention: Now that we know how to obtain the similarity scores $SimX$ for 7 choices of X (V, A, F, I, P, W, E), we would like to further know which specific fields the user may be focusing on while browsing papers. This is to tailor the recommendations for those fields, be it the semantic similarity, venues or authors. These "attention features" are implicit. However, we can derive the attention features through past user interactions. In particular, we assume that they can be inferred from P_{recent} (users' recently browsed papers). We hypothesize that for a set of papers, if the average pairwise similarities of certain aspects are higher than other fields, it is probably because users are paying attention to these aspects. For instance, high word space similarity indicates that users are sticking to a specific topic. Likewise, if the venue and freshness similarity scores are high, this could be that the user is mostly checking papers that are both recent and are from a specific journal. We use the averaged pairwise similarities calculated by each field as the field-level attention feature.

The attention feature for $field_i$ is the sum of its pairwise similarities divided by the number of paper pairs in P_{recent}:

$$\alpha_{field_i} = \frac{\sum_{p_i,p_j \in P_{recent}, i \neq j} Sim_{field_i}(p_i, p_j)}{C^2_{|P_{recent}|}}, \tag{6.19}$$

where $C^2_{|P_{recent}|}$ refers to the number of paper pairs.

Recent and history attention: The users' recent and historical paper interactions may both provide evidence to surface good recommendations. We make the distinction between recent and historical papers because users' interests may evolve over time. When the users' recent interests are significantly different from their historical interests, the recommender should be aware of this deviation. Therefore, we define attention features for this situation, where α_{recent} and $\alpha_{history}$ represent the two attention scores on the user's recent and historic papers.

α_{recent} and $\alpha_{history}$ are calculated using the browsed papers (P_{recent} and $P_{history}$). The more the user's recent interests deviate from the historic interests, the higher the value of α_{recent}, hence providing a bias feature to consider the more recent user activities. It is calculated as follows:

$$\alpha_{recent} = Distance(P_{recent}, P_{history}) \tag{6.20}$$

$$\alpha_{history} = 1 - \alpha_{recent}. \tag{6.21}$$

The distance $Distance(P_{recent}, P_{history})$ is calculated by averaging over the distance of each paper in P_{recent} to its closest match in $P_{history}$. The idea of finding each paper's closest match instead of averaging over all papers in $P_{history}$ is because the history may be diversified: a recent paper may be very similar to one paper in the history but different to the rest. In case there is at least one similar paper in $P_{history}$, we consider that the current paper being examined does not deviate far from the history.

Formally, the distance is defined as follows:

$$Distance(P_{recent}, P_{history}) = \frac{\sum_{p_i \in P_{recent}} \min_{p_j \in P_{history}} (1 - \cos(W_{p_i}, W_{p_j}))}{|P_{recent}|}, \quad (6.22)$$

where $1 - \cos(W_{p_i}, W_{p_j})$ is the cosine distance between two papers' tf-idf vectors.

So far, we have explained how we exploit the content aspects for recommendation that are based on the paper metadata. Next we introduce the behavior aspect where user-paper interactions are concerned.

Behavior. Paper metadata provides evidence for recommendations from the content perspective. User interactions, i.e., users' browsing behavior on the search engine and clicks on recommendation emails, also provide signals for generating good recommendations. In our scenario, the users have past browsing behavior but no clicks prior to their first interaction with the recommender system.

Nevertheless, the paper-paper browsing similarities are available to us (as used by the production system). They provide a measure of behavior-based similarity based on readership of all users on ScienceDirect.[2] Naturally, we can incorporate this external similarity information into our model.

We devise a behavioral model, that utilizes both browsing and click behavior in the interaction log. The motivations of our model are given below:

1. For new users, there are no prior email clicks for predicting their interactions with the recommender. To address the issue, we complement the absence of click ratings by using the browsing history. Obviously, browsing papers on the search engine and clicking a paper in the email are two different user interactions with papers. Thus a mapping function is required to transform browsed papers to email clicks. It is not possible to learn the mapping for every user as there may be no click at the time of recommendation, but it is possible to utilize the browsed papers and email clicks of other users (and this data quantity will grow over time). Essentially, we try to infer the clicks of new users from other users' mappings, using supervised learning.
2. As paper recommendations are shown in a relatively compact email, we assume that users have noticed all the papers. Therefore, a user's clicks on the 5 shown papers in the email entail implicit pairwise preferences. For instance, given 5 papers p_1, p_2, p_3, p_4 and p_5, if the user clicks paper p_2 and p_3 in the list of 5 papers, then it is reasonable to assume that they prefer paper p_2 and p_3 over paper p_1, p_4 and p_5.
3. The paper-paper similarity based on a user's browsing history is available. It is likely more accurate than the similarity from user clicks in emails, because it is based on the complete set of ScienceDirect users, which is several orders of magnitudes larger. Moreover, it captures transitive similarities from a global perspective. Therefore, it is important for our model to preserve this similarity.

Recall that our notation was introduced in Table 6.1; it is used in the following model descriptions. We propose to learn a mapping function from user browsed papers to user clicks on the email, denoted as:

$$R \sim BM, \quad (6.23)$$

[2]The involved users are larger than the users we study in our experiments by several orders of magnitude.

where $B, R \in \mathbb{R}^{m \times n}$ are the matrices for browses and clicks respectively and $M \in \mathbb{R}^{n \times n}$ is a mapping matrix. In practice, n is generally very large so that it could pose a great burden to learn M. Thus we propose to factorize M into the multiplication of a low-dimensional paper factor $Q \in \mathbb{R}^{n \times k}$, shown below:

$$M \sim QQ^T, \tag{6.24}$$

where T is the transpose operator of a matrix.

Based on the assumptions given in Eq. (6.23) and (6.24), we can predict the click of user u on paper p_i by the equation below:

$$\tilde{r}_{ui} = b_i + \boldsymbol{q}_i^T \sum_{t \in \mathcal{B}_u^+} \boldsymbol{q}_t, \tag{6.25}$$

where b_i is a scalar for the bias of paper p_i. $\mathcal{B}_u^+$ is the set of papers browsed by user u. Here, we ignore the user bias in Eq. (6.25) since it is unknown for new users. Note that we do not exclude p_i from $\mathcal{B}_u^+$, as suggested by the item-based collaborative filtering methods. This is because the set of candidate papers does not overlap with the set of browsed papers, i.e., $\mathcal{B}_u^+ \cap \mathcal{C}_u = \emptyset$. Each time we draw a pair of papers (p_i, p_j) for each user to learn Q and b_i, b_j, where $p_i \in \mathcal{R}_u^+$ and $p_j \in \mathcal{R}_u^-$, and optimize a pairwise loss function given by Bayesian personalized ranking [188]:

$$\mathcal{L}(u, p_i, p_j) = -\log \sigma\left(\tilde{r}_{ui} - \tilde{r}_{uj}\right), \tag{6.26}$$

where $\sigma(\cdot)$ stands for the sigmoid function. To preserve paper-paper similarities from the browsing history, we follow the assumption that the distance between $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ is small when s_{ij} is large. Without loss of generality, we adopt the Euclidean distance, e.g., $\|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2$. We can then define the following similarity regularization terms:

$$\begin{aligned} \frac{1}{2} \sum_{i,j}^{n} \|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2 s_{ij} &= \sum_{i=1}^{n} \boldsymbol{q}_i^T \boldsymbol{q}_i d_{ii} - \sum_{i,j}^{n} \boldsymbol{q}_i^T \boldsymbol{q}_j s_{ij} \\ &= \mathrm{Tr}\left(Q^T D Q\right) - \mathrm{Tr}\left(Q^T S Q\right) = \mathrm{Tr}\left(Q^T L Q\right), \end{aligned} \tag{6.27}$$

where $\mathrm{Tr}(\cdot)$ is the trace operator of a matrix, D is a diagonal matrix whose entries are the row sums of the browsing similarity matrix S (S is symmetric), i.e., $d_{ii} = \sum_{j=1}^{n} s_{ij}$, and $L = D - S$ is the Laplacian matrix of the graph [53]. Putting Eq. (6.26) and (6.27) together, the model is given as follows:

$$\min_{Q, \{b_i\}} \quad \sum_{u=1}^{m} \sum_{\substack{p_i \in \mathcal{R}_u^+, \\ p_j \in \mathcal{R}_u^-}} -\log \sigma\left(\tilde{r}_{ui} - \tilde{r}_{uj}\right) + \alpha \mathrm{Tr}\left(Q^T L Q\right) + \frac{\lambda}{2} \|Q\|_F^2. \tag{6.28}$$

The first term in the objective function captures the pairwise preferences of every user over the papers shown in the emails. The second term preserves the paper-paper similarities in the browsing history through graph regularization. Graph regularization is widely used to preserve similarities, e.g., social regularization [154] and locality regularization [198]. The third term regularizes Q so as to avoid overfitting. α and λ are hyper-parameters.

We optimize Eq. (6.28) via Stochastic Gradient Descent. The optimizing procedure is similar to [71], i.e., for each user u, we sample a positive item $i \in \mathcal{R}_u^+$ and a negative item $j \in \mathcal{R}_u^-$, and optimize Eq. (6.28) w.r.t. (u, i, j). Note that we train Eq. (6.28) first, and then train HRM (Eq. (6.1)) given the output scores generated by the behavior-based model. While we can devise an end-to-end model to train the behavior-based model jointly with HRM, the optimization procedure can be very complex and inefficient. This is because the training procedures of Eq. (6.28) and HRM are very different. The behavior-based model is trained via sampling to capture the implicit relationships between items, whereas HRM assumes the independency among inputs to perform mini-batch training.

6.3 Experiments

In this section we describe the experiments, including research questions, data preparation, and experimental setup.

6.3.1 Research questions

We aim to find out how to utilize both content and behavior to rerank paper recommendations. We are interested in whether HRM which utilizes content and behavior can beat the production baseline, and how useful the different input features are. Specifically, we answer the following research questions.

RQ5.1 Does HRM, which utilizes content and behavior, provide improvements in reranking over the production baseline?

RQ5.2 What is the utility of the content features and behavior features in reranking, respectively?

RQ5.3 Within the content features, for paper similarity based on various paper aspects, which paper aspects contribute to good reranking performance and which do not?

6.3.2 Dataset

We use a dataset provided by ScienceDirect,[3] a popular academic search engine that offers access to millions of academic papers. Users can gain access either by a subscription service, or by individual purchases of papers. The dataset contains anonymized user activity logs from signed in users. We look at newly signed up users and their interactions on the first paper recommendation email. The paper recommendations emails were sent between December 12, 2017 and January 21, 2018. For each user, browsed papers on ScienceDirect prior to receiving the email were also obtained. A browsing action is characterized by any form of a click on a paper, such as a click on the search engine result page, or a click on related papers shown on the detailed paper content page. For email recommendation data, each email contains 5 candidate paper recommendations where users' responses to each one of them are logged (clicked or not clicked). To obtain paper metadata, we use the paper metadata from Scopus,[4] which can be obtained by querying paper IDs from papers in the ScienceDirect database.

[3] https://www.sciencedirect.com/

[4] https://www.scopus.com

Since we want to study how *content* contributes to better reranking, we need users that have at least a few papers in their browsing history in order to utilize the content information. The content data of the browsed papers should be clean and complete. Also, we need users who have at least one click on the recommendation email so that we can perform offline evaluations for reranking and calculate the metrics. Correspondingly, we apply the following filtering steps prior to obtaining the data: (1) we filter out cold start users with fewer than 5 browsed papers prior to the recommendation; (2) we remove users whose browsed papers have incomplete or corrupt fields of data; and (3) we remove the recommendation emails without any clicks. In total we have obtained 4,392 recommendation sessions for our experiments. Each session contains one recommendation email with a field that indicates whether each paper has a click, and also the user's browsing history prior to the recommendation's timestamp.

Also readily available are the item-item collaborative filtering scores based on readership of papers from ScienceDirect users. The scores of paper pairs are symmetrical so that $s_{ij} = s_{ji}$.

6.3.3 Experimental setup

The experiments on our dataset are conducted through 5-fold cross validation. For each run, 4 folds are used for training and 1 fold is used for testing. There are one or more clicks on the candidate paper recommendations for each email. We code relevance as a binary label, which is 1 for clicked papers and 0 for the rest. We compute the mean average precision (MAP) and Precision at k ($k = 1, 2, 3$, denoted as Pre@k for short) as the evaluation metrics.

Significance tests are applied when comparing the results of different models. Specifically, we apply the two-tailed student t test to MAP and Wilcoxon signed rank test to Pre@k, according to assumptions underlying the significance tests.

We select the optimal hyper-parameters for HRM by iterating over possible parameter combinations. For the content component of HRM, we have $\theta = 3$ chosen from $\{1, 2, 3, 4, 5\}$; for the behavior component of HRM, we have $\alpha = \lambda = 0.01$ and $k = 100$; for the scoring function of HRM, the hidden layer contains 32 nodes (more nodes may lead to overfitting and worse performances in the experiments), and the learning rate is set as 0.001.

What are appropriate baselines to consider? The first and obvious baseline is the production system that we seek to improve over; this baseline mainly uses item-item similarity based user browsing data on ScienceDirect. In addition, two families of approaches appear to be natural candidates: *learning to rerank methods* and *collaborative filtering methods*.

As to learning to rerank models, to the best of our knowledge, approaches to learning to rerank a production system published in the literature focus on learning from interaction data (see § 6.5.5). We, however, focus on similarity-based models. Thus, we consider an (offline) pointwise learning to rerank model based on logistic regression with Adagrad optimization [67], which has achieved state-of-the art performance [211]. We also use the state-of-the-art pairwise model RankSVM [104] and listwise model LambdaMART [27]. The hyperparameter c_{rank} of RankSVM is selected from $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$. We use the default parameters of LambdaMART in the Ranklib [58] implementation and tune the trees and

leaves parameters. We also consider an (offline) linear pairwise learning model that is trained using pairwise hinge loss, which differs from HRM by using a linear scoring function instead of the neural structure. These baselines use the same inputs as HRM.

As to collaborative filtering methods as possible baselines against which to compare the approaches in this chapter, we compare with libFM [186] and SVDFeature [34]. libFM and SVDFeature construct the feature matrix from user ratings; both can provide effective recommendations even if the ratings are sparse [95, 176].[5]

We describe how to construct the feature matrix for libFM and SVDFeature. 1. The first m values represent the users; 2. the following n values represent the candidate paper recommendations for the user; 3. the next n values represent the browsed papers on the search engine; 4. the final value indicates whether the user clicked the paper from the candidate recommendations. An example is given as follows. Suppose we have 3 users and 10 papers. Suppose for user 1, papers $1, 3, 4, 6, 9$ are presented to them as candidate recommendations, among which they clicked paper $1, 3$. Besides, the user also browsed papers $2, 5$ on the search engine. We use red, blue, green and black color to represent the users, recommended papers, browsed papers and clicks respectively in the feature matrix constructed for user 1, as shown below:

1	0	0	1	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0.7	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0.7	0	0	0.7	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0.7	0	0	0.7	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0.7	0	0	0.7	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0.7	0	0	0.7	0	0	0	0	0	0
	users							recommended papers								browsed papers						clicks

For SVDFeature, we tune the regularization parameters for user factor λ_1 and item factor λ_2. We tune parameter for user bias b_1 and item bias b_2. We also tune the dimension of latent factors k. The parameters $\lambda_1, \lambda_2, b_1, b_2$ are explored from $\{0.01, 0.1, 1, 10\}$; k is explored from $\{5, 10, 15, 20\}$.

We use stochastic gradient descent (SGD) to optimize libFM. We tune the regularization parameters for bias α, 1-way interaction β, 2-way interaction λ and the dimension of latent factors k. Similarly, α, β, λ are tuned from $\{0.01, 0.1, 1, 10\}$ and k from $\{5, 10, 15, 20\}$.

Note that when answering the second and third research questions (examining the feature utility), certain components' feature size is very small. For instance, the behavior component consists of a feature size of two: one from our proposed behavioral model, one from the browsing similarity. Such a small feature vector is not suitable as input for the neural structure in HRM. Therefore, we use the pairwise linear model in this case.

6.4 Results and Analyses

In this section we present the experimental results, including the results of different models, and break-down analyses on different components of the model.

[5]The rating density is less than 0.02% even if we consider both browses and clicks as ratings on papers, which is significantly less than common recommendation datasets (Movielens 100K: 6.30%, Movielens 1M: 4.47%, FilmTrust: 1.14%).

Table 6.2: Results of reranking candidate paper recommendations across models. **W**in/**T**ie/**L**oss are the number of users for which a model performs better than, the same as, or worse than the production baseline.

Model	MAP	Pre@1	Pre@2	Pre@3	W/T/L
Production baseline	0.588	0.392	0.350	0.323	-/-/-
libFM	0.525	0.302	0.295	0.293	1854/ 999/1539
SVDFeature	0.525	0.305	0.296	0.292	1837/1004/1551
Linear pointwise l2r	0.534	0.330	0.296	0.280	1595/ 753/2044
Linear pairwise l2r	0.620	0.432	0.378	0.343	1822/1254/1316
RankSVM	0.615	0.423	0.376	0.341	1924/1220/1248
LambdaMART	0.627	0.443	0.383	0.345	2006/1102/1284
HRM	0.663	0.502	0.453	0.421	2005/1171/1216

6.4.1 RQ5.1: Overall comparison

To address our first research question (Does HRM, which utilizes content and behavior, provide improvements in reranking over the production baseline?), we compare HRM against the production baseline, as well as the other baselines, see Table 6.2.
Compared with all baselines, significant improvements are made in the hybrid reranking model HRM that combines content and behavior ($p < 0.01$). Compared with the production baseline, HRM performs better or the same for 72.3% of the users. There is a relative 13% increase in MAP and a relative 28% increase in Prec@1 for HRM, meaning that users are more likely to click the top candidates in the reranked list. This answers the first research question.

Besides, when given the same input features, HRM also performs better than all four reranking baselines, although leading LambdaMART only by a small margin. LambdaMART is a strong baseline and it has more users that perform better than the production baseline, compared with HRM. Interestingly, the pointwise learning to rerank method is beaten by the production baseline on all metrics. This shows that learning absolute user preferences of papers based on clicks is not optimal in our scenario. Models based on pairwise and listiwise learning (HRM, LambdaMART, RankSVM and the linear pairwise model) have produced better results by learning relative user preferences.

The behavioral baselines, i.e., libFM and SVDFeature demonstrate worse performance than HRM. A possible explanation is that libFM and SVDFeature cannot utilize paper browsing similarity which contains useful information to recover user behavior patterns, and they also do not capture content similarities.

We have also attempted to replace our behavior model in HRM by libFM and SVDFeature to see how they work with content similarities, which yields worse results than the original HRM. The scores are neglected for brevity.

6.4.2 RQ5.2: Utility of content and behavior features

To answer the second research question (What is the utility of the content features and behavior features in reranking, respectively?), we analyze the utility of the input features of individual components in HRM, shown in Figure 6.2. Specifically, we look at the reranking performance using the following input features separately.

– Proposed behavior feature.
– All behavior features: browsing similarity features and proposed behavior feature.
– Only recent content similarity: S_{recent}.
– Only historical content similarity: $S_{history}$.
– All content features without attention features: $S_{recent}, S_{history}$.
– All content features: $S_{recent}, S_{history}$, attention features.

The behavior features have small sizes (a single feature from our behavioral model and the production system, respectively). Therefore we opt for the linear pairwise model, because the small input feature vector is not suitable for the neural structure in HRM. For other features that have larger sizes we use the neural structure of HRM for reranking; see Table 6.3 for the results.

Table 6.3: The performance of reranking candidate paper recommendations using different input features of HRM shown in Figure 6.2.

Model	MAP	Pre@1	Pre@2	Pre@3
Proposed behavior feature	0.540	0.332	0.302	0.288
All behavior	0.602	0.411	0.358	0.327
Only recent content	0.590	0.384	0.354	0.333
Only historical content	0.582	0.374	0.343	0.327
all content without attention	0.598	0.398	0.359	0.337
All content	0.601	0.402	0.365	0.338

Using behavior and content separately for reranking, the results (MAP score of 0.602 and 0.601, respectively) already outperform the production baseline (0.588) that mainly uses item-item collaborative filtering. The proposed behavior feature provides a boost for the behavior component in addition to using the browsing similarity features from the production system ($p < 0.01$). On the other hand, the content component has a performance quite close to the behavior component. The attention features lead to a slight improvement over the model without them. We also find that using the recently browsed papers is better for reranking paper candidates than to using historically browsed papers, and even better is to use both recently and historically browsed papers. This answers the second research question.

6.4.3 RQ5.3: Utility of paper aspects

To answer the third research question (Within the content features, for paper similarity based on various paper aspects, which paper aspects contribute to good reranking performance and which do not?), we continue to delve into the content similarity in HRM, which contains similarity measures for different aspects of papers. We are interested to see the reranking performance of features based on a single paper aspect. For each paper aspect, we take the recent/historic similarity and the recent/historical attention scores as the input features for reranking. Similar to § 6.4.2, we use the pairwise linear model due to the small input feature size. The results are shown in Table 6.4.

The reranking performance of the paper candidates differs among the paper aspects. In general, the similarity measures based on semantics or entities perform better than those that do not. The two entity space measures: the author and paper entity sim-

Table 6.4: Reranking candidate paper recommendations by restricting the pairwise linear learning rerank model to using only one paper aspect.

Field	MAP	Pre@1	Pre@2	Prec@3
Freshness	0.426	0.153	0.248	0.242
Popularity	0.453	0.154	0.276	0.284
Venue	0.468	0.203	0.272	0.276
Impact	0.489	0.257	0.283	0.267
Word	0.526	0.312	0.291	0.284
Author	0.549	0.327	0.320	0.311
Paper entity	0.550	0.330	0.319	0.311

ilarities perform better than other measures, also beating the word-space similarity. Comparing three entity based measures, the author similarity performs similarly to the paper entity similarity, this is due to the high correlation between them (Pearson correlation coefficient being 0.88); the author similarity performs much better than the venue similarity (0.549 vs 0.468 for MAP scores). This may suggest that users pay attention to the authorship of the paper more than the venue. Using freshness, popularity, or impact similarity alone does not generate good performance, understandably, as these measures do not consider semantic relevance or entity relationships. Combining all paper aspects produces the best performance. The third research question is hence answered by the above comparisons of paper aspects' utility in reranking.

6.5 Related work

In this section we discuss the related work to our study. The related work spans several topics: academic search, paper recommendation, citation recommendation, top-N recommendation, and learning to rerank the output of a production system. We introduce them below and explain how they are related to our work.

6.5.1 Academic search

Our work is relevant to academic search because we are examining the recommendation service attached to an academic search engine. Academic search engines [10, 86, 127, 201, 219] have given users convenient access to academic resources such as papers, journals, and authors. Mitra et al. [163] found that different academic search engines have their own coverage of literature and ranking strategy, and the overlap among search results is low. Compared to general web search, there is far less research on user behavior in academic search, possibly due to a lack of public datasets. Research on academic search has examined user behavior through surveys [171, 181, 182] and aggregated usage statistics such as query frequencies [112]. Khabsa et al. [113] studied user queries on Microsoft Academic Search and proposed a query classifier.

Recently, more studies have been conducted on user behavior within and across search sessions, based on a large-scale user transaction log. Li et al. [138] have studied the null query phenomenon in academic search and proposed a query suggestion method as a remedy. Li et al. [137] have revealed correlations of query reformulation and topic shift in academic search. Li et al. [135] have studied characteristics of user queries in academic search following major scientific events. Li et al. [136] have also

studied download behavior in academic search and proposed a download prediction model. There has also been research aimed to improve the search experience. Tang et al. [220] combined topic modeling with random walks to improve academic search retrieval performance. Khazaei et al. [114] proposed a visual search interface via citation links to help users better navigate through search results. Xiong et al. [245] proposed improving paper rankings in academic search using entity embeddings.

Our work in this chapter differs from previous work in academic search in that we do not directly deal with search. We utilize the browsing history on the academic search engine to make improvements to a paper recommender.

6.5.2 Academic paper recommendation

Our recommendation task falls in the broad category of academic paper recommendations. Generally, based on the system inputs, paper recommendation tasks can be classified into the following scenarios: the system generates a list of paper recommendations given a single paper as input [18, 103, 166]; the system generates a list of paper recommendations given a set of papers as input (without ordering) [121, 205, 224]; the system generates recommendations given a time-ordered set of papers as input [98, 241]. The first and second scenarios include cases where a user is browsing a paper, or a list of relevant papers is available (e.g., through a set of papers selected by the user). The system assumes the input to be representative of a user's interests, then provides related papers as recommendations. These are the most common scenarios that are being studied. The third scenario is rarely studied because: a) it is relatively difficult to acquire user data that spans a long period, for instance, users' paper browsing history; b) it is more difficult to model user interests based on a sequence of inputs, compared to static inputs in the first two scenarios.

Common methods involved in making recommendations can be classified as: collaborative filtering (CF), content-based filtering (CBF) and hybrid models that combine the two. CBF involves using various parts of the paper contents, such as titles, abstracts, and keywords, to suggest related papers based on their similarity with input paper(s) [74, 103, 217]. While they are able to expose related papers that are similar by content, CBF models do not take into account user-paper interactions. CF models, on the other hand, utilize the user-paper interactions to generate recommendations, and can result in strong performance [30, 98, 179]. However, a common drawback of CF models is the cold start problem, which is severe in our academic recommendations when using real user-paper interaction data. Finally, there are hybrid models that combine CBF and CF models for paper recommendations [70, 224, 229]. The hybridization process is usually rule-based instead of learned: either the system first runs CBF models and then uses its output as input to run CF models to generate recommendations (cascade hybrid); or it simply mixes results that are separately generated from CBF and CF models (mixed hybrid).

Our work in this chapter differs from previous work on academic paper recommendation in that we study a rarely examined, but real scenario: generating paper recommendations given an ordered sequence as input. Specifically, we make recommendations for new users that sign up for the recommendations based on their browse history on the search engine. Compared to [241], which uses a simulated and artificial recommendation setting, our scenario concerns real user interactions with a recom-

mender system. We have proposed a hybrid model that combines content similarities, that draws distinction between multiple aspects of paper contents, and behavior-based similarities. We have applied pointwise and pairwise learning approach to train the model, unlike the rule based approaches to generate paper recommendation that do not apply learning techniques [70, 224].

6.5.3 Citation recommendation

Citation recommendation is sometimes mixed with paper recommendation. Hence, we draw the distinction between our paper recommendation task and citation recommendation. We consider citation recommendation to be the task of recommending papers to an author who is writing a manuscript. A citation recommender may take a complete or incomplete manuscript as input, identify places where citations are needed, and recommend relevant citations [90, 215]. It may also take a piece of "context" as input, which is represented as a few sentences, and generate relevant citation suggestions [68, 101]. It is obvious that the citation recommendation task is mainly focused on similarity. Even when collaborative filtering is applied, it is using the citation relation matrix as a paper similarity measure [148], instead of using the user-paper rating matrix. The evaluation setup is also confined to predicting the cited papers of an input paper or paragraph.

Our work in this chapter differs from previous work on citation recommendations in terms of the methods we propose, the recommendation goal, and the evaluation setup.

6.5.4 Top-N recommendation

In the context of more general recommendation problems, our scenario is related to top-N recommendation [63]. Top-N recommender systems provide users with a ranked list of items based on predicted scores of individual items, where the relative ranking matters more than the absolute item scores. This is similar to our problem as we aim to produce a ranking of papers according to the predicted scores. However, the candidate set from which we make recommendations is different: we pick the papers from a recommendation email, while a typical top-N recommender selects from all items that have not been rated by users.

Top-N recommenders have been intensively studied [192]. In general, there are approaches that use latent space models [57] and approaches that rely on neighborhood-based models (whether user-based or item-based) [63]. While latent factor models can also generate top-N recommendation, they are originally designed for rating prediction tasks. Therefore, they are sub-optimal for top-N recommendation. Neighborhood-based methods identify similar users or items, and have been shown to be more suitable for the top-N recommendation problem [4, 63, 107, 169]. Item-based methods have been shown to outperform user-based methods for the top-N recommendation task [51]. Similarity models have recently been proposed to improve item-based neighborhood models. They learn a coefficient matrix that is analogous to the item-item similarities [47, 107, 110, 169] directly from the data. A novel similarity model, sparse linear method (SLIM), has been proposed by [169]. Several authors have proposed improvements to SLIM. Low-rankness has been investigated to capture transitive relations [47, 107, 110]. Kabbur et al. [107] proposed a factored item similarity

model (FISM), which factorizes the coefficient matrix into two low-dimensional factor matrices. Cheng et al. [47] proposed the low-rank sparse linear method (lorSLIM), which introduces a rank regularization to SLIM. Kang et al. [110] made improvements over lorSLIM by providing a better proxy to approximate the rank of the coefficient matrix. Instead of estimating a single model for all users, Christakopoulou et al. [51] clustered users and estimated several local models. Zhao et al. [269] minimized a combined heterogeneous loss function, which is a combination of pair-wise ranking loss and point-wise recovery loss. Wu et al. [239] generalized FISM from linear to non-linear by incorporating a denoising auto-encoder.

Our work in this chapter differs from previous work on top-N recommendation in important ways. First, directly applying top-N recommendation models to our task will lead to two problems: new users have no clicks on the recommendation emails, a situation that cannot be handled by existing top-N recommenders. Also, we have two types of interaction between users and items: user browses and user clicks. Existing top-N recommenders focus only on homogeneous interactions.

6.5.5 Reranking the output of a production system

Like us, Lefortier et al. [126], Moon et al. [165] and Zoghi et al. [278] use a commercial search engine as their main baseline that they learn to improve. Lefortier et al. [126] and Moon et al. [165]'s methods directly use click-through rates, with a focus on documents that appear in the first position; both also focus on recency ranking and queries with shifting intent. Zoghi et al. [278] learn from a pairwise signal – out of order clicks in the top 5 produced by the production ranker.

Our work in this chapter differs from previous work on reranking the output of a production search engine or recommender system in that we do not restrict ourselves to recency ranking. Moreover, we do not work in an online setting and we do include content-based signals, not just behavior-based ones.

6.6 Summary

In this chapter, we have answered **RQ5** by proposing a hybrid approach that combines a content component and a behavior component to rerank papers. The content component measures similarities of various paper aspects between users' browsed articles and candidate recommendations, and also considers the user's attention on paper aspects and on recent/historical browsing. The behavior component learns a mapping from browsed articles to user clicks in the recommendations. The model combines content and behavior through a pairwise learning approach that is based on user interaction data.

We have found that our hybrid reranking model HRM significantly improves over the production baseline. We have dug into the components of our model to see what works and what does not. In the content component, the graph embeddings work the best, especially the author similarity based on soft matching; users' recently browsed articles can lead to better recommendations compared to historical browsing; on the other hand, popularity and impact similarities are not sufficient to bring up good recommendations alone. In the behavior component, our learned scores combined with browsing similarity scores have led to better performance than the production baseline.

The best performance is achieved when combining content and behavior through learning. Our hybrid reranking model HRM can be seen as a module that can be plugged into a recommendation system. Besides, we also have generalizable insights for other paper recommenders. For instance, we have revealed how each paper aspect contributes to the reranking performance.

Different from previous chapters that studied general methodologies, we study recommendation with heterogeneous information on a real application. Next, we study a generic method for top-N recommendation, which can overcome come the issue brought by the high-dimensionality when combining multiple types of information (Chapter 7).

7

Personalized Interaction Selection for Factorization Machines

In the previous chapters, we have studied how to utilize high-dimensional information for top-N recommendation (Chapter 2–4). Later, we have researched on integrating heterogeneous information for recommending top-N new items (Chapter 5). We have also investigated scientific paper reranking by combining content information and user behavior (Chapter 6). In this chapter, we study personalized feature interaction selection for factorization machines. The proposed method provides a generic way to utilize high-dimensional and heterogeneous information for top-N recommendations, which answers the following research question proposed in Chapter 1:

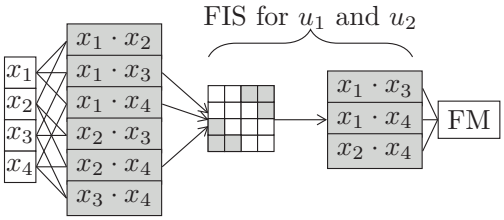
RQ6 How should we integrate high-dimensional and heterogeneous information for top-N recommendation?

7.1 Introduction

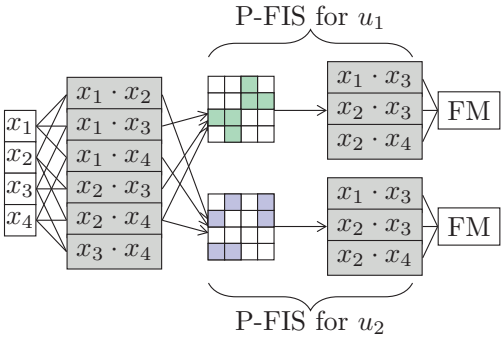
Factorization machines (FMs) [185, 186] are a generic supervised learning approach that combines the advantage of support vector machines (SVMs) [214] with factorization models [119]. factorization machines (FMs) account for interactions between features with factorized parameters [24, 206]. Feature interactions are crafted as combinations of individual features [46]. For example, in the movie recommendation scenario, the features for the movie "Spider-Man" can be "*comics*", "*marvel*" and "*avengers*". Accordingly, feature interactions can be combinations such as, e.g., "(*comics*, *marvel*)", "(*comics*, *avengers*)", etc.

FMs are widely applied in predictive analytics, ranging from recommendation [187], computational advertising [106], to search ranking [152] and toxicogenomics prediction [250]. FMs have been well studied for recommendations, due to their effective use of historical interactions between users and items [185]. FMs can incorporate additional information associated with users or items, including content descriptions [258], context information [189], social networks [31, 40], sequential dependencies [129]. While the wide availability of auxiliary data provides rich sources that may help reveal user preferences, they also increase the dimensionality of the feature space [43]. The problem of high-dimensionality is particularly severe for FMs, because FMs consider

This chapter was published as [37].



(a) Feature interaction selection.



(b) Personalized feature interaction selection.

Figure 7.1: Comparison between feature interaction selection and personalized feature interaction selection. $x_1, \ldots, x_4$ are features and $x_1 \cdot x_2, \ldots, x_3 \cdot x_4$ are feature interactions. The 4×4 matrices indicate masks for the selection of feature interactions. $x_i \cdot x_j$ will pass through the grid in the i-th row and j-th column or the j-th row and i-th column. The white grids of the matrices filter out feature interactions. FIS selects identical feature interactions for u_1 and u_2 while P-FIS selects feature interactions for u_1 and u_2 separately.

feature interactions. Hence, the complexity of FM models grows exponentially with the order of feature interactions. But not all feature interactions are helpful [45]; incorporating unnecessary feature interactions may bring in noise, which adversely impacts the recommendation accuracy and increases the difficulty of interpreting outcomes [247].

Feature interaction selection (FIS) has been proposed to select useful feature interactions and filter out useless feature interactions. Existing feature interaction selection (FIS) methods can be divided into two classes: *wrapper methods* [156] and *embedded methods* [45, 247]. Wrapper methods evaluate subsets of feature interactions by training a model with each subset and scoring on a held-out set. Although wrapper methods are more flexible as they do not depend on the recommendation model to use, they suffer from poor scalability. Embedded methods perform interaction selection during model training, which is more efficient and effective. Sparse factorization machines (SFMs) [176, 247, 271] are an example of embedded methods; they utilize sparsity regularization [221, 233] to achieve FIS. Existing FIS-based methods, including SFMs, overwhelmingly select a common set of interactions for all users non-personally, on the assumption that the same feature interactions are equally powerful to predict user's behavior. This assumption may not be valid, as it overlooks the individuality and personality of user's behavior, especially for recommendation tasks.

We introduce and study *personalized feature interaction selection* (P-FIS). As shown in Figure 7.1, unlike FIS, personalized feature interaction selection (P-FIS) aims to achieve adaptive FIS for each individual user. P-FIS is a more challenging task than non-personalized FIS since we have a limited number of instances associated with each user to select user-specific feature interactions. Although we can train a particular sparse factorization machine for each user, this is problematic for at least two reasons. First, it is both time and storage inefficient since we would need to maintain a model for each user. Second, it is less effective because estimating a model separately for each user fails to take advantage of collaborative filtering. To this end, we propose a *Bayesian personalized feature interaction selection* (BP-FIS) mechanism for FMs. First, instead of learning expensive and less effective personalized feature embeddings for each user, we estimate a single set of feature embeddings shared by all users to preserve the advantage of collaborative filtering. We achieve P-FIS by introducing personalized interaction selection variables and employ Bayesian variable selection (BVS) to estimate the selection variables, which allows us to avoid expensive cross-validation required by sparsity regularizations [221]. The widely used sparsity priors [11] for BVS are not ideal since they assign zero probability mass to events associated with weights having zero value [222]. Instead, we propose a hereditary spike-and-slab prior (HSSP), a variant of the commonly used spike-and-slab prior in BVS [9, 75, 162]. We formulate the BP-FIS as a probabilistic hierarchical generation procedure and derive an evidence lower bound (ELBO). Inspired by variational auto-encoders (VAEs) [116, 141], we use a stochastic gradient variational Bayes (SGVB) estimator to approximate posteriors of the latent variables and propose an efficient algorithm to optimize the model. BP-FIS can be seamlessly integrated into both traditional FMs (linear) and neural FMs (nonlinear).

We summarize the contributions of this chapter as follows:

1. To the best of our knowledge, we are the first to study P-FIS for FMs to improve recommendation performance.
2. We propose hereditary spike-and-slab priors to assign non-zero probabilities to zero values while preserving hereditary relations.
3. We formulate the P-FIS task as a probabilistic hierarchical generation procedure and conduct variational inference to derive the ELBO for optimization.
4. We implement two FM variants under our proposed Bayesian personalized feature interaction selection mechanism and verify their effectiveness through extensive experiments.

7.2 Preliminaries

7.2.1 Factorization machines

In the recommendation scenario, FMs try to predict the rating of an item based on its feature vector $\boldsymbol{x} \in \mathbb{R}^d$. A general formulation is shown in Eq. (7.1):

$$\hat{r}(\boldsymbol{x}) = b_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} w_{ij} x_i x_j, \tag{7.1}$$

Table 7.1: Summary of symbols and notation.

	Notation	Description
Sets and numbers	$\mathcal{U}$	Set of users
	$\mathcal{F}$	Set of features
	$\mathcal{X}$	Set of feature vectors
	$\mathcal{R}$	Set of ratings
	m	Number of users, i.e., $m = \lvert\mathcal{U}\rvert$
	d	Number of features, i.e., $d = \lvert\mathcal{F}\rvert$
	n	Number of ratings, i.e., $n = \lvert\mathcal{X}\rvert = \lvert\mathcal{R}\rvert$
	k	Dimension of feature embedding
Bayesian variables	$\boldsymbol{x} \in \mathbb{R}^d$	Feature vector
	$r(\boldsymbol{x}) \in \mathbb{R}$	Rating associated with feature $\boldsymbol{x}$
	w_{ui}	Personalized first-order feature interaction weight of user u for feature i
	w_{uij}	Personalized second-order feature interaction weight of user u between feature i and j
	W	Set of interaction weights, i.e., $W = \{w_{ui}\} \cup \{w_{uij}\}$
	$s_{ui}, \tilde{w}_i$	Variables to reformulate w_{ui}, i.e., $w_{ui} = s_{ui}\tilde{w}_i$
	$s_{uij}, \tilde{w}_{ij}$	Variables to reformulate w_{uij}, i.e., $w_{uij} = s_{uij}\tilde{w}_{ij}$
	$S, \tilde{W}$	Set of variables for reformulation, i.e., $S = \{s_{ui}\} \cup \{s_{uij}\}$ and $\tilde{W} = \{\tilde{w}_i\} \cup \{\tilde{w}_{ij}\}$
Parameters	$\boldsymbol{v}_i \in \mathbb{R}^k$	Embedding for feature i
	$V \in \mathbb{R}^{d \times k}$	Embeddings of all features
	b_u	Parameter for user bias of user u
	π	Variational parameters for S
	ρ	Variational parameters for $\tilde{W}$

where $\hat{r}(\boldsymbol{x})$ is the predicted rating for $\boldsymbol{x}$; $x_i \in \boldsymbol{x}$ is the i-th feature of $\boldsymbol{x}$; b_0, w_i, w_{ij} are the parameters, where b_0 models the global bias, w_i models the first-order interaction, i.e., the interaction between the feature i and the target, and w_{ij} models the second-order interaction, i.e., the interaction between feature i and j. Instead of learning a fixed interaction parameter w_{ij}, FMs factorize it as $w_{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$, where $\boldsymbol{v}_i \in \mathbb{R}^k$ is the embedding of feature i and k is the dimension of the latent space.

7.2.2 Bayesian variable selection

Variable selection is an important problem in statistical analysis, which selects a subset of variables that should be taken into consideration [221]. Bayesian variable selection (BVS) attempts to estimate the marginal posterior of a variable as the probability that the variable should be selected. Depending on the definition of priors, various BVS methods have been proposed [62]; spike-and-slab priors (SSPs) [237] have been widely studied.

A variable w following spike-and-slab prior (SSP) is sampled from a linear combination of two distributions:

$$w \sim \pi \mathcal{N}(\mu, \sigma^2) + (1 - \pi)\delta_0,$$

where $\mathcal{N}(\mu, \sigma^2)$ is the slab prior, which is modeled using a Gaussian distribution with mean μ and variance σ^2; δ_0 is the spike prior, which is modeled using a Dirac delta mass function centered at zero. The Dirac delta function is a generalized distribution that is used to model the density of an idealized point mass as a function equal to zero everywhere except for zero and whose integral over the entire real line is equal to one. SSP can assign non-zero probability for the event $w = 0$ ($p(w = 0) = 1 - \pi$). Therefore, SSP is the ideal distribution for variable selection. However, the presence of the Dirac delta function δ_0 in the SSP complicates inference. Titsias et al. [222] reformulate SSP as follows:

$$s \sim Bernoulli(\pi), \quad \tilde{w} \sim \mathcal{N}(0, 1), \quad w = \tilde{w} \cdot s. \tag{7.2}$$

This brings two additional variables, $\tilde{w}$ and s, where $\tilde{w}$ represents the weight of the variable and s indicates whether to select the variable. The SSP in Eq. (7.2) is amenable to approximate inference [222].

7.3 Model Description

We first present a personalized FM framework. Then, we propose a Bayesian personalized feature interaction selection (BP-FIS) method within this framework. To incorporate collaborative filtering into BP-FIS, we propose hereditary spike-and-slab priors (HSSPs). Finally, we present the loss function of BP-FIS by conducting variational inference.

7.3.1 Personalized factorization machines

To incorporate P-FIS for FMs, we reformulate Eq. (7.1) as a personalized FM by introducing personalized feature interaction parameters, as indicated in Eq. (7.3):

$$\hat{r}(\boldsymbol{x}) = b_u + \sum_{i=1}^{d} w_{ui} x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} w_{uij} x_i x_j. \tag{7.3}$$

Here, $b_u, \{w_{ui}\}, \{w_{uij}\}$ are the personalized coefficients of user u, and w_{ui} and w_{uij} reflect the preferences of user u over first- and second-order feature interactions. While the FM in Eq. (7.1) can also account for personalization by taking user ID as features, it fails to personalize every interactions, which is required by FIS.

7.3.2 Bayesian personalized feature interaction selection

We formulate a Bayesian generation model, BP-FIS, for Eq. (7.3). The graphical model of BP-FIS is depicted in Figure 7.2. According to BP-FIS, each rating in Eq. (7.3) is generated with the procedure detailed in Algorithm 3. Note that we treat $\{w_{ui}\}, \{w_{uij}\}$ as variables and b_u as the parameter.

In the first part of the generation (line 1–10), we generate personalized feature interaction weights w_{ui} and w_{uij} for all users. We reformulate w_{ui} by $s_{ui} \cdot \tilde{w}_i$ and w_{uij} by $s_{uij} \cdot \tilde{w}_{ij}$ as suggested by [222]. Through the reformulation, we can take advantage of collaborative filtering by learning a single set of feature interaction weights $\tilde{W} = \{\tilde{w}_i\} \cup \{\tilde{w}_{ij}\}$ for all users and operationalize P-FIS by learning the personalized

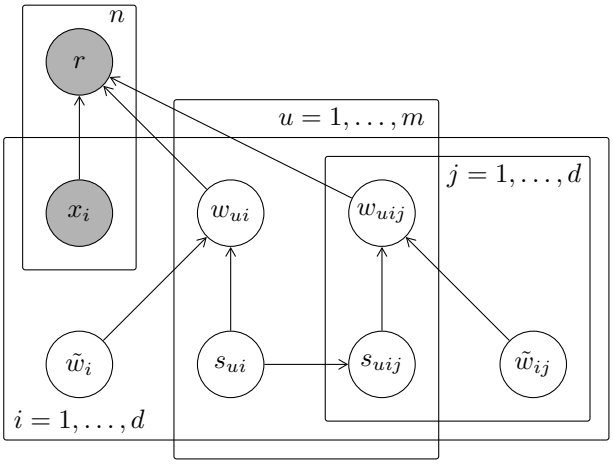


Figure 7.2: Graphical model of BP-FIS. Nodes represent random variables and edges represent dependencies between variables.

interaction selection variable $S = \{s_{ui}\} \cup \{s_{uij}\}$. The generation of s_{uij} is conditioned on s_{ui} and s_{uj}, which captures the hereditary relation between the first- and second-order interactions; see §7.3.3 for details on s_{ui} and s_{uij}.

In the second part (line 11–13), we calculate the rating prediction by Eq. (7.3) and generate the rating $r(\boldsymbol{x})$, following $p(r \mid \hat{r}(\boldsymbol{x}))$. The distribution of $r(\boldsymbol{x})$ determines the likelihood function for optimization. For example, if we assume $r(\boldsymbol{x})$ to follow a Gaussian distribution $\mathcal{N}(\hat{r}(\boldsymbol{x}), 1)$, we can derive the Gaussian log-likelihood:

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} (r(\boldsymbol{x}) - \hat{r}(\boldsymbol{x}))^2. \tag{7.4}$$

If $r(\boldsymbol{x})$ follows a Bernoulli distribution $Bernoulli(\sigma(\hat{r}(\boldsymbol{x})))$, the logistic log-likelihood can be derived:

$$\sum_{\boldsymbol{x} \in \mathcal{X}} r(\boldsymbol{x}) \log \sigma\left(\hat{r}(\boldsymbol{x})\right) + (1 - r(\boldsymbol{x})) \log\left(1 - \sigma(\hat{r}(\boldsymbol{x}))\right). \tag{7.5}$$

The likelihood functions in Eq. (7.4) and (7.5) correspond to the squared loss and cross entropy loss, which are widely used by collaborative filtering methods [141]. Besides, ranking loss can also be derived, e.g., pairwise ranking loss [188] or listwise ranking loss [258]. However, deriving the proper likelihood function for optimization is beyond the scope of this chapter. In this work, we employ the Gaussian likelihood of Eq. (7.4) for optimization.

7.3.3 Hereditary spike-and-slab prior

Although we can learn the personalized parameters in Eq. (7.3) for each user separately, there are two disadvantages to this. First, selecting interactions directly based on Eq. (7.3) fails to take advantage of collaborative filtering. Second, Eq. (7.3) raises some challenges for optimization due to the large number of parameters ($O(md^2)$).

Therefore, we reformulate w_{ui} by $s_{ui} \cdot \tilde{w}_i$ (line 5) and w_{uij} by $s_{uij} \cdot \tilde{w}_{ij}$ (line 10) in the generation procedure in §7.3.2. In order to model s_{ui}, s_{uij}, we propose the hered-

Algorithm 3: Generation procedure

for each feature $i \in \mathcal{F}$ **do**
  draw first-order interaction weight $\tilde{w}_i \sim \mathcal{N}(0, 1)$;
  for each user $u \in \mathcal{U}$ **do**
    draw first-order interaction selection variable $s_{ui} \sim Bernoulli(\pi_1)$;
    $w_{ui} = s_{ui} \cdot \tilde{w}_i$;
for each feature pair $i, j \in \mathcal{F}$ **do**
  draw second-order interaction weight $\tilde{w}_{ij} \sim \mathcal{N}(0, 1)$;
  for each user $u \in \mathcal{U}$ **do**
    draw second-order interaction selection variable $s_{uij} \sim p(s_{uij} \mid s_{ui}, s_{uj}, \pi_2)$;
    $w_{uij} = s_{uij} \cdot \tilde{w}_{ij}$;
for each feature vector $\boldsymbol{x} \in \mathcal{X}$ **do**
  calculate the rating prediction $\hat{r}(\boldsymbol{x})$ by Eq. (7.3);
  draw $r(\boldsymbol{x}) \sim p(r \mid \hat{r}(\boldsymbol{x}))$;

itary spike-and-slab prior (HSSP), which optimizes over SSP by capturing heredity constraints [49] between first- and second-order interactions. The intuition is that there are natural hereditary constraints among the first- and second-order interactions [158], i.e., feature i and feature j are the "parents" of feature interaction (i, j). The hereditary constraints help to dramatically reduce the number of candidate interactions.

The hereditary spike-and-slab prior (HSSP) is based on two hereditary constraints: *strong heredity* and *weak heredity*, where we define as follows based on FMs.

Definition 1 (Strong heredity)**.** *Given a FM,* strong heredity *is the constraint that if the first-order interactions x_i and x_j are selected for the FM, the second-order interaction of their combination, i.e., (x_i, x_j) will also be selected.*

According to *strong heredity*, we have:

$$\text{if } s_{ui} = 1 \text{ or } s_{uj} = 1 \text{ then } s_{uij} = 1.$$

Definition 2 (Weak heredity)**.** *Given a FM,* weak heredity *is the constraint that if one of the first-order interactions x_i or x_j is selected for the FM, then the second-order interaction of their combination (x_i, x_j) will have a non-zero probability to be selected.*

According to *weak heredity*, we have:

$$\text{if } s_{ui} = 1 \text{ or } s_{uj} = 1 \text{ then } p(s_{uij} = 1) > 0.$$

Based on the definitions of *strong heredity* and *weak heredity*, we derive the HSSP by modifying the priors for s_{uij} of SSP:

$$\begin{aligned}
p(s_{ui} = 1) = p(s_{uj} = 1) &= \pi_1 \\
p(s_{uij} = 1 \mid s_{ui} s_{uj} = 1) &= 1 && \text{(\textit{Strong heredity})} \\
p(s_{uij} = 1 \mid s_{ui} + s_{uj} = 1) &= \pi_2 && \text{(\textit{Weak heredity})} \\
p(s_{uij} = 1 \mid s_{ui} + s_{uj} = 0) &= 0,
\end{aligned}$$

where $\pi_1, \pi_2 \in [0, 1]$ are constant values.

7.3.4 Variational inference

To optimize BP-FIS, we need to maximize the posterior $p(\tilde{W}, S \mid \mathcal{R}, \mathcal{X})$, where $S = \{s_{ui}\} \cup \{s_{uij}\}$ and $\tilde{W} = \{\tilde{w}_i\} \cup \{\tilde{w}_{ij}\}$. However, exact inference for $p(\tilde{W}, S \mid \mathcal{R}, \mathcal{X})$ requires an infeasible combinatorial search over $O(2^{md^2})$ possible models. To speed up the process, we conduct variational inference to approximate the posterior $p(\tilde{W}, S \mid \mathcal{R}, \mathcal{X})$ by a variational distribution $q(\tilde{W}, S)$. The closeness of the posterior and the variational distribution is measured by the kullback-leiber (KL)-divergence, i.e., $\mathbb{KL}(q(\tilde{W}, S) \parallel p(\tilde{W}, S \mid \mathcal{R}, \mathcal{X}))$. The KL-divergence can be derived as follows:

$$\begin{aligned} &\mathbb{KL}(q(\tilde{W}, S) \parallel p(\tilde{W}, S \mid \mathcal{R}, \mathcal{X})) \\ &\quad = -\mathbb{E}_q \left[\log p(\tilde{W}, S, \mathcal{R}, \mathcal{X}) - \log q(\tilde{W}, S)\right] + \log p(\mathcal{R}, \mathcal{X}), \end{aligned} \tag{7.6}$$

where $\mathcal{L} = \mathbb{E}_q \left[\log p(\tilde{W}, S, \mathcal{R}, \mathcal{X}) - \log q(\tilde{W}, S)\right]$ is the evidence lower bound (ELBO); $\log p(\mathcal{R}, \mathcal{X})$ is the marginal likelihood which does not depend on $q(\tilde{W}, S)$. Eq. (7.6) states that minimizing the KL-divergence is the same as maximizing the ELBO. To maximize the ELBO, we first discuss the variational distribution $q(\tilde{W}, S)$.

Variational distribution. To ensure the quality of approximation, we assume the hereditary constraints also hold in the variational distributions. Therefore, $q(\tilde{W}, S)$ can be factorized as follows:

$$q(\tilde{W}, S) = \prod_{i=1}^{d} \prod_{j=i+1}^{d} q(\tilde{w}_i) q(\tilde{w}_{ij}) \prod_{u=1}^{m} q(s_{ui}) q(s_{uij} \mid s_{ui}, s_{uj}). \tag{7.7}$$

The factorized distributions are modeled as follows:

$$\begin{aligned} q(w_i \mid \mu_i, \sigma_i) &= \mathcal{N}(\mu_i, \sigma_i) \\ q(w_{ij} \mid \mu_{ij}, \sigma_{ij}) &= \mathcal{N}(\mu_{ij}, \sigma_{ij}) \\ q(s_{ui} \mid \pi_{ui}) &= Bernoulli(\pi_{ui}) \\ q(s_{uij} \mid s_{ui} s_{uj} = 1) &= s_{uij} \\ q(s_{uij} \mid s_{ui} + s_{uj} = 1, \pi_{uij}) &= Bernoulli(\pi_{uij}) \\ q(s_{uij} \mid s_{ui} + s_{uj} = 0) &= 1 - s_{uij}, \end{aligned}$$

where $\rho = \{\mu_i, \sigma_i\} \cup \{\mu_{ij}, \sigma_{ij}\}$ and $\pi = \{\pi_{ui}\} \cup \{\pi_{uij}\}$ are the variational parameters.

Evidence lower bound. Given the variational distribution $q(\tilde{W}, S \mid \rho, \pi)$, the ELBO can be derived as follows:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_q \left[\log p(\tilde{W}, S, \mathcal{R}, \mathcal{X}) - \log q(\tilde{W}, S \mid \rho, \pi)\right] \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_q \left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}))\right] - \mathbb{KL}\left(q(\tilde{W}, S \mid \rho, \pi) \parallel p(\tilde{W}, S)\right), \end{aligned}$$

where $\mathbb{E}_q[\cdot]$ stands for the expectation w.r.t. $q(\tilde{W}, S \mid \rho, \pi)$. In the ELBO, $\mathbb{KL}(q(\tilde{W}, S \mid \rho, \pi) \parallel p(\tilde{W}, S))$ can be analytically derived:

$$\mathbb{KL}\left(q(\tilde{W}, S \mid \rho, \pi) \parallel p(\tilde{W}, S)\right) = \sum_{i=1}^{d} \mathbb{KL}\left(q(\tilde{w}_i) \parallel q(\tilde{w}_i)\right) \sum_{u=1}^{d} \mathbb{KL}\left(q(s_{ui}) \parallel p(s_{ui})\right) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \mathbb{KL}\left(q(\tilde{w}_{ij}) \parallel q(\tilde{w}_{ij})\right) \sum_{u=1}^{m} \mathbb{KL}\left(q(s_{uij} \mid s_{ui}, s_{uj}) \parallel p(s_{uij} \mid s_{ui}, s_{uj})\right),$$

where

$$\begin{aligned}
&\mathbb{KL}\left(q(\tilde{w}_i) \parallel p(\tilde{w}_i)\right) = \frac{1}{2}\left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2\right), \\
&\mathbb{KL}\left(q(\tilde{w}_{ij}) \parallel p(\tilde{w}_{ij})\right) = \frac{1}{2}\left(1 + \log \sigma_{ij}^2 - \mu_{ij}^2 - \sigma_{ij}^2\right), \\
&\mathbb{KL}\left(q(s_{ui}) \parallel p(s_{ui})\right) = \pi_{ui} \log \frac{\pi_{ui}}{\pi_1} + (1 - \pi_{ui}) \log \frac{1 - \pi_{ui}}{1 - \pi_1}, \\
&\mathbb{KL}\left(q(s_{uij} \mid s_{ui}, s_{uj}) \parallel p(s_{uij} \mid s_{ui}, s_{uj})\right) = \\
&\quad (\pi_{ui} + \pi_{uj} - 2\pi_{uij}) \left(\pi_{uij} \log \frac{\pi_{uij}}{\pi_2} + (1 - \pi_{uij}) \log \frac{1 - \pi_{uij}}{1 - \pi_2}\right).
\end{aligned}$$

However, it is problematic to derive $\mathbb{E}_q\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}))\right]$. Therefore, we approximate the expectation with Monte Carlo estimation as follows:

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_q\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}))\right] \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2}\left(r(\boldsymbol{x}) - \hat{r}(\boldsymbol{x})^{(l)}\right)^2, \tag{7.8}$$

where $\hat{r}(\boldsymbol{x})^{(l)}$ is calculated by Eq. (7.3) with the l-th sampling $W^{(l)}$. We write $\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x})^{(l)})$ as a Gaussian log-likelihood as we assume $r(\boldsymbol{x})$ to follow $\mathcal{N}(\hat{r}(\boldsymbol{x}), 1)$ (Eq. (7.4) in §7.3.2).

Reparameterization. When sampling w_{ui} and w_{uij}, we apply the reparameterization trick [116]:

$$\begin{aligned}
\epsilon_1, \epsilon_2 &\sim \mathit{Uniform}(0, 1), \quad \varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1) \\
s_{ui} &= [\![\epsilon_1 \geq \pi_{ui}]\!] \\
s_{uij} &= s_{ui} s_{uj} + (1 - s_{ui} s_{uj})(s_{ui} + s_{uj}) [\![\epsilon_2 \geq \pi_{uij}]\!] \\
\tilde{w}_i &= \mu_i + \varepsilon_1 \sigma_i \\
\tilde{w}_{ij} &= \mu_{ij} + \varepsilon_2 \sigma_{ij} \\
w_{ui} &= \tilde{w}_i \cdot s_{ui}, \quad w_{uij} = \tilde{w}_{ij} \cdot s_{uij}.
\end{aligned} \tag{7.9}$$

By doing so, the stochasticity in the sampling process is isolated and the gradient with respect to $\rho = \{\mu_i, \sigma_i\} \cup \{\mu_{ij}, \sigma_{ij}\}$ can be back-propagated through the sampled w_{ui} and w_{uij}. Unfortunately, the above procedure fails to take the gradient of $\pi = \{\pi_{ui}\} \cup \{\pi_{uij}\}$ due to the discrete nature of $S = \{s_{ui}\} \cup \{s_{uij}\}$. We follow [223] by

marginalizing out the variable of interest:

$$\mathbb{E}_q\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}))\right] = \pi_{ui}\mathbb{E}_{q\setminus\{s_{ui}\}}\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}), s_{ui}=1)\right] + (1-\pi_{ui})\mathbb{E}_{q\setminus\{s_{ui}\}}\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}), s_{ui}=0)\right].$$

The gradient of $\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x})^{(l)} \setminus \{\pi_{ui}\})$ over π_{ui} is:

$$\begin{aligned}\nabla_{\pi_{ui}} &= \mathbb{E}_{q\setminus\{s_{ui}\}}\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}), s_{ui}=1)\right] - \\ &\quad \mathbb{E}_{q\setminus\{s_{ui}\}}\left[\log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x}), s_{ui}=0)\right] \\ &= \frac{1}{L}\sum_{l=1}^{L} \log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x})^{(l)}, s_{ui}=1) - \log p(r(\boldsymbol{x}) \mid \hat{r}(\boldsymbol{x})^{(l)}, s_{ui}=0).\end{aligned}$$

The gradient of π_{uij} can be computed in a similarly way.

Faster inference. To further speed up the variational inference, we factorize the variational parameter π_{uij} as $\pi_{ui} \cdot \pi_{uj}$. In this way, we only need to preserve $\{\pi_{ui}\}$ for each user, decreasing the learning parameters from $O(md^2)$ to $O(md)$. For $\{\tilde{w}_{ij}\}$, we introduce feature embeddings $V \in \mathbb{R}^{d\times k}$ and replace the variational parameters for $\tilde{w}_{ij}$ as follows:

$$\mu_{ij} = \mu(\boldsymbol{v}_i, \boldsymbol{v}_j), \quad \sigma_{ij} = \sigma(\boldsymbol{v}_i, \boldsymbol{v}_j),$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are the transformation functions, and $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^k$ are the embeddings for feature i, j, respectively. Note that we can have different definitions for $\mu(\cdot)$ and $\sigma(\cdot)$. Inspired by [95, 240], we define the transformations as follows:

$$\mu(\boldsymbol{v}_i, \boldsymbol{v}_j) = \boldsymbol{\mu}^T f_\phi(\boldsymbol{v}_i \circ \boldsymbol{v}_j), \quad \sigma(\boldsymbol{v}_i, \boldsymbol{v}_j) = \boldsymbol{\sigma}^T f_\phi(\boldsymbol{v}_i \circ \boldsymbol{v}_j), \tag{7.10}$$

where $\circ$ is the element-wise product operation and $f_\phi(\cdot)$ is the transformation parameterized by ϕ, which transforms a vector from $\mathbb{R}^k$ to $\mathbb{R}^h$; $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the vectors in $\mathbb{R}^h$. Note that $\mu(\cdot)$ and $\sigma(\cdot)$ can be either linear transformations, e.g., $f_\phi(\boldsymbol{v}) = \boldsymbol{v}$, or non-linear transformations, e.g., $f_\phi(\cdot)$ is a neural network. The variational parameter for $\tilde{W}$ is $\rho = \{\phi, \boldsymbol{\mu}, \boldsymbol{\sigma}, V\}$.

Learning. We propose to learn BP-FIS via stochastic gradient variational Bayes (SGVB). As the reparameterization procedures for estimating gradients of ρ and π are different, we propose to optimize the variational parameter π and ρ alternatively. Specifically, at iteration t, by fixing $\rho^{(t-1)}$, we train $\pi^{(t)}$ to update the probabilities that a user will select the specific feature interactions. Then by fixing $\pi^{(t)}$, rather than training $\rho^{(t)}$ based on $\rho^{(t-1)}$, we train $\rho^{(t)}$ from scratch. This is because when π is updated, we will have different user preferences of feature interactions, where ρ should be optimized accordingly. As we have experimented, adapting ρ fitting for $\pi^{(t-1)}$ to $\pi^{(t)}$ could adversely bias the learning.

Prediction. Once the model has been trained, we can generate predictions via point estimation in Eq. (7.11):

$$\mathbb{E}\left[\hat{r}(\boldsymbol{x})\right] = b_u + \sum_{i=1}^{d}\mathbb{E}\left[w_{ui}\right]x_i + \sum_{i=1}^{d}\sum_{j=i+1}^{d}\mathbb{E}\left[w_{uij}\right]x_i x_j, \tag{7.11}$$

where $\mathbb{E}\left[w_{ui}\right] = \pi_{ui}\mu_i$ and $\mathbb{E}\left[w_{uij}\right] = [\pi_{ui}\pi_{uj} + (1-\pi_{ui}\pi_{uj})(\pi_{ui}+\pi_{uj})\pi_{ui}\pi_{uj}]\mu_{ij}$.

Table 7.2: Statistics of the datasets.

Dataset	*#User*	*#Item*	*#Feature*	*#Rating*
MLHt	2,112	5,682	11,945	731,215
LastFM	1,892	17,632	29,200	341,104
Delicious	1,867	69,223	77,735	372,609

7.4 Experimental Setup

We evaluate BP-FIS using the top-N recommendation task.

7.4.1 Research questions

We seek to answer the following research questions:

RQ6.1 Does P-FIS help to improve the performance of FMs on the top-N recommendation task? Specifically, how well does BP-FIS perform against state-of-the-art FMs?

RQ6.2 How does the embedding size impact the ability of BP-FIS to improve the performance of FMs?

RQ6.3 How does the alternating optimization procedure affect the performance of BP-FIS?

RQ6.4 How can BP-FIS provide explainability for the recommendations generated by the FMs?

7.4.2 Dataset

We use three benchmark recommendation datasets from HetRec [29] in our experiments. Summary statistics are provided in Table 7.2.

– *Movielens hetrec (MLHt)*: Extends the MovieLens10M [89] dataset.[1] It links the movies of the MovieLens dataset with their corresponding web pages at the Internet Movie Database (IMDb)[2] and Rotten Tomatoes movie review systems.[3] MLHt only contains users with both rating and tagging information.

– *LastFM*: Contains social networking, tagging, and music artist listening information from the Last.fm online music system.[4] Each user has a list of most listened music artists, tag assignments, and friend relations within the social network.

– *Delicious*: Contains social networking, bookmarking, and tagging information from the Delicious social bookmarking system.[5] Each user has bookmarks, tag assignments, and contact relations within the social network.

For MLHt, users rate each movie with stars, on a scale from 1 to 5. We treat ratings of at least 3 as positive ($r = 1$ if rating ≥ 3) and treat all other ratings as missing ($r = 0$) [107, 239]. For LastFM, we regard user's listening history of artists as implicit ratings, i.e., $r = 1$ if the user listened to a song by the artist. Similarly for Delicious, we regard the bookmarks added by the users as implicit ratings. The side information of these datasets is utilized as additional features, i.e., user tags, movie genres and social

[1] http://www.grouplens.org
[2] http://www.imdb.com
[3] http://www.rottentomatoes.com
[4] http://www.last.fm
[5] http://www.delicious.com

networks. Ratings with less than 10 non-zero feature values are discarded. Therefore, our statistics may slightly differ from the original datasets.

7.4.3 Baselines

To evaluate the effectiveness of BP-FIS, we provide two implementations, namely BP-FM and BP-NFM. They differ in the definition of $f_\phi(\cdot)$ in Eq. (7.10): $f_\phi(\cdot)$ is an identify transformation for BP-FM and a stack of fully connected network layers for BP-NFM. We implement BP-FM and BP-NFM using PyTorch.[6] We compare BP-FM and BP-NFM with the following baseline methods:

– *Factorization machine* (FM) [185]: The FM originally proposed by Rendle. The official implementation is specifically optimized for the rating prediction task, whereas we evaluate the performance with top-N recommendation metrics. Therefore, we provide a PyTorch implementation of FM for a fair comparison.
– *Sparse factorization machine* (SFM) [176, 247, 271]: The FM that learns sparse first- and second-order interactions. We implement SFM, on top of the implementation of FM, to utilize general features for top-N recommendation, where all feature embeddings are penalized by group Lasso regularizations.
– *Attentional factorization machine* (AFM) [240]: The FM that learns the importance of each feature interaction from data via an attention network. We utilize the tensorflow implementation[7] of AFM released by the authors in our experiments.
– *Neural factorization machines* (NFM) [95]: The FM that models higher-order interactions through neural networks, which is the state-of-the-art neural extension of factorization machines. We utilize the tensorflow implementation[8] of neural factorization machines (NFM) released by the authors.

7.4.4 Evaluation

We adopt the leave-one-out evaluation, which has been widely used in the literature [111, 169]. For each user, we randomly hold-out one of her interactions as the test set and utilize the remaining data for training. Since it is too time-consuming to rank all items for every user during evaluation, we follow the common strategy [119, 141] which randomly samples items that are not interacted with by the user, ranking the test item among 100 items. The recommendation quality is measured using hit rate (HR) and average reciprocal hit-rank (ARHR). HR is defined as follows:

$$\text{HR} = \frac{\#Hit}{\#User}, \quad \text{ARHR} = \frac{1}{\#User} \sum_{i=1}^{\#Hit} \frac{1}{p_i},$$

where $\#User$ is the total number of users, and $\#Hit$ is the number of users whose item in the test set is recommended (i.e., a hit) in the size-N recommendation list. p_i is the position of the item in the ranked recommendation list, if an item of a user is among the list. ARHR is a weighted version of HR and it measures how strongly an item is recommended, in which the weight is the reciprocal of the hit position in the recommendation list.

[6] https://pytorch.org/
[7] https://github.com/hexiangnan/attentional_factorization_machine
[8] https://github.com/hexiangnan/neural_factorization_machine

7.4.5 Implementation details

For a fair comparison, we have the following identical experimental settings for all compared methods. (1) All models are optimized using Adam [115]. Adam computes individual adaptive learning rates for different parameters. Therefore, we do not need to tune the learning rate. In practice, setting the initial learning rate as 0.001 provides a good default value. (2) All models are optimized by the mean square loss, accounting for the fairness of comparison and efficiency of training. (3) We hold-out ratings from the training set for validation. We tune parameters of all methods and select the ones with the best performance. (4) We set the maximum training epochs to 50. We apply early-stop for all methods, where we stop training if no further performance gain is observed for 4 successive epochs. (5) Feature embeddings are randomly initialized according to $\mathcal{N}(0, 0.01)$. (6) We tune the parameter k (the latent dimension of feature embeddings) from $64, 128, 256$.

For BP-FIS, we set $\pi_1 = \pi_2 = 0.5$ as we presume no prior knowledge about the selection. For the baselines, we follow the experimental settings in [95, 240] to tune parameters. We tune the ℓ_2-norm regularization parameter for FM in $1e^{-6}, 5e^{-6}, \ldots, 1e^{-1}$. Similarly, we tune the $\ell_{2,1}$-norm regularization parameter for SFM in $1e^{-6}, 5e^{-6}, \ldots, 1e^{-1}$. We use dropout for NFM and AFM. For NFM, we fix the dropout rate of the hidden layers as 0.8 and tune the dropout rate for bi-interaction layer in $0.1, 0.2, \ldots, 1.0$. For AFM, we use dropout for the embedding layer, which is searched from $0.1, 0.2, \ldots, 1.0$. As suggested by the author, we tune the ℓ_2-norm regularization parameter for the attention layer in $0, 0.5, 1, 2, 4, 8, 16$.

For the network structure of NFM and BP-NFM, we follow [95] and set identical dimensions for hidden layers. According to [95], NFM with one hidden layer generates the best performance. Therefore, we also use one hidden layer for NFM and BP-NFM in our experiments. We use ReLU as the activation function.

7.5 Experimental Result and Analysis

We answer the research questions listed in §7.4.1 in four subsections.

7.5.1 RQ6.1: Results

We report the recommendation performance of all models in Table 7.3, in terms of HR@1, HR@10 and ARHR@10. Table 7.3 shows that BP-FM and BP-NFM consistently outperform the other methods. This proves the effectiveness of BP-FIS for improving the performance of both linear and non-linear FMs for top-N recommendation.

To answer RQ6.1, we analyze the results by separating the comparison between linear and non-linear models. Among linear models, SFM outperforms FM and AFM on the LastFM and Delicious datasets. This supports the need for conducting FIS for FMs. However, SFM cannot always improve over FMs. It fails to beat FM on MLHt. In contrast, BP-FM achieves improvements over FM on the same dataset. This shows that selecting or weighing a single set of feature interactions for all users might overlook the personalization over features, and conducting P-FIS is required. Except for HR@1 on LastFM and Delicious, the improvement achieved by BP-FM is significant, especially HR@10 on MLHt, where BP-FM improves FM by 17.286%. This demonstrates the

Table 7.3: Comparison of top-N recommendation methods.

	Method	λ	drop	k	HR@1	HR@10	ARHR@10
MLHt	FM	0.1	–	64	0.2371	0.6028	0.3398
	SFM	0.01	–	64	0.2294	0.6052	0.3351
	AFM	2	0.1	64	0.1138	0.4273	0.1969
	BP-FM	–	–	128	**0.2401***	**0.7070****	**0.3932****
	NFM	–	0.2	256	0.2180	0.6257	0.3389
	BP-NFM	–	–	128	**0.2519****	**0.6831****	**0.3814****
LastFM	FM	0.1	–	64	0.1894	0.6403	0.3215
	SFM	0.05	–	256	0.2118	0.6542	0.3449
	AFM	8	0.7	256	0.2166	0.6126	0.3332
	BP-FM	–	–	256	**0.2209**	**0.6798****	**0.3581****
	NFM	–	0.6	64	0.2150	0.6798	0.3563
	BP-NFM	–	–	256	**0.2257**	**0.6910**	**0.3660**
Delicious	FM	0.1	–	64	0.0202	0.1147	0.0440
	SFM	0.1	–	128	0.0229	0.1212	0.0465
	AFM	2	0.1	64	0.0274	0.1169	0.0494
	BP-FM	–	–	128	**0.0278**	**0.1240****	**0.0509***
	NFM	–	0.1	64	0.0229	0.1065	0.0426
	BP-NFM	–	–	128	**0.0268**	**0.1289****	**0.0504****

Boldface scores indicate best results for linear and non-linear FMs on each metric. We conducted two-sided tests for the null hypothesis that the best and the second best have identical average values. * and ** indicate that the best score is significantly better than the second best score with $p < 0.1$ and $p < 0.05$, respectively.

efficacy of BP-FIS.

The comparison between non-linear models, i.e., NFM and BP-NFM, shows similar results. BP-NFM steadily achieves better performance than NFM on all datasets and all metrics, which shows that the improvement achieved by P-FIS is orthogonal to the non-linear modeling of interactions.

7.5.2 RQ6.2: Impact of embedding size

To answer RQ6.2, we analyze the performance of all models with different embedding sizes. Specifically, we plot figures to show results w.r.t. HR@1, HR@5, HR@10, ARHR@5 and ARHR@10 of all models on the MLHt dataset, as shown in Figure 7.3. Generally, we can see that BP-FM and BP-NFM achieve better a performance than the other models. This demonstrates the robustness of BP-FIS as it constantly improves the performance of FMs, regardless of the embedding size and evaluation metric.

Specifically, for the setting $k = 64$, almost all models show a competitive performance. This is because the space for the performance improvement is limited when $k = 64$ on the MLHt dataset. P-FIS has insignificant effect on FMs due to the restricted embedding size. In contrast, for the setting $k = 128$, while FM, SFM and NFM achieve comparable results, BP-FM and BP-NFM significantly outperform them. This

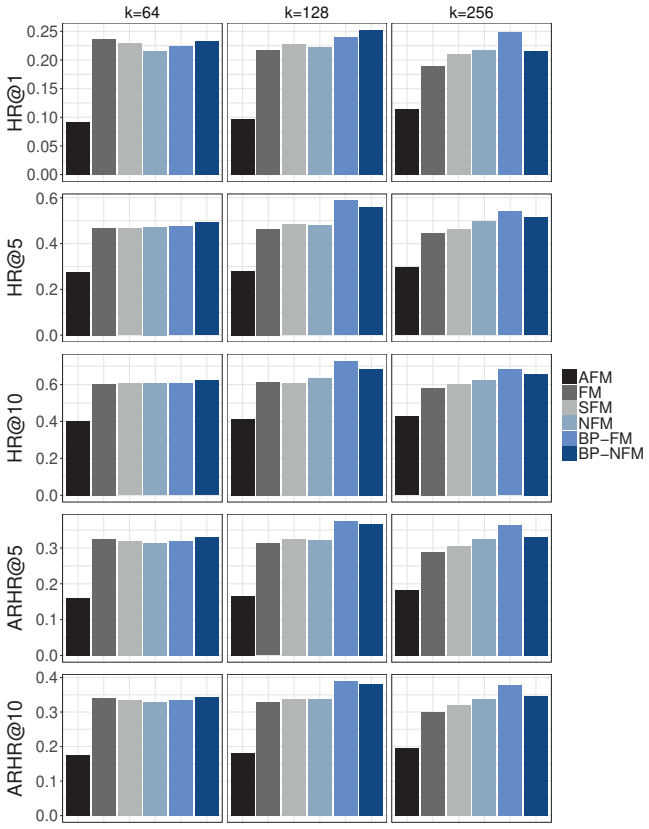


Figure 7.3: Performance comparison w.r.t. different embedding sizes. ARHR@1 is actually HR@1.

means that the effect of P-FIS can be better expressed by increasing the embedding size for FMs. Interestingly, while BP-FM performs better than BP-NFM for most metrics, the performance gain achieved by BP-NFM is more remarkable on HR@1. Similar performance levels are seen when $k = 256$, except that BP-NFM does not outperform BP-FM. This might be because BP-NFM has far more parameters than BP-FM when the embedding size is large, which brings extra difficulty to avoid overfitting.

7.5.3 RQ6.3: Impact of training

To analyze alternative training procedure of BP-FIS, we show the performance of BP-FM and BP-NFM after each iteration in Figure 7.4. A general trend could be revealed by Figure 7.4 that the recommendation performance of both BP-FM and BP-NFM grows initially and fluctuate successively. Although BP-NFM can mostly achieve better performance than BP-FM, it also shows higher variation.

When $k = 64$, BP-FM outperforms BP-NFM w.r.t. HR@1 and performs competitively with BP-NFM w.r.t. HR@5 and HR@10. When $k = 128$, we can witness a relatively stable growth in BP-FM, especially for HR@5 and HR@10. While a certain level of convergence can be witnessed for HR@5 and HR@10, BP-NFM still fluctu-

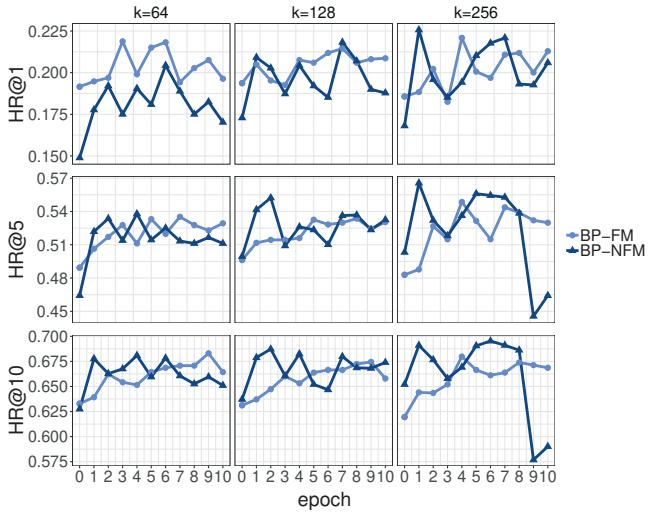


Figure 7.4: Training procedure of BP-FM and BP-NFM on LastFM dataset.

ates more than BP-FM during the training procedure. These observations might suggest that the training of BP-FIS shows better robustness for linear FMs than non-linear ones. BP-NFM is more unstable in terms of HR@5 and HR@10 when $k = 256$, the performance of which drops sharpely during the 9-th iteration. Thus, more iterations do not help a lot for improving the performance but might adversely harm the performance. Another observation is that the training procedure of BP-FM and BP-NFM varies with different embedding sizes. Training BP-FM and BP-NFM with $k = 128$ provides the most stable procedure. This shows that a proper selection of embedding size can further extend the potential of BP-FIS.

7.5.4 RQ6.4: Explainability for recommendation

Table 7.4: Case study.

	48 Hrs.	Spider-Man	Lethal Weapon	True Lies
action (a)	✓	✓	✓	
buddy (b)	✓		✓	✓
clever (cl)	✓			✓
comedy (co)	✓		✓	✓
funny (f)		✓		✓
sequel (se)		✓	✓	
special effects (sp)		✓		✓
user 1	–	top-5	top-1	top-10
user 2	–	top-5	top-5	top-10
user 3	–	–	–	–

Ticks indicate whether a movie has the feature. We also indicate for each user whether the movie is within the top-1, top-5 or top-10 recommendation.

To answer RQ6.4, we provide a case study based on the experiments on the MLHt

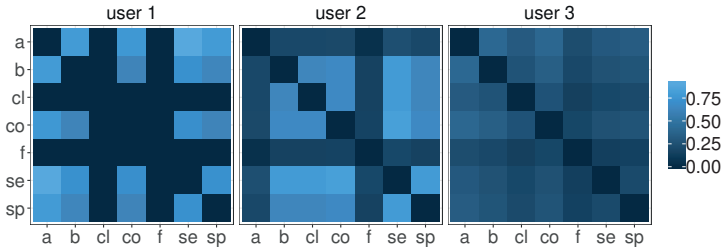


Figure 7.5: Visualization of second-order feature interaction selection w.r.t. the case studies in Table 7.4. The color depth corresponds to the probability of selecting the corresponding interaction. (a, b, cl, co, f, se, sp) are the abbreviations of the features listed in Table 7.4.

dataset. We select four movies ("48 Hrs.", "Spider-Man", "Lethal Weapon" and "True Lies") with seven shared features ("action", "buddy", "clever", "comedy", "funny", "sequel" and "special effects"). We generate the recommendation results with BP-FM for three users ("user 1", "user 2" and "user 3") and check if the four movies are in the top-1, top-5 or top-10 list. The results are shown in Table 7.4.

We also visualize the selection probability of second-order feature interactions ($p(s_{uij} = 1)$) in Figure 7.5. The color depth indicates how likely the corresponding feature interaction will be selected for the user. The probability indicates the predictive power of the feature interaction for the specific user.

Figure 7.5 shows the diverse selections of feature interactions for the three users. All interactions are unlikely to be selected for user 3. Therefore, the four movies sharing these features are not recommended to her. In comparison, several interactions have high probabilities to be selected for user 1 and user 2. Some interactions are predictive for both users, e.g., (sequel, special effects), (comedy, sequel), (buddy, comedy), and some are useful only for a specific user, e.g., (action, buddy), (action, comedy), (action, sequel) for user 1 and (buddy, clever), (buddy, comedy) for user 2.

The commonality of feature interactions for user 1 and user 2 explains the recommendation of "Spider-Man" and "True-Lies". "Spider-Man" has (sequel, special effects) and "True-Lies" has (buddy, commedy), which have been selected for both users. The personalization of feature interactions explains the recommendation of "Lethal Weapon". While the movie has the interaction (buddy, comedy) that was selected for both users, it also has the specific interaction (action, comedy) that is only selected for user 1. Therefore, "Lethal Weapon" is the top-1 item for user 1 and the top-5 item for user 2. On the other hand, "48Hrs." is not recommended to any user. Although the movie has (action, buddy) and (action, comedy) selected for user 1 or (buddy, clever) and (buddy, comedy) selected for user 2, it has no interactions like (comedy, sequel) or (sequel, special effects), which might account largely for the recommendation.

7.6 Related Work

In this section, we survey related work on factorization machines and feature (interaction) selection, respectively.

7.6.1 Factorization machines

FMs have been widely studied and are commonly used in practical systems for their effectiveness and flexibility. Early studies [185, 189] show the generality of FMs. In contrast to matrix factorization (MF) [212] or tensor factorization (TF) [190], which model interactions between categorical variables only, FMs provides a generic way to model interactions between any real valued features. Rendle [185] show that FMs can mimic many of the most successful factorization models (including MF, parallel factor analysis, and SVD++ [119]).

Recently, successive variants of FMs have been developed [3, 23, 24, 105, 143, 152]. They have achieved promising performance in different recommendation scenarios [95, 167, 176, 178, 258]. Juan et al. [105] propose the field-aware factorization machine (FFM) to factorize the interactions of fields (the category of features). Blondel et al. [23] present the higher-order factorization machine (HOFM), which provides an efficient algorithm to train FMs with higher-order interactions. Besides rating prediction, FMs have also been optimized for the top-N recommendation task. Yuan et al. [258] introduce boosted factorization machiness (boostFMs) to incorporate contextual information into FMs for context-aware recommendation (CAR) [159, 167, 189]. Xiao et al. [240] propose the attentional factorization machine (AFM), which uses an attention network to learn the importance of each feature interaction. To investigate the linear expressiveness limitation of FMs, He et al. [95] propose neural factorization machiness (NFMs), which perform non-linear transformations on the latent space of second-order feature interactions.

Despite the effectiveness of modeling various feature interactions, existing FM variants suffer from the high-dimensionality issue which limits their application in high-dimensional scenarios.

7.6.2 Feature selection

An effective approach to alleviate the high-dimensionality issue of FMs is feature selection. Cheng et al. [45] propose to select feature interactions though a greedy interaction feature selection algorithm based on gradient boosting. Xu et al. [247] apply group Lasso [259] to user and item feature embeddings to select interactions between user and item features. Zhao et al. [271] propose to select meta-graph based features. Similarly, they also apply group Lasso for feature selection. Mao et al. [156] propose to select context features for FMs. They first choose features based on predictive power. Then, they subsample the set of features selected in the first step.

Feature selection has also been well investigated for feature-based recommender systems (FRSs), which often utilize high-dimensional auxiliary information. Ronen et al. [193] propose to select content features. Their algorithm selects the most informative features by computing relevance scores based on pluggable feature similarity functions. Koenigstein et al. [118] propose to select content features for Xbox movies by incorporating sparsity priors on feature parameters. Different feature weighing methods have also been proposed to select context features [274]. Li et al. [130] study personalized feature selection for unsupervised learning; they learn a specific model for each user, which is only applicable with a limited number of users.

The differences between our method and the above methods are at least three-fold.

First, we target personalized feature interaction selection, which better captures a user's personalized preferences over different features. Second, we provide a generic way to achieve feature selection, which can be seamlessly integrated to different FM variants. Third, we opt for Bayesian variable selection with spike-and-slab priors, rather than the sparsity induced regularizations that have previously been considered.

7.7 Summary

In this chapter, we have answered **RQ6** by studying a generic method to utilize high-dimensional and heterogeneous information for top-N recommendations. We have proposed a Bayesian personalized feature interaction selection (BP-FIS) method to address the personalized feature interaction selection (P-FIS) task for factorization machines (FMs). BP-FIS fuses hereditary spike-and-slab prior (HSSP) to achieve P-FIS while taking advantage of collaborative filtering. We have conducted variational inference and proposed a stochastic gradient variational Bayes (SGVB) method to optimize BP-FIS. Experimental results show that BP-FIS significantly improves the performance of both linear and non-linear FMs. Further analytical experiments show that: (1) BP-FIS is effective for both linear and non-linear FMs; (2) BP-FIS is robust for performance gain, regardless of the embedding size; and (3) it is preferable to train BP-FIS with a limited number of iterations.

Unlike previous chapters that works on top-N recommendation with high-dimensional information or heterogeneous information only, we provide a generic method to utilize both high-dimensional and heterogeneous information in this chapter. Next, in Chapter 8, we conclude the thesis by harvesting the answers to our research question and formulating ideas for future work.

8 Conclusions

In the previous chapters, we have introduced how we proposed methods to address the research questions raised in Chapter 1. In this chapter, we first revisit our research questions introduced in Chapter 1 and summarize the main findings and implications of our research in § 8.1. Then, in § 8.2, we describe the main limitations of our work and possible future directions.

8.1 Main Findings

8.1.1 Item clustering for top-N recommendation

We started with clustering items for top-N recommendation and asked:

RQ1 How can we effectively perform item clustering for item-based collaborative filtering methods?

To answer this question, we proposed a method called Block-aware similarity regularization (BSR) for item-based collaborative filtering (ICF) in Chapter 2. BSR is a regularization term for ICF methods that encourages the learned item similarity matrix to be, or to be close to, a c-block diagonal, where c is the number of blocks. Unlike existing clustering-based methods that intrinsically treat clustering and model estimation as separate procedures, BSR integrates item clustering into the learning of item similarities, where in-block similarities are encouraged and off-block similarities are penalized. We applied BSR to Similarity models (SMs) and proposed a block regularized similarity model (BSM). We proved several theoretical properties of BSM, namely block-diagonality, sparsity and transitivity. We further extended BSR to block-aware similarity dropout (BSD). We regularized feature-based similarity models (FSMs) by BSR and proposed a block regularized factored similarity model (BFSM).

We empirically evaluated the effectiveness of BSR by extensive experiments. Our analysis indicates that: a) item groups exist in many real-world applications and BSR is able to capture grouping property to improve top-N recommendation. SMs generally outperform FSMs for top-N recommendations and BSM further improves the performance of SMs significantly; b) BSR can positively impact the performance of top-N recommendation. Although solely applying it has a negative effect on the performance, the similarity constraint provides an excellent remedy; and c) besides improving performance, BSR also helps with stability at the cost of a slight slow down of

the convergence rate.

8.1.2 Leveraging high-dimensional side information

To utilize high-dimensional side information to overcome the sparsity of ratings, we resorted to feature reduction techniques and asked:

RQ2 Can we reduce the dimensions of side information for an effective top-N recommendation?

To answer **RQ2**, we performed feature reduction via Locality preserving projection (LPP) based on the manifold assumption that we can expect an intrinsic low-dimensional feature behind the high-dimensional side information. We presented a joint learning model to simultaneously perform LPP and learn item similarities. An alternating optimization method has conceived to solve the model. We experimented with the proposed projection regularized item similarity model (Prism) on four datasets with high-dimensional side information and compared it with state-of-the-art baselines. The experimental evaluation shows that the proposed method enjoys a performance gain of up to 21.2% on hit rate at 10 (HR@10) and 36.8% on average reciprocal hit-rank at 10 (ARHR@10).

However, the effectiveness of Prism can be ensured only if the side information is noise-free. Unfortunately, in many real-world applications, side information contains noise where we can hardly expect the intrinsic low-dimensional features. In order to utilize noisy side information, we sought to answer **RQ3**:

RQ3 How can we utilize high-dimensional side information with noise for top-N recommendation?

To utilize high-dimensional and noisy side information, we took advantage of variational auto-encoders (VAEs), which has shown a strong property of denoising. Instead of learning item representations via VAE as existing methods, we learned feature embeddings. By learning feature embeddings from side information, the model is no longer affected by high-dimensionality, because the dimension of side information is the number of inputs to VAE rather than the input scale. Inspired by collective sparse linear method (cSLIM) [170], we proposed a collective variational auto-encoder (cVAE), which can be regarded as a neural extension to cSLIM. Since user factors and feature embeddings are both encoded from and decoded to information with respect to items, we proposed to learn them collectively by sharing network parameters. We evaluated cVAE on two datasets from Amazon, both of which are extremely high-dimensional. On the datasets, state-of-the-art VAE-based methods show poor performance, being outperformed by the linear method cSLIM. In comparison, cVAE improves over cSLIM significantly.

8.1.3 Cold-start recommendation

We then focused on another task, recommending new items (Chapter 5), and asked:

RQ4 How can we effectively fuse item features with ratings for the recommendation of top-N new items?

To recommend new items, item features is utilized to represent items. Therefore, even

for new items, we can still generate recommendations since item features can characterize items. We provided an ICF method, local variational feature-based similarity model (LVSM), to learn item similarities based on item features. We estimated multiple local similarity functions based on user groups and a single global similarity function to understand item similarities comprehensively. LVSM learns local similarities directly from item features and learns global similarities based on the item representation encoded by VAE. LVSM is a Bayesian generative model that seamlessly integrates ICF with deep learning and user clustering. To efficiently optimize LVSM, we conducted variational inference, based on which we proposed a variational expectation maximization (EM) algorithm.

Empirically, while FSMs are superior amongest feature-based methods, LVSM outperforms other FSMs significantly. LVSM can also provide robust recommendations even when ratings or item features are extremely sparse. An evaluation on scalability showed that LVSM can scale to large-scale datasets and it is even more powerful compared with other baselines. The only limitation of LVSM is that it is mathematically more difficult than other feature-based methods.

Besides cold-start items, we also studied cold-start users in recommendation. We worked on scientific paper recommendations and asked:

RQ5 Can we address the challenge of recommending papers to cold-start users by effectively utilizing the available heterogeneous information?

We examined an interesting recommendation scenario for an academic search engine, namely, to rerank paper recommendations in email newsletters for newly signed up users. To answer **RQ5**, we proposed an approach to rerank candidate recommendations that utilizes both paper content and user behavior. To handle cold-start users, we proposed a model to learn a mapping from users' browsed articles to user clicks on the recommendations. We combine both content and behavior into a hybrid reranking model (HRM). HRM is trained using a pairwise learning approach based on real user interaction data. The outputs are reranked paper recommendations that are significantly better than the production system in terms of yielding user clicks. We found that both content and behavior contribute to better recommendations. We also detailed individual contributions from different paper aspects and components of the model. Our reranking model can be applied to production recommender systems as a separate module.

A limitation of the study is that we have not performed online evaluations, such as A/B testing, to validate the model's effect on user engagement. Another limitation is due to the production dataset: our reranking is limited to the candidate articles generated by the production system. Therefore, if the inputs are not of high quality, it will impact our final recommendation performance.

8.1.4 Personalized feature interaction selection

Finally, we worked on factorization machines (FMs), which provide a way to utilize heterogeneous information for recommendations. FMs are effective for recommendation due to the modeling of feature interactions with factored parameters. We studied feature interaction selection (FIS) for FMs and raised the following question:

RQ6 How can we integrate high-dimensional heterogeneous information for top-N recommendation?

To answer **RQ6**, we provided a model based on Bayesian variable selection (BVS) to select first- and second-order feature interactions for FMs. In the scenario of recommendation, where features or feature interactions are not necessarily equally powerful to predictions for all users, we proposed the problem of personalized feature interaction selection (P-FIS). To complete the task, we provided Bayesian personalized feature interaction selection (BP-FIS), a plug-and-play framework for FMs. Based on the spike-and-slab prior (SSP), we proposed a hereditary spike-and-slab prior (HSSP), which also captures hierarchical relations among first- and second-order feature interactions. To evaluate BP-FIS, we implemented two methods following the framework, both linear and non-linear. The empirical evaluation validated the need for personalized feature interaction selection (P-FIS) when utilizing high-dimensional and heterogeneous information for FMs and the effectiveness of BP-FIS to achieve the task.

8.2 Future work

We list possible directions for future work grouped by our main research questions.

8.2.1 RQ1: Scalability and online grouping

The principal shortcoming of BSR we proposed for **RQ1** is that the theoretical properties of BSM do not hold for BFSM. BSR is also less empirically effective for FSMs than for SMs. Unfortunately, BSM cannot scale to datasets with a large number of items. It is interesting to find a better way to balance scalability and theoretical properties. A possible solution is to design a regularization to regularize factored item representations directly.

The effectiveness of BSR has been demonstrated in offline recommendations, theoretically and empirically. In an online recommendation scenario, clustering items on-the-fly has been less studied. Existing work periodically updates clusters in online settings, which actually reduces the problem to offline. In comparison, utilizing BSR to investigate online ICF helps to capture the dynamicity of item groups in real-time. The problem can be even more interesting if new items are arriving in real-time. Recent work has raised this new issue, referred to as "online cold-start recommendation" [5]. Assigning new items to the proper groups in real-time can find its application in news recommendation. Therefore, extending BSR to online clustering opens up a new perspective to solve this challenging yet significant problem.

8.2.2 RQ2 & RQ3: Fine-designed neural extension

To answer **RQ2** and **RQ3**, we studied a linear and a neural method, respectively, for the sake of leveraging high-dimensional side information for top-N recommendation. Similar to the situation we studied for item grouping, we faced the dilemma that the linear method has good theoretical properties while the neural method enjoys better scalability and empirical performance with a sacrifice on the theoretical guarantees. A possible break to this dilemma is to extend the linear model with good theoretical properties to neural methods. Although neural methods do not have rigorous theories, they can capture the intuition of linear methods with rigorous theories, where improvements

achieved through neural methods are easier to interpret. Note that cVAE is regarded as an extension of the linear model cSLIM. It will also be interesting to extend Prism to neural methods.

cVAE also has limitations. cVAE is designed to encode from and decode to both ratings and side information via networks with shared parameters. This implies an underlying assumption that item similarity information encoded by side information should be similar to the same information contained in ratings. Unfortunately, this assumption is not always true in real applications. A better design of the network structure is needed, in order to share parameters of networks while also accounting for the information difference between ratings and side information.

8.2.3 RQ4 & RQ5: Incorporating semantic information

We answered **RQ4** by proposing the local variational feature-based similarity model. LVSM shows strong and robust performance for recommending top-N new items. Improvement in performance can be further expected if we can also utilize the semantic information behind item features. In our experiments, although we utilized text features, we ignored semantic information as LVSM is designed to utilize generic content features. Potentially valuable semantic information is ignored in order to ensure the generalization of LVSM. The model we studied for **RQ5** also reveals the advantage of combining semantic information with user behavior. In practice, text features are widely available. In future, we can optimize LVSM specifically for semantic information. The straightforward way is to replace or initialize the latent item representations by the summation or concatenation of word embeddings. A better way is to design a sequential VAE to capture the sequential information behind text features. It is also possible to extend LVSM to integrate semantic analysis, e.g., sentiment analysis [64], topic modeling [229] and etc.

8.2.4 RQ6: Group-level personalized selection

While BP-FIS provides promising results to answer **RQ6**, it is time-consuming to train. One way to speed up is to conduct feature interaction selection in group-level, i.e., select a set of feature interactions for each user group rather than for individuals. The group-level personalized selection is expected to be much more efficient, with a little reduction in performance. A straightforward implementation is to pre-define user groups. However, it will be more interesting to integrate user clustering with BP-FIS. We can design a Bayesian generative model, where user clustering and feature interaction selection are simultaneously learned and mutually enhanced.

Currently, BP-FIS is specially designed for FMs with first- and second-order feature interactions. Higher-order factorization machine (HOFM) [23] provides an efficient way to infer higher-order feature interactions. Higher-order feature interactions are more powerful to capture the complex relations behind features. The problem brought by high-dimensionality can be even more severe in the HOFM. Studying P-FIS for HOFM can address the issue of high-dimensionality and further boost the performance of HOFM. Besides the performance gain, it also provides a way to explain higher-order relations, which are extremely difficult to explain for deep neural network (DNN).

BP-FIS also reveals a certain connection with dropout [213]. This also explains

why BP-FIS improves the performance of FMs. BP-FIS shares a certain level of similarity with the adaptive dropout, where the dropout rates are adaptively adjusted during training. It is interesting to further study the connection and difference between BP-FIS and dropout in theory. The power of BP-FIS should not be restricted to FMs but also be generalized to broader machine learning methods.

Acronyms

Short	Full
AFM	attentional factorization machine (*see pp.* 120, 126)
AP@N	average precision at N (*see p.* 47)
ARHR	average reciprocal hit-rank (*see pp.* 25, 36, 120)
ARHR@10	average reciprocal hit-rank at 10 (*see pp.* 32, 130)
BDR	block-diagonal representation (*see p.* 16)
BFSM	block regularized factored similarity model (*see pp.* 7, 15, 21, 23, 129)
boostFM	boosted factorization machines (*see p.* 126)
BP-FIS	Bayesian personalized feature interaction selection (*see pp.* 6, 7, 111, 127, 132, 153)
BPR	bayesian personalized ranking (*see p.* 24)
BSD	block-aware similarity dropout (*see pp.* 15, 22, 30, 129)
BSM	block regularized similarity model (*see pp.* 6, 15, 16, 129)
BSR	block-aware similarity regularization (*see pp.* 6, 13, 14, 16, 30, 129)
BVS	Bayesian variable selection (*see pp.* 6, 111, 132, 153)
CAR	context-aware recommendation (*see p.* 126)
CBF	content-based filtering (*see p.* 105)
CF	collaborative filtering (*see pp.* 1, 13, 16, 39, 105, 153)
cfVAE	collaborative variational auto-encoder (*see pp.* 47, 73)
coSim	simple cosine-similarity (*see pp.* 36, 72, 73)
cSLIM	collective sparse linear method (*see pp.* 41, 130)
CUL-a	CiteULike article (*see pp.* 70, 86)
CUL-t	CiteULike tag (*see pp.* 70, 86)
cVAE	collective variational auto-encoder (*see pp.* vi, 6, 7, 40, 130)
DAE	denoising auto-encoder (*see pp.* 4, 15)
DCG@N	discounted cumulative gain at N (*see p.* 71)
DNN	deep neural network (*see pp.* 22, 63, 86, 133)
ELBO	evidence lower bound (*see pp.* 45, 57, 111)
EM	expectation maximization (*see pp.* 7, 65, 86, 131)
FAE	feature-side autoencoder (*see p.* 42)
FBSM	feature-based factorized bilinear similarity model (*see p.* 61)
FFM	field-aware factorization machine (*see p.* 126)
FIS	feature interaction selection (*see pp.* 5, 110, 131)
FISM	factored item similarity model (*see pp.* 13, 15, 106)
FM	factorization machine (*see pp.* viii, 5, 109, 127, 131, 153)
FRS	feature-based recommender system (*see p.* 126)
FSM	feature-based similarity model (*see pp.* 15, 30, 56, 86, 87, 129)
HOFM	higher-order factorization machine (*see pp.* 126, 133)
HR	hit rate (*see pp.* 25, 36, 120)
HR@10	hit rate at 10 (*see pp.* 32, 130)
HRM	hybrid reranking model (*see pp.* 2, 5, 6, 90, 131, 153)
HSSP	hereditary spike-and-slab prior (*see pp.* 111, 113–115, 127, 132)
IAE	item-side autoencoder (*see p.* 42)
ICF	item-based collaborative filtering (*see pp.* 2, 3, 13, 15, 30, 31, 129, 153)
IFM	item feature mapping (*see p.* 59)
item$k$NN	item-based k-nearest-neighbor (*see pp.* 24, 26, 31)

KKT	Karush-Kuhn-Tucker (*see p.* 21)
KL	kullback-leiber (*see pp.* 44, 116)
LCE	local collective embeddings (*see pp.* 73, 74)
LFM	latent factor model (*see p.* 59)
LOOCV	leave-one-out cross-validation (*see pp.* 25, 36)
lorSLIM	low-rank sparse linear method (*see pp.* 14, 15, 107)
LPP	locality preserving projection (*see pp.* 4, 32, 130)
LSM	local feature-based similarity model (*see p.* 79)
LVSM	local variational feature-based similarity model (*see pp.* 6, 7, 56, 86, 131)
MAP	maximum a posteriori (*see p.* 65)
MAP@N	mean average precision at N (*see p.* 48)
MF	matrix factorization (*see p.* 126)
MLHt	movielens hetrec (*see p.* 119)
MLP	multi layer perceptron (*see pp.* 45, 73)
MRR	mean reciprocal rank (*see p.* 25)
mVAE	multinomial variational auto-encoder (*see pp.* 25, 47)
NAIS	neural attentive item similarity model (*see p.* 24)
NDCG	normalized discounted cumulative gain (*see p.* 25)
NFM	neural factorization machines (*see p.* 120)
NSPR	neural semantic personalized ranking (*see p.* 73)
P-FIS	personalized feature interaction selection (*see pp.* 6, 109–111, 127, 132)
PFW	personalized feature weighting (*see p.* 60)
Pre@N	precision at N (*see p.* 47)
Prism	projection regularized item similarity model (*see pp.* 6, 36, 130)
pureSVD	pure singular-value-decomposition (*see p.* 25)
Rec@N	recall at N (*see pp.* 47, 71)
ReLU	rectified linear unit (*see p.* 91)
SFM	sparse factorization machine (*see p.* 110)
SGD	stochastic gradient descent (*see pp.* 23, 101)
SGVB	stochastic gradient variational Bayes (*see pp.* 7, 111, 127)
SLIM	sparse linear method (*see pp.* 13, 15, 31, 41, 106)
SM	similarity model (*see pp.* 15, 30, 129)
SSLIM	sparse linear method with side information (*see p.* 32)
SSP	spike-and-slab prior (*see pp.* 111, 112, 132)
SVDFeature	feature-based singular value decomposition (*see p.* 72)
SVM	support vector machine (*see p.* 109)
t-SNE	t-distributed stochastic neighbor embedding (*see p.* 81)
TF	tensor factorization (*see p.* 126)
UAE	user-side autoencoder (*see p.* 42)
UFSM	user-specific feature-based similarity model (*see pp.* 32, 61)
UM	user modeling (*see p.* 59)
VAE	variational auto-encoder (*see pp.* 2, 15, 39, 62, 63, 86, 111, 130, 153)
WRMF	weighted regularized matrix factorization (*see p.* 25)

Bibliography

[1] Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '09), 19–28 (cited on pages 56, 59).

[2] Eugene Agichtein, Eric Brill and Susan T. Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '06), 19–26 (cited on page 1).

[3] Michal Aharon, Natalie Aizenberg, Edward Bortnikov, Ronny Lempel, Roi Adadi and etc. 2013. Off-set: one-pass factorization of feature sets for online recommendation in persistent cold start settings. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys '13), 375–378 (cited on page 126).

[4] Fabio Aiolli. 2013. A preliminary study on a recommender system for the million songs dataset challenge. In *Proceedings of the 4th Italian Information Retrieval Workshop* (CEUR '13), 73–83 (cited on pages 13, 60, 106).

[5] Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré and Antoine Cornuéjols. 2018. Adaptive collaborative topic modeling for online recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems* (RecSys '18), 338–346 (cited on page 132).

[6] Marie Al-Ghossein, Talel Abdessalem and Anthony Barré. 2018. Dynamic local models for online recommendation. In *Proceedings of the 27th International Conference on World Wide Web Companion* (WWW '18), 1419–1423 (cited on pages 14, 16).

[7] Mohammad Aliannejadi and Fabio Crestani. 2018. Personalized context-aware point of interest recommendation. *ACM Trans. Inf. Syst.*, 36, 4, 45:1–45:28 (cited on page 59).

[8] James Allan, W. Bruce Croft, Alistair Moffat and Mark Sanderson. 2012. Frontiers, challenges, and opportunities for information retrieval: report from SWIRL the second strategic workshop on information retrieval in lorne. *SIGIR Forum*, 46, 1, 2–32 (cited on page 1).

[9] Taha Alshaybawee, Rahim Alhamzawi, Habshah Midi and Intisar Ibrahim Allyas. 2018. Bayesian variable selection and coefficient estimation in heteroscedastic linear regression model. *Journal of Applied Statistics*, 45, 14, 2643–2657 (cited on page 111).

[10] Emmanouil Amolochitis, Ioannis T. Christou, Zheng-Hua Tan and Ramjee Prasad. 2013. A heuristic hierarchical scheme for academic search and retrieval. *Inf. Process. Manage.*, 49, 6, 1326–1343 (cited on page 104).

[11] Cédric Archambeau and Francis R. Bach. 2008. Sparse probabilistic projections. In *Proceedings of the 22nd Neural Information Processing Systems* (NlPS '08), 73–80 (cited on page 111).

[12] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley (cited on page 1).

[13] Evangelos Banos, Ioannis Katakis, Nick Bassiliades, Grigorios Tsoumakas and Ioannis P. Vlahavas. 2006. Personews: a personalized news reader enhanced by machine learning and semantic filtering. In *Proceedings of On the Move to Meaningful Internet Systems* (OTM '06), 975–982 (cited on page 59).

[14] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian and Hayder Radha. 2015. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems* (RecSys '15), 91–98 (cited on page 56).

[15] Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian and Hayder Radha. 2016. Cold-start recommendation with provable guarantees: A decoupled approach. *IEEE Trans. Knowl. Data Eng.*, 28, 6, 1462–1474 (cited on page 56).

[16] Konstantin Bauman, Bing Liu and Alexander Tuzhilin. 2017. Aspect based recommendations: recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '17), 717–725 (cited on pages 61, 62).

[17] Immanuel Bayer, Xiangnan He, Bhargav Kanagal and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web* (WWW '17), 1341–1350 (cited on page 25).

[18] Joeran Beel, Akiko Aizawa, Corinna Breitinger and Bela Gipp. 2017. Mr. dlib: recommendations-as-a-service (raas) for academia. In *Proceedings of ACM/IEEE Joint Conference on Digital Libraries* (JCDL, 17), 1–2 (cited on page 105).

[19] Mikhail Belkin. 2003. *Problems of Learning on Manifolds*. PhD thesis. The University of Chicago (cited on page 22).

[20] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 15th Neural Information Processing Systems* (NIPS '01), 585–591 (cited on page 22).

[21] Alex Beutel, Ed Huai-hsin Chi, Zhiyuan Cheng, Hubert Pham and John R. Anderson. 2017. Beyond globally optimal: focused learning for improved recommendations. In *Proceedings of the 26th International Conference on World Wide Web* (WWW '17), 203–212 (cited on pages 16, 61).

[22] Daniel Billsus and Michael J Pazzani. 1999. A hybrid user model for news story classification. In *Proceedings of the 7th International Conference on User Modeling* (UM '99), 99–108 (cited on pages 60, 72–77).

[23] Mathieu Blondel, Akinori Fujino, Naonori Ueda and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Proceedings of the 30th Neural Information Processing Systems* (NIPS '16), 3351–3359 (cited on pages 126, 133).

[24] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino and Naonori Ueda. 2016. Polynomial networks and factorization machines: new insights and efficient training algorithms. In *Proceedings of the 33nd International Conference on Machine Learning* (ICML '16), 850–858 (cited on pages 109, 126).

[25] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th Neural Information Processing Systems* (NeurIPS '13), 2787–2795 (cited on pages 93–95).

[26] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30, 1-7, 107–117 (cited on page 1).

[27] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: an overview. *Learning*, 11, 23-581, 81 (cited on page 100).

[28] Deng Cai, Xiaofei He, Xiaoyun Wu and Jiawei Han. 2008. Non-negative matrix factorization on manifold. In *Proceedings of the 8th IEEE International Conference on Data Mining* (ICDM '08), 63–72 (cited on page 22).

[29] Iván Cantador, Peter Brusilovsky and Tsvi Kuflik. 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM Conference on Recommender Systems* (RecSys '11) (cited on page 119).

[30] Laurent Charlin, Richard S Zemel and Hugo Larochelle. 2014. Leveraging user libraries to bootstrap collaborative filtering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '14), 173–182 (cited on page 105).

[31] Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao and Yong Yu. 2012. Collaborative personalized tweet recommendation. In *Proceedings of the 35th International ACM SIGIR conference on research and development in Information Retrieval* (SIGIR '12), 661–670 (cited on page 109).

[32] Li Chen, Guanliang Chen and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Model. User-Adapt. Interact.*, 25, 2, 99–154 (cited on page 61).

[33] Lini Chen, Xin Xin, Dong Wong and Yue Ding. 2017. Hcom: item-based similarity model for heterogeneous implicit feedback. In *Proceedings of the 18th IEEE International Conference on Mobile Data Management* (MDM '17), 40–49 (cited on page 2).

[34] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng and Yong Yu. 2012. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, 13, 3619–3622 (cited on pages 59, 72–77, 101).

[35] Xu Chen, Zheng Qin, Yongfeng Zhang and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (SIGIR '16), 305–314 (cited on page 62).

[36] Yifan Chen and Maarten de Rijke. 2018. A collective variational autoencoder for top-n recommendation with side information. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems* (DLRS '18@RecSys), 3–9 (cited on pages 8, 39).

[37] Yifan Chen, Pengjie Ren, Yang Wang and Maarten de Rijke. 2019. Bayesian personalized feature interaction selection for factorization machines. In *Proceedings of the 42th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '19), 665–674 (cited on pages 9, 59, 109).

[38] Yifan Chen, Yang Wang, Xiang Zhao, Jie Zou and Maarten de Rijke. 2019. Block-aware similarity regularizations for item-based collaborative filtering. *ACM Trans. Inf. Syst.* Under review. (cited on pages 8, 13).

[39] Yifan Chen, Xiang Zhao, Junjiao Gan, Junkai Ren and Yanli Hu. 2016. Content-based top-n recommendation using heterogeneous relations. In *Proceedings of the 27th Australasian Database Conference* (ADC '16), 308–320 (cited on page 9).

[40] Yifan Chen, Xiang Zhao, Xuemin Lin, Yang Wang and Deke Guo. 2019. Efficient mining of frequent patterns on uncertain graphs. *IEEE Trans. Knowl. Data Eng.*, 31, 2, 287–300 (cited on pages 9, 109).

[41] Yifan Chen, Xiang Zhao, Jinyuan Liu, Bin Ge and Weiming Zhang. 2019. Learning to select user-specific features for top-n recommendation of new items. *Journal of Computer Science and Technology*. Under review. (cited on page 9).

[42] Yifan Chen, Yang Wang, Xiang Zhao, Hongzhi Yin, Ilya Markov and Maarten de Rijke. 2019. Local variational feature-based similarity models for recommending top-n new items. *ACM Trans. Inf. Syst.* Under review. (cited on pages 8, 55).

[43] Yifan Chen, Xiang Zhao and Maarten de Rijke. 2017. Top-n recommendation with high-dimensional side information via locality preserving projection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '17). ACM, 985–988 (cited on pages 2, 3, 8, 16, 31, 40, 41, 109).

[44] Zheng Chen, Minmin Chen, Kilian Q. Weinberger and Weixiong Zhang. 2015. Marginalized denoising for link prediction and multi-label learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (AAAI '15), 1707–1713 (cited on page 1).

[45] Chen Cheng, Fen Xia, Tong Zhang, Irwin King and Michael R. Lyu. 2014. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender Systems* (RecSys '14), 265–272 (cited on pages 110, 126).

[46] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (DLRS@RecSys '16), 7–10 (cited on pages 63, 109).

[47] Yao Cheng, Li'ang Yin and Yong Yu. 2014. Lorslim: low rank sparse linear methods for top-n recommendations. In *Proceedings of the 15th IEEE International Conference on Data Mining* (ICDM '14), 90–99 (cited on pages 14, 15, 24, 106, 107).

[48] Zhiyong Cheng, Ying Ding, Lei Zhu and Mohan S. Kankanhalli. 2018. Aspect-aware latent factor model: rating prediction with ratings and reviews. In *Proceedings of the 27th International Conference on World Wide Web* (WWW '18), 639–648 (cited on pages 61, 62).

[49] Nam Hee Choi, William Li and Ji Zhu. 2010. Variable selection with the strong heredity constraint and its oracle property. *Journal of the American Statistical Association*, 105, 489, 354–364 (cited on page 115).

[50] Szu-Yu Chou, Yi-Hsuan Yang, Jyh-Shing Roger Jang and Yu-Ching Lin. 2016. Addressing cold start for next-song recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems* (RecSys '16), 115–118 (cited on page 56).

[51] Evangelia Christakopoulou and George Karypis. 2016. Local item-item models for top-n recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems* (RecSys '16), 67–74 (cited on pages 13, 14, 16, 25, 55, 56, 58, 61, 106, 107).

[52] Evangelia Christakopoulou and George Karypis. 2018. Local latent space models for top-n recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (SIGKDD '18), 1235–1243 (cited on pages 14, 16, 55, 58, 61).

[53] Fan RK Chung. 1997. *Spectral Graph Theory*. Number 92. American Mathematical Soc. (cited on pages 17, 22, 98).

[54] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan and Shun-ichi Amari. 2009. *Nonnegative Matrix and Tensor Factorizations - Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley (cited on page 21).

[55] Gabriella Contardo, Ludovic Denoyer and Thierry Artières. 2014. Representation learning for cold-start recommendation. In *Proceedings of the 3rd International Conference on Learning Representations* (ICLR '14) (cited on page 59).

[56] Paul Covington, Jay Adams and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (RecSys '16), 191–198 (cited on page 88).

[57] Paolo Cremonesi, Yehuda Koren and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 2010 ACM Conference on Recommender Systems* (RecSys '10), 39–46 (cited on pages 13, 25, 26, 55, 106).

[58] Van Dang. 2018. The lemur project-wiki-ranklib. lemur project. `https://sourceforge.net/projects/lemur/`. (2018) (cited on page 100).

[59] Abhinandan Das, Mayur Datar, Ashutosh Garg and Shyamsundar Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web* (WWW '07), 271–280 (cited on page 1).

[60] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston and Dasarathi Sampath. 2010. The youtube video recommendation system. In *Proceedings of the 4th ACM Conference on Recommender Systems* (RecSys '10), 293–296 (cited on page 88).

[61] Manuel de Buenaga Rodríguez, Manuel J. Maña López, Alberto Díaz Esteban and Pablo Gervás Gómez-Navarro. 2001. A user model based on content analysis for the intelligent personalization of a news service. In *Proceedings of the 8th International Conference on User Modeling* (UM '01), 216–218 (cited on page 59).

[62] Petros Dellaportas, Jonathan J. Forster and Ioannis Ntzoufras. 2002. On bayesian model and variable selection using mcmc. *Statistics and Computing*, 12, 1, 27–36 (cited on page 112).

[63] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.*, 22, 1, 143–177 (cited on pages 1, 13, 24–26, 55, 60, 106).

[64] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '14), 193–202 (cited on page 133).

[65] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin and Jiajie Yu. 2018. Improving implicit recommender systems with view data. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (IJCAI '18), 3343–3349 (cited on page 57).

[66] X. Du, H. Yin, L. Chen, Y. Wang, Y. Yang and X. Zhou. 2018. Personalized video recommendation using rich contents from videos. *IEEE Trans. Knowl. Data Eng.* (cited on page 3).

[67] John Duchi, Elad Hazan and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159 (cited on page 100).

[68] Travis Ebesu and Yi Fang. 2017. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '17), 1093–1096 (cited on page 106).

[69] Travis Ebesu and Yi Fang. 2017. Neural semantic personalized ranking for item cold-start recommendation. *Inf. Retr. Journal*, 20, 2, 109–131 (cited on pages 73–77).

[70] Michael D Ekstrand, Praveen Kannan, James A Stemper, John T Butler, Joseph A Konstan and John T Riedl. 2010. Automatically building research reading lists. In *Proceedings of the 4th ACM Conference on Recommender Systems* (RecSys '10), 159–166 (cited on pages 105, 106).

[71] Asmaa Elbadrawy and George Karypis. 2015. User-specific feature-based similarity models for top-*n* recommendation of new items. *ACM Trans. Intel. Syst. Tech.*, 6, 3, 33 (cited on pages 32, 36, 50, 56, 57, 60, 61, 71, 73–77, 99).

[72] Ky Fan. 1949. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences*, 35, 11, 652–655 (cited on page 18).

[73] Jiashi Feng, Zhouchen Lin, Huan Xu and Shuicheng Yan. 2014. Robust subspace segmentation with block-diagonal prior. In *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '14), 3818–3825 (cited on pages 14, 16).

[74] Felice Ferrara, Nirmala Pudota and Carlo Tasso. 2011. A keyphrase-based paper recommender system. In *Proceedings of the 7th Italian Research Conference on Digital Libraries and Archives* (IRCDL '11), 14–25 (cited on page 105).

[75] Cédric Févotte and Simon J. Godsill. 2006. Sparse linear regression in unions of bases via bayesian variable selection. *IEEE Signal Processing Letters*, 13, 7, 441–444 (cited on page 111).

[76] Rana Forsati, Mehrdad Mahdavi, Mehrnoush Shamsfard and Mohamed Sarwat. 2014. Matrix factorization with explicit trust and distrust side information for improved social recommendation. *ACM Trans. Inf. Syst.*, 32, 4, 17:1–17:38 (cited on page 59).

[77] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 10th IEEE International Conference on Data Mining* (ICDM '10), 176–185 (cited on pages 32, 56, 60).

[78] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *Proceedings of the 35th IEEE International Conference on Data Engineering* (ICDE '19), 1554–1557 (cited on page 59).

[79] Huiji Gao, Jiliang Tang, Xia Hu and Huan Liu. 2015. Content-aware point of interest recommendation on location-based social networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (AAAI '15), 1721–1727 (cited on page 1).

[80] Xue Geng, Hanwang Zhang, Jingwen Bian and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In *Proceedings of IEEE International Conference on Computer Vision* (ICCV '15), 4274–4282 (cited on page 24).

[81] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the 5th IEEE International Conference on Data Mining* (ICDM '05), 625–628 (cited on pages 16, 61).

[82] Kostadin Georgiev and Preslav Nakov. 2013. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *Proceedings of the 30th International Conference on Machine Learning* (ICML '13), 1148–1156 (cited on page 60).

[83] Xavier Glorot, Antoine Bordes and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (AISTATS '2011), 315–323 (cited on page 73).

[84] David Goldberg, David A. Nichols, Brian M. Oki and Douglas B. Terry. 1992. Using collaborative filtering to weave an information tapestry. *ACM Commun.*, 35, 12, 61–70 (cited on page 1).

[85] Carlos A. Gomez-Uribe and Neil Hunt. 2016. The netflix recommender system: algorithms, business value, and innovation. *ACM Trans. Management Inf. Syst.*, 6, 4, 13:1–13:19 (cited on page 1).

[86] Google Scholar. 2018. `https://scholar.google.com/`. Last accessed April 26, 2018. (2018) (cited on pages 87–89, 104).

[87] Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng and Tat-Seng Chua. 2019. Attentive aspect modeling for review-aware recommendation. *ACM Trans. Inf. Syst.*, 37, 3, 28:1–28:27 (cited on page 61).

[88] Guibing Guo, Jie Zhang, Zhu Sun and Neil Yorke-Smith. 2015. Librec: A java library for recommender systems. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization* (UMAP '15) (cited on page 25).

[89] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: history and context. *ACM Trans. Inter. Intel. Syst.*, 5, 4, 19:1–19:19 (cited on page 119).

[90] Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra and C Lee Giles. 2011. Citation recommendation without author supervision. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining* (WSDM '11), 755–764 (cited on page 106).

[91] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (AAAI '16), 144–150 (cited on page 3).

[92] Xiangnan He, Hanwang Zhang, Min-Yen Kan and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (SIGIR '16), 549–558 (cited on page 25).

[93] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang and Tat-Seng Chua. 2018. NAIS: neural attentive item similarity model for recommendation. *IEEE Trans. Knowl. Data Eng.*, 30, 12, 2354–2366 (cited on pages 15, 23–28).

[94] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* (WWW '17), 173–182 (cited on pages 39, 50, 63).

[95] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Devel-*

opment in Information Retrieval (SIGIR '17), 355–364 (cited on pages 63, 101, 118, 120, 121, 126).

[96] Xiaofei He and Partha Niyogi. 2003. Locality preserving projections. In *Proceedings of the 17th Neural Information Processing Systems* (NIPS '03), 153–160 (cited on pages 4, 32).

[97] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*. arXiv: 1207.0580 (cited on page 22).

[98] Maya Hristakeva, Daniel Kershaw, Marco Rossetti, Petr Knoth, Benjamin Pettit, Saúl Vargas and Kris Jack. 2017. Building recommender systems for scholarly information. In *Proceedings of the 1st Workshop on Scholarly Web Mining* (SWM@WSDM '17), 25–32 (cited on page 105).

[99] Binbin Hu, Chuan Shi, Wayne Xin Zhao and Philip S. Yu. 2018. Leveraging meta-path based context for top-N recommendation with A neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (SIGKDD '18), 1531–1540 (cited on page 16).

[100] Yifan Hu, Yehuda Koren and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (ICDM '08), 263–272 (cited on pages 25, 26, 57).

[101] Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra and C Lee Giles. 2015. A neural probabilistic model for context based citation recommendation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (AAAI '15), 2404–2410 (cited on page 106).

[102] Mohsen Jamali and Laks V. S. Lakshmanan. 2013. Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the 22nd International Conference on World Wide Web* (WWW '13), 643–654 (cited on page 59).

[103] Yichen Jiang, Aixia Jia, Yansong Feng and Dongyan Zhao. 2012. Recommending academic papers via users 'reading purposes. In *Proceedings of the 6th ACM Conference on Recommender Systems* (RecSys '12), 241–244 (cited on page 105).

[104] Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '06), 217–226 (cited on page 100).

[105] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems* (RecSys '16), 43–50 (cited on page 126).

[106] Yuchin Juan, Damien Lefortier and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion* (WWW '17), 680–688 (cited on page 109).

[107] Santosh Kabbur, Xia Ning and George Karypis. 2013. FISM: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '13), 659–667 (cited on pages 13, 15, 23–27, 41, 55, 60, 63, 106, 119).

[108] Wang-Cheng Kang, Mengting Wan and Julian McAuley. 2018. Recommendation through mixtures of heterogeneous item relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (CIKM '18), 1143–1152 (cited on pages 2, 5).

[109] Zhao Kang, Chong Peng, Ming Yang and Qiang Cheng. 2016. Top-n recommendation on graphs. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management* (CIKM '16), 2101–2106 (cited on page 16).

[110] Zhao Kang and Qiang Cheng. 2016. Top-n recommendation with novel rank approximation. In *Proceedings of the 2016 SIAM International Conference on Data Mining* (SDM '16), 126–134 (cited on pages 15, 24, 106, 107).

[111] George Karypis. 2001. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management* (CIKM '01), 247–254 (cited on page 120).

[112] Hao-Ren Ke, Rolf Kwakkelaar, Yu-Min Tai and Li-Chun Chen. 2002. Exploring behavior of e-journal users in science and technology: transaction log analysis of Elsevier's ScienceDirect OnSite in Taiwan. *Library & Information Science Research*, 24, 3, 265–291 (cited on page 104).

[113] Madian Khabsa, Zhaohui Wu and C Lee Giles. 2016. Towards better understanding of academic search. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries* (JCDL '16), 111–114 (cited on page 104).

[114] Taraneh Khazaei and Orland Hoeber. 2017. Supporting academic search tasks through citation visualization and exploration. *International Journal on Digital Libraries*, 18, 1, 59–72 (cited on page 105).

[115] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations* (ICLR '15) (cited on pages 91, 121).

[116] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations* (ICLR '14) (cited on pages 39, 45, 47, 56, 62, 63, 66, 67, 111, 117).

[117] Peter Knees and Markus Schedl. 2015. Music retrieval and recommendation: A tutorial overview. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '15), 1133–1136 (cited on page 1).

[118] Noam Koenigstein and Ulrich Paquet. 2013. Xbox movies recommendations: variational bayes matrix factorization with embedded feature selection. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys '13), 129–136 (cited on page 126).

[119] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '08), 426–434 (cited on pages 15, 22, 58, 109, 120, 126).

[120] Mark A Kramer. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37, 2, 233–243 (cited on page 42).

[121] Onur Küçüktunç, Erik Saule, Kamer Kaya and Ümit V Çatalyürek. 2012. Recommendation on academic networks using direction aware citation analysis. *CoRR*. arXiv: 1205.1143 (cited on page 105).

[122] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *Proceedings of the 14th Neural Information Processing Systems* (NIPS '00), 556–562 (cited on page 21).

[123] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer and Samy Bengio. 2016. LLORMA: local low-rank matrix approximation. *Journal of Machine Learning Research*, 17, 15:1–15:24 (cited on pages 16, 61).

[124] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon and Yoram Singer. 2014. Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web* (WWW '14), 85–96 (cited on pages 16, 61).

[125] Wonsung Lee, Kyungwoo Song and Il-Chul Moon. 2017. Augmented variational autoencoders for collaborative filtering with auxiliary information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (CIKM '17), 1139–1148 (cited on pages 39, 43).

[126] Damien Lefortier, Pavel Serdyukov and Maarten de Rijke. 2014. Online exploration for detecting shifts in fresh intent. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (CIKM '14), 589–598 (cited on page 107).

[127] Huajing Li, Isaac Councill, Wang-Chien Lee and C Lee Giles. 2006. Citeseerx: an architecture and web service design for an academic document search engine. In *Proceedings of the 15th international conference on World Wide Web* (WWW '06), 883–884 (cited on pages 89, 104).

[128] Huayu Li, Yong Ge, Richang Hong and Hengshu Zhu. 2016. Point-of-interest recommendations: learning potential check-ins from friends. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '16), 975–984 (cited on page 1).

[129] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 26th ACM on Conference on Information and Knowledge Management* (CIKM '17), 1419–1428 (cited on page 109).

[130] Jundong Li, Liang Wu, Harsh Dani and Huan Liu. 2018. Unsupervised personalized feature selection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI '18), 3514–3521 (cited on page 126).

[131] Lei Li and Tao Li. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining* (WSDM '13), 305–314 (cited on page 1).

[132] Lei Li, Dingding Wang, Tao Li, Daniel Knox and Balaji Padmanabhan. 2011. SCENE: a scalable two-stage personalized news recommendation system. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '11), 125–134 (cited on page 1).

[133] Sheng Li, Jaya Kawale and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management* (CIKM '15), 811–820 (cited on pages 50, 56).

[134] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '17), 305–314 (cited on pages 4, 39, 47, 50, 57, 60, 64, 73, 74).

[135] Xinyi Li and Maarten de Rijke. 2017. Academic search in response to major scientific events. In *Proceedings of the 5th Workshop on Bibliometric-enhanced Information Retrieval (BIR) co-located with the 39th European Conference on Information Retrieval* (BIR@ECIR '17), 41–50 (cited on page 104).

[136] Xinyi Li and Maarten de Rijke. 2019. Characterizing and predicting downloads in academic search. *Inf. Process. Manage.*, 56, 3, 394–407 (cited on page 104).

[137] Xinyi Li and Maarten de Rijke. 2017. Do topic shift and query reformulation patterns correlate in academic search? In *Proceedings of the 39th European Conference on IR Research* (ECIR '17), 146–159 (cited on pages 94, 104).

[138] Xinyi Li, Bob J.A. Schijvenaars and Maarten de Rijke. 2017. Investigating queries and search failures in academic search. *Inf. Process. Manage.*, 53, 3, 666–683 (cited on page 104).

[139] Xinyi Li, Yifan Chen, Benjamin Pettit and Maarten de Rijke. 2019. Personalised reranking of paper recommendations using paper content and user behavior. *ACM Trans. Inf. Syst.*, 37, 3, Article 31 (cited on pages 8, 87).

[140] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen and Yong Rui. 2014. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '14), 831–840 (cited on page 1).

[141] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 27th International Conference on World Wide Web* (WWW '18), 689–698 (cited on pages 4, 15, 25, 26, 39, 46, 47, 111, 114, 120).

[142] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan and Tat-Seng Chua. 2013. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '13), 283–292 (cited on page 56).

[143] Xiao Lin, Wenpeng Zhang, Min Zhang, Wenwu Zhu, Jian Pei, Peilin Zhao and Junzhou Huang. 2018. Online compact convexified factorization machine. In *Proceedings of the 27th International Conference on World Wide Web* (WWW '18), 1633–1642 (cited on page 126).

[144] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (AAAI '15), 2181–2187 (cited on page 93).

[145] Greg Linden, Brent Smith and Jeremy York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7, 1, 76–80 (cited on page 90).

[146] Greg Linden, Brent Smith and Jeremy York. 2003. Industry report: amazon.com recommendations: item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 4, 1 (cited on page 1).

[147] Bin Liu, Yanjie Fu, Zijun Yao and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGIR '13), 1043–1051 (cited on page 1).

[148] Haifeng Liu, Xiangjie Kong, Xiaomei Bai, Wei Wang, Teshome Megersa Bekele and Feng Xia. 2015. Context-based collaborative filtering for citation recommendation. *IEEE Access*, 3, 1695–1703 (cited on page 106).

[149] Nathan Nan Liu, Xiangrui Meng, Chao Liu and Qiang Yang. 2011. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *Proceedings of the 2011 ACM Conference on Recommender Systems* (RecSys '11), 37–44 (cited on page 59).

[150] C. Lu, J. Feng, Z. Lin, T. Mei and S. Yan. 2019. Subspace clustering by block diagonal representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41, 2, 487–501 (cited on pages 14, 16).

[151] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang and Shuicheng Yan. 2012. Robust and efficient subspace segmentation via least squares regression. In *Proceedings of the 12th European Conference on Computer Vision* (ECCV '12), 347–360 (cited on page 14).

[152] Chun-Ta Lu, Lifang He, Weixiang Shao, Bokai Cao and Philip S. Yu. 2017. Multilinear factorization machines for multi-task multi-view learning. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining* (WSDM '17), 701–709 (cited on pages 109, 126).

[153] Jin Lu, Guannan Liang, Jiangwen Sun and Jinbo Bi. 2016. A sparse interactive model for matrix completion with side information. In *Proceedings of the 30th Neural Information Processing Systems* (NIPS '16), 4071–4079 (cited on page 32).

[154] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining* (WSDM '11), 287–296 (cited on page 98).

[155] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 2579–2605 (cited on page 81).

[156] X. Mao, S. Mitra and V. Swaminathan. 2017. Feature selection for fm-based context-aware recommendation systems. In *Proceedings of the 19th IEEE International Symposium on Multimedia* (ISM '17), 252–255 (cited on pages 110, 126).

[157] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rrating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys '13), 165–172 (cited on pages 1, 23, 46, 62, 70).

[158] Peter McCullagh and John A Nelder. 1989. *Generalized Linear Models*. Volume 37. CRC press (cited on page 115).

[159] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma and Jian-Yun Nie. 2018. An attentive interaction network for context-aware recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (CIKM '18), 157–166 (cited on page 126).

[160] Prem Melville, Raymond J. Mooney and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence* (AAAI '02), 187–192 (cited on page 59).

[161] Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML PKDD '11), 437–452 (cited on page 1).

[162] Toby J. Mitchell and John J. Beauchamp. 1988. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83, 404, 1023–1032 (cited on page 111).

[163] Anasua Mitra and Amit Awekar. 2017. On low overlap among search results of academic search engines. In *Proceedings of the 26th International Conference on World Wide Web Companion* (WWW '17), 823–824 (cited on page 104).

[164] Bojan Mohar. 1991. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*. Volume 2. Wiley, 871–898 (cited on page 17).

[165] Taesup Moon, Wei Chu, Lihong Li, Zhaohui Zheng and Yi Chang. 2012. An online learning framework for refining recency search results with user click feedback. *ACM Trans. Inf. Syst.*, 30, 4, 20:1–20:28 (cited on page 107).

[166] Cristiano Nascimento, Alberto HF Laender, Altigran S da Silva and Marcos André Gonçalves. 2011. A source independent framework for research paper recommendation. In *Proceedings of the 2011 Joint International Conference on Digital Libraries* (JCDL '11), 297–306 (cited on page 105).

[167] Trung V. Nguyen, Alexandros Karatzoglou and Linas Baltrunas. 2014. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '14), 63–72 (cited on page 126).

[168] Feiping Nie, Wei Zhu and Xuelong Li. 2016. Unsupervised feature selection with structured graph optimization. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (AAAI '16), 1302–1308 (cited on page 20).

[169] Xia Ning and George Karypis. 2011. SLIM: sparse linear methods for top-n recommender systems. In *Proceedings of the 11th IEEE International Conference on Data Mining* (ICDM '11), 497–506 (cited on pages 3, 13, 15, 18, 24, 26, 27, 31–33, 41, 43, 50, 55, 60, 106, 120).

[170] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the 6th ACM Conference on Recommender Systems* (RecSys '12), 155–162 (cited on pages 1, 3, 16, 31, 32, 36, 39, 41, 47, 50, 130).

[171] Xi Niu and Bradley M Hemminger. 2012. A study of factors that affect the information-seeking behavior of academic scientists. *Journal of the American Society for Information Science and Technology*, 63, 2, 336–353 (cited on page 104).

[172] Uros Ocepek, Joze Rugelj and Zoran Bosnic. 2015. Improving matrix factorization recommendations for examples in cold start. *Expert Syst. Appl.*, 42, 19, 6784–6794 (cited on page 59).

[173] Mark O'Connor and Jon Herlocker. 1999. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*. Volume 128 (cited on page 16).

[174] John William Paisley, David M. Blei and Michael I. Jordan. 2012. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning* (ICML '12) (cited on page 45).

[175] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining* (ICDM '08), 502–511 (cited on page 57).

[176] Zhen Pan, Enhong Chen, Qi Liu, Tong Xu, Haiping Ma and Hongjie Lin. 2016. Sparse factorization machines for click-through rate prediction. In *Proceedings of the 16th IEEE International Conference on Data Mining* (ICDM '16), 400–409 (cited on pages 101, 110, 120, 126).

[177] Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *Proceedings of the 3rd ACM Conference on Recommender Systems* (RecSys '09), 21–28 (cited on page 58).

[178] Rajiv Pasricha and Julian McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems* (RecSys '18), 63–71 (cited on page 126).

[179] David M Pennock, Eric Horvitz, Steve Lawrence and C Lee Giles. 2000. Collaborative filtering by personality diagnosis: a hybrid memory-and model-based approach. In *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence* (UAI '00), 473–480 (cited on page 105).

[180] Stefan Pero and Tomás Horváth. 2013. Opinion-driven matrix factorization for rating prediction. In *Proceedings of the 21th International Conference on User Modeling, Adaptation, and Personalization* (UMAP '13), 1–13 (cited on page 62).

[181] Sheila Pontis, Ann Blandford, Elke Greifeneder, Hesham Attalla and David Neal. 2015. Keeping up to date: an academic researcher's information journey. *Journal of the American Society for Information Science and Technology*, 68, 1, 22–35 (cited on page 104).

[182] Sheila Pontis and Ann Blandford. 2015. Understanding "influence:" an exploratory study of academics' processes of knowledge construction through iterative and interactive information seeking. *Journal of the Association for Information Science and Technology*, 66, 8, 1576–1593 (cited on page 104).

[183] Alexandrin Popescul, Lyle H. Ungar, David M. Pennock and Steve Lawrence. 2001. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence* (UAI '01), 437–444 (cited on page 59).

[184] Ian Porteous, Arthur U. Asuncion and Max Welling. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (AAAI '10) (cited on page 59).

[185] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining* (ICDM '10), 995–1000 (cited on pages 59, 109, 120, 126).

[186] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Trans. Intell. Syst. & Tech.*, 3, 3, 57:1–57:22 (cited on pages 101, 109).

[187] Steffen Rendle. 2012. Learning recommender systems with adaptive regularization. In *Proceedings of the 15th International Conference on Web Search and Web Data Mining* (WSDM '12), 133–142 (cited on page 109).

[188] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. 2009. BPR: bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence* (UAI '09), 452–461 (cited on pages 24–26, 98, 114).

[189] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '11), 635–644 (cited on pages 109, 126).

[190] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining* (WSDM '10), 81–90 (cited on page 126).

[191] Bradley J. Rhodes and Pattie Maes. 2000. Just-in-time information retrieval agents. *IBM Systems Journal*, 39, 3&4, 685–704 (cited on page 1).

[192] Francesco Ricci, Lior Rokach and Bracha Shapira, editors. 2015. *Recommender Systems Handbook*. Springer (cited on pages 13, 32, 106).

[193] Royi Ronen, Noam Koenigstein, Elad Ziklik and Nir Nice. 2013. Selecting content-based features for collaborative filtering recommenders. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys '13), 407–410 (cited on page 126).

[194] Gerard Salton and Michael McGill. 1984. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company (cited on page 1).

[195] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl. 2000. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce* (EC '00), 158–167 (cited on page 88).

[196] Badrul M Sarwar, George Karypis, Joseph Konstan and John Riedl. 2002. Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering. In *Proceedings of the 5th International Conference on Computer and Information Technology* (ICCIT '02), 291–324 (cited on page 3).

[197] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference* (WWW '01), 285–295 (cited on pages 13, 55, 90).

[198] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems* (RecSys '14), 89–96 (cited on pages 22, 32, 56, 59, 73–77, 98).

[199] J. Ben Schafer, Dan Frankowski, Jonathan L. Herlocker and Shilad Sen. 2007. Collaborative filtering recommender systems. In *Proceedings of the Adaptive Web, Methods and Strategies of Web Personalization*, 291–324 (cited on page 1).

[200] Amir Schorr. 1982. Fast algorithm for sparse matrix multiplication. *Inf. Process. Lett.*, 15, 2, 87–89 (cited on page 21).

[201] ScienceDirect. 2015. `https://sciencedirect.com`. Last accessed September 14, 2015. (2015) (cited on pages 87, 88, 104).

[202] ScienceDirect. 2016. `https://www.elsevier.com/solutions/sciencedirect/features`. Last accessed April 26, 2018. (2016) (cited on page 88).

[203] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner and Lexing Xie. 2015. Autorec: autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion* (WWW '15), 111–112 (cited on pages 39, 42).

[204] Semantic Scholar. 2018. `https://www.semanticscholar.org/`. Last accessed December 1, 2018. (2018) (cited on page 89).

[205] Aravind Sesagiri Raamkumar, Schubert Foo and Natalie Pang. 2018. Can i have more of these please? assisting researchers in finding similar research papers from a seed basket of papers. *The Electronic Library* (cited on page 105).

[206] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu and J. C. Mao. 2016. Deep crossing: web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '16), 255–262 (cited on pages 63, 109).

[207] Mohit Sharma, Jiayu Zhou, Junling Hu and George Karypis. 2015. Feature-based factorized bilinear similarity model for cold-start top-*n* item recommendation. In *Proceedings of the 2015 SIAM International Conference on Data Mining* (SDM '15), 190–198 (cited on pages 50, 56, 57, 60, 61, 63, 71, 73–77).

[208] Lei Shi, Wayne Xin Zhao and Yi-Dong Shen. 2017. Local representative-based matrix factorization for cold-start recommendation. *ACM Trans. Inf. Syst.*, 36, 2, 22:1–22:28 (cited on pages 56, 58).

[209] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '08), 650–658 (cited on page 59).

[210] Ian Soboroff and Charles Nicholas. 1999. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*. Volume 99, 86–91 (cited on page 59).

[211] Guocong Song. 2014. Point-wise approach for yandex personalized web search challenge. In *Proceedings of the WSDM Workshop on Web Search Click Data* (WSCD@WSDM '14) (cited on page 100).

[212] Nathan Srebro, Jason D. M. Rennie and Tommi S. Jaakkola. 2004. Maximum-margin matrix factorization. In *Proceedings of the 18th Neural Information Processing Systems* (NIPS '04), 1329–1336 (cited on page 126).

[213] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1, 1929–1958 (cited on pages 22, 133).

[214] Ingo Steinwart and Andreas Christmann. 2008. *Support Vector Machines*. Springer Science & Business Media (cited on page 109).

[215] Trevor Strohman, W Bruce Croft and David Jensen. 2007. Recommending citations for academic papers. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '07), 705–706 (cited on page 106).

[216] Florian Strub, Romaric Gaudel and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (DLRS@RecSys '16), 11–16 (cited on page 42).

[217] Kazunari Sugiyama and Min-Yen Kan. 2010. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th Joint International Conference on Digital Libraries* (JCDL '10), 29–38 (cited on page 105).

[218] Yunzhi Tan, Min Zhang, Yiqun Liu and Shaoping Ma. 2016. Rating-boosted latent topics: understanding users and items with ratings and reviews. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (IJCAI '16), 2640–2646 (cited on page 62).

[219] Jie Tang. 2016. Aminer: toward understanding big scholar data. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining* (WSDM '16), 467–467 (cited on pages 87, 104).

[220] Jie Tang, Ruoming Jin and Jing Zhang. 2008. A topic modeling approach and its integration into the random walk framework for academic search. In *Proceedings of the 8th IEEE International Conference on Data Mining* (ICDM '08), 1055–1060 (cited on page 105).

[221] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 267–288 (cited on pages 110–112).

[222] Michalis K. Titsias and Miguel Lázaro-Gredilla. 2011. Spike and slab variational inference for multi-task and multiple kernel learning. In *Proceedings of the 25th Neural Information Processing Systems* (NIPS '11), 2339–2347 (cited on pages 111, 113).

[223] Seiya Tokui and Issei Sato. 2016. Reparameterization trick for discrete variables. *CoRR*. arXiv: 1611.01239 (cited on page 117).

[224] Roberto Torres, Sean M McNee, Mara Abel, Joseph A Konstan and John Riedl. 2004. Enhancing digital libraries with techlens+. In *Proceedings of the 4th ACM/IEEE Joint Conference on Digital Libraries* (JCDL '04), 228–236 (cited on pages 105, 106).

[225] Michele Trevisiol, Luca Maria Aiello, Rossano Schifanella and Alejandro Jaimes. 2014. Cold-start news recommendation with domain-dependent browse graph. In *Proceedings of the 8th ACM Conference on Recommender Systems* (RecSys '14), 81–88 (cited on page 56).

[226] Aäron van den Oord, Sander Dieleman and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Proceedings of the 27th Neural Information Processing Systems* (NIPS '13), 2643–2651 (cited on page 60).

[227] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11, 3371–3408 (cited on page 56).

[228] Maksims Volkovs and Guang Wei Yu. 2015. Effective latent models for binary feedback in recommender systems. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '15), 313–322 (cited on page 57).

[229] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '11), 448–456 (cited on pages 1, 3, 60, 70, 105, 133).

[230] Hao Wang, Naiyan Wang and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '15), 1235–1244 (cited on pages 3, 50, 56, 57, 60, 64).

[231] Hao Wang, Binyi Chen and Wu-Jun Li. 2013. Collaborative topic regression with social regularization for tag recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* (IJCAI '13), 2719–2725 (cited on page 70).

[232] Keqiang Wang, Wayne Xin Zhao, Hongwei Peng and Xiaoling Wang. 2016. Bayesian probabilistic multi-topic matrix factorization for rating prediction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (IJCAI '16), 3910–3916 (cited on pages 16, 61).

[233] Meng Wang, Xueliang Liu and Xindong Wu. 2015. Visual classification by ℓ_1-hypergraph modeling. *IEEE Trans. Knowl. Data Eng.*, 27, 9, 2564–2574 (cited on page 110).

[234] Xinxi Wang and Ye Wang. 2014. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia* (MM '14), 627–636 (cited on page 60).

[235] Yang Wang, Xuemin Lin, Lin Wu and Wenjie Zhang. 2017. Effective multi-query expansions: collaborative deep networks for robust landmark retrieval. *IEEE Trans. Image Processing*, 26, 3, 1393–1404 (cited on page 59).

[236] Zengmao Wang, Yuhong Guo and Bo Du. 2018. Matrix completion with preference ranking for top-n recommendation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (IJCAI '18), 3585–3591 (cited on page 16).

[237] Mike West. 2003. Bayesian factor regression models in the "large p, small n" paradigm. In *Bayesian Statistics*. Oxford University Press, 723–732 (cited on page 112).

[238] Yao Wu, Xudong Liu, Min Xie, Martin Ester and Qing Yang. 2016. CCCF: improving collaborative filtering via scalable user-item co-clustering. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining* (WSDM '16), 73–82 (cited on pages 14, 16, 61).

[239] Yao Wu, Christopher DuBois, Alice X. Zheng and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining* (WSDM '16), 153–162 (cited on pages 4, 15, 39, 42, 48, 50, 107, 119).

[240] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu and Tat-Seng Chua. 2017. Attentional factorization machines: learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (IJCAI '17), 3119–3125 (cited on pages 118, 120, 121, 126).

[241] Zhibo Xiao, Feng Che, Enuo Miao and Mingyu Lu. 2014. Increasing serendipity of recommender system with ranking topic model. *Applied Mathematics & Information Sciences*, 8, 4, 2041 (cited on page 105).

[242] Junyuan Xie, Ross B. Girshick and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33nd International Conference on Machine Learning* (ICML '16). Volume 48, 478–487 (cited on page 56).

[243] Xingyu Xie, Xianglin Guo, Guangcan Liu and Jun Wang. 2018. Implicit block diagonal low-rank representation. *IEEE Trans. Image Processing*, 27, 1, 477–489 (cited on page 16).

[244] Eric P Xing, Michael I Jordan and Stuart Russell. 2002. A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence* (UAI '02), 583–591 (cited on page 44).

[245] Chenyan Xiong, Russell Power and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web* (WWW '17), 1271–1279 (cited on page 105).

[246] Bin Xu, Jiajun Bu, Chun Chen and Deng Cai. 2012. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st International Conference on World Wide Web* (WWW '12), 21–30 (cited on pages 14, 16, 61).

[247] Jianpeng Xu, Kaixiang Lin, Pang-Ning Tan and Jiayu Zhou. 2016. Synergies that matter: efficient interaction selection via sparse factorization machine. In *Proceedings of the 2016 SIAM International Conference on Data Mining* (SDM '16), 108–116 (cited on pages 110, 120, 126).

[248] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Trans. Inf. Syst.*, 37, 3, 33:1–33:25 (cited on pages 15, 23).

[249] Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu and Zheng Chen. 2005. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '05), 114–121 (cited on pages 14, 16).

[250] Makoto Yamada, Wenzhao Lian, Amit Goyal, Jianhui Chen, Kishan Wimalawarne, Suleiman A. Khan, Samuel Kaski, Hiroshi Mamitsuka and Yi Chang. 2017. Convex factorization machine for toxicogenomics prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '17), 1215–1224 (cited on page 109).

[251] Chunfeng Yang, Huan Yan, Donghan Yu, Yong Li and Dah Ming Chiu. 2017. Multi-site user behavior modeling and its application in video recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '17), 175–184 (cited on page 61).

[252] Diyi Yang, Tianqi Chen, Weinan Zhang, Qiuxia Lu and Yong Yu. 2012. Local implicit feedback mining for music recommendation. In *Proceedings of 6th ACM Conference on Recommender Systems* (RecSys '12), 91–98 (cited on page 1).

[253] Liu Yang, Qi Guo, Yang Song, Sha Meng, Milad Shokouhi, Kieran McDonald and W. Bruce Croft. 2016. Modeling user interests for zero-query ranking. In *Proceedings of the 38th European Conference on IR Research* (ECIR '16), 171–184 (cited on page 1).

[254] Xitong Yang. 2017. Understanding the variational lower bound. `https://xyang35.github.io/2017/04/14/variational-lower-bound/`. (2017) (cited on page 65).

[255] Hilmi Yildirim and Mukkai S. Krishnamoorthy. 2008. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2nd ACM Conference on Recommender Systems* (RecSys '08), 131–138 (cited on page 14).

[256] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu and Ling Chen. 2014. LCARS: A spatial item recommender system. *ACM Trans. Inf. Syst.*, 32, 3, 11:1–11:37 (cited on page 59).

[257] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys '13), 347–350 (cited on page 1).

[258] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu and Weinan Zhang. 2017. Boostfm: boosted factorization machines for top-n feature-based recommendation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (IUI '17), 45–54 (cited on pages 109, 114, 126).

[259] Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68, 1, 49–67 (cited on page 126).

[260] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun and Nadia Magnenat-Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '13), 363–372 (cited on page 1).

[261] Jun Zhang, Chaokun Wang, Philip S. Yu and Jianmin Wang. 2013. Learning latent friendship propagation networks with interest awareness for link prediction. In *Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval* (SIGIR '13), 63–72 (cited on page 1).

[262] Liang Zhang, Deepak Agarwal and Bee-Chung Chen. 2011. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the 2011 ACM Conference on Recommender Systems* (RecSys '11), 13–20 (cited on page 60).

[263] Shuai Zhang, Lina Yao and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52, 1, 5:1–5:38 (cited on page 50).

[264] Weinan Zhang, Tianming Du and Jun Wang. 2016. Deep learning over multi-field categorical data – A case study on user response prediction. In *Proceedings of the 38th European Conference on IR Research* (ECIR '16), 45–57 (cited on page 63).

[265] Yi Zhang and Jonathan Koren. 2007. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '07), 47–54 (cited on page 59).

[266] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '14), 83–92 (cited on page 62).

[267] Yongfeng Zhang, Qingyao Ai, Xu Chen and W. Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 26th*

ACM on Conference on Information and Knowledge Management (CIKM '17), 1449–1458 (cited on pages 2, 5).

[268] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma and Shi Feng. 2013. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22nd International Conference on World Wide Web* (WWW '13), 1511–1520 (cited on pages 14, 16, 61).

[269] Feipeng Zhao and Yuhong Guo. 2016. Improving top-n recommendation with heterogeneous loss. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (IJCAI '16), 2378–2384 (cited on pages 16, 107).

[270] Feipeng Zhao, Min Xiao and Yuhong Guo. 2016. Predictive collaborative filtering with side information. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (IJCAI '16), 2385–2391 (cited on pages 32, 36, 41, 50).

[271] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD '17), 635–644 (cited on pages 110, 120, 126).

[272] Lei Zheng, Vahid Noroozi and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining* (WSDM '17), 425–434 (cited on pages 61, 62).

[273] Yin Zheng, Bangsheng Tang, Wenkui Ding and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33nd International Conference on Machine Learning* (ICML '16). Volume 48, 764–773 (cited on page 39).

[274] Yong Zheng, Robin D. Burke and Bamshad Mobasher. 2013. Recommendation with differential context weighting. In *Proceedings of the 21th International Conference on User Modeling, Adaptation, and Personalization* (UMAP '13), 152–164 (cited on page 126).

[275] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee and Guillermo Sapiro. 2012. Kernelized probabilistic matrix factorization: exploiting graphs and side information. In *Proceedings of the 12th SIAM International Conference on Data Mining* (SDM '12), 403–414 (cited on page 59).

[276] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie and Qing He. 2017. Representation learning via dual-autoencoder for recommendation. *Neural Networks*, 90, 83–89 (cited on pages 42, 50).

[277] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web* (WWW '05), 22–32 (cited on page 23).

[278] Masrour Zoghi, Tomáš Tunys, Lihong Li, Damien Jose, Junyan Chen, Chun Ming Chin and Maarten de Rijke. 2016. Click-based hot fixes for underperforming torso queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '16), 195–204 (cited on page 107).

[279] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67, 2, 301–320 (cited on page 18).

[280] Jie Zou, Yifan Chen and Evangelos Kanoulas. 2020. Towards question-based recommender systems. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining* (WSDM '20). Under review. (cited on page 9).

Summary

Top-N recommendations have been widely adopted to recommend ranked lists of items so as to help users identify the items that best fit their personal tastes. Collaborative filtering (CF) has been widely studied to generate recommendations by utilizing the information of user's historical interactions with items (a.k.a. ratings). However, due to the *high-dimensionality* of ratings in practical applications with a large number of users and items, existing CF-based methods are facing severe challenges in terms of scalability. The sparsity of ratings caused by the high-dimensionality further challenges the performance of recommendation. Typically, additional auxiliary information associated with users or items (a.k.a. side information) is exploited to overcome the rating sparsity. Unfortunately, in recent multimedia scenarios, such information is also high-dimensional. Besides ratings and side information, other information that is relevant for recommendation is also collected from different data sources. How to effectively integrate such *heterogeneous* information while preserving the effectiveness of CF is also a challenge. In this thesis, we research on top-N recommendations by learning from high-dimensional information and heterogeneous information, based on which we divide the thesis into two parts.

In the first part of the thesis, we focus on leveraging high-dimensional information for top-N recommendations. The first part contains three research chapters, where we respectively utilize high-dimensional ratings (Chapter 2) and high-dimensional side information (Chapter 3 and 4). In Chapter 2, we propose a new regularization term for item-based collaborative filtering (ICF) to overcome issues brought by the high-dimensionality of ratings. In Chapter 3, we propose a joint learning method that simultaneously performs dimension reduction on high-dimensional side information and estimates parameters of an ICF model. In Chapter 4, we propose a new network structure on top of variational auto-encoder to denoise and harness high-dimensional side information.

In the second part of the thesis, we focus on integrating heterogeneous information for top-N recommendations. The second part also contains three chapters, where we respectively combine ratings with item features (Chapter 5), user behavior with content features (Chapter 6) and generic heterogeneous features (Chapter 7). In Chapter 5, we study the problem of recommending top-N new items by estimating local and global similarity functions that calculate item similarities based on item features. We form a Bayesian generative model to seamlessly integrate item-based collaborative filtering with user clustering and deep learning. In Chapter 6, we work on the problem of reranking research paper recommendations by designing a hybrid reranking model. The proposed method combines information behind user multiple behaviors (click, browse, download and etc.) and paper content features (author, venue, entity and etc.). In Chapter 7, we take the effectiveness of factorization machines (FMs) to utilize heterogeneous information for top-N recommendations. We address the high-dimensionality of feature interactions by designing a Bayesian variable selection model. We propose Bayesian personalized feature interaction selection (BP-FIS) as a framework to select personalized feature interactions for FMs.

Samenvatting

Top-N-aanbevelingssystemen worden vaak toegepast om geordende lijsten van aanbevolen artikelen aan gebruikers te tonen, zodat ze artikelen kunnen vinden die het best bij hun persoonlijke smaak passen. Collaborative Filtering (CF) is een veel bestudeerde manier om aanbevelingen te genereren aan de hand van interacties met de gebruiker in het verleden (i.e., recensiescores). Echter, door de hoge dimensionaliteit van recensiescores in praktische toepassingen met een groot aantal gebruikers en artikelen, ervaren bestaande CF-methoden grote moeilijkheden met schaalbaarheid. De schaarste van recensiescores veroorzaakt door de hoge dimensionaliteit zorgt voor nog meer uitdagingen. De traditionele aanpak om met schaarste om te gaan is door extra informatie over de gebruiker en artikelen te gebruiken. Helaas heeft deze extra informatie ook een hoge dimensionaliteit in moderne multimedia-scenarios. Naast recensiescores en standaard extra informatie, is er nog veel meer relevante informatie beschikbaar van verschillende databronnen. De integratie van zulke ongelijksoortige informatie in een effectieve toepassing van CF is een grote uitdaging. In dit proefschrift onderzoeken we top-N-aanbevelingssystemen die leren van informatie met hoge dimensionaliteit en uit heterogene databronnen; aan de hand van deze twee onderwerpen is het proefschrift in twee delen verdeeld.

In het eerste deel van het proefschrift concentreren we ons op manieren waarmee informatie met een hoge dimensionaliteit kan worden gebruikt door top-N aanbevelingssystemen. Dit deel bestaat uit drie onderzoekshoofdstukken, waar we eerst recensiescores met een hoge dimensionaliteit gebruiken (hoofdstuk 2), en vervolgens gebruik maken van extra informatie met een hoge dimensionaliteit (hoofdstuk 3 en 4). In hoofdstuk 2 introduceren we een nieuwe regularisatie-term voor artikel-gebaseerd CF (ICF), als oplossing voor de problemen met de hoge dimensionaliteit van recensiescores. In hoofdstuk 3 stellen we een nieuwe gezamenlijke leermethode voor, die tegelijkertijd de dimensionaliteit van extra informatie reduceert en de parameters van een ICF-model afleidt. In hoofdstuk 4 introduceren we een nieuwe netwerkstructuur voor variational auto-encoders om ruis te verminderen en extra informatie met hoge dimensionaliteit te benutten.

In het tweede deel van het proefschrift richten we ons op het integreren van ongelijksoortige informatie voor top-N-aanbevelingssystemen. Dit tweede deel van het proefschrift is ook opgedeeld in drie hoofdstukken: Eerst kijken we naar het combineren van artikeleigenschappen met recensiescores (hoofdstuk 5). Dan naar gebruikersgedrag en artikeleigenschappen (hoofdstuk 6) en generieke heterogene informatie (hoofdstuk 7). In hoofdstuk 5 bestuderen we het probleem van nieuwe artikelen voor top-N-aanbevelingen door lokale en globale similariteitsfuncties af te leiden; hiermee kan de similariteit tussen artikelen worden afgeleid van hun eigenschappen. We vormen een Bayesiaans generatief-model dat artikel-gebaseerde CF met het clusteren van gebruikers en deep learning combineert. In hoofdstuk 6 werken we aan het herordenen van onderzoeksartikel aanbevelingen door een hybride systeem te ontwikkelen. De voorgestelde methode combineert informatie van meerdere soorten gebruikersgedrag (klikken, doorbladeren, downloaden, etc.) en artikelinformatie (auteur, conferentie, entiteiten, etc.). In hoofdstuk 7 gebruiken we de effectiviteit van Factorisatie Machines (FMs) om heterogene informatie in top-N-aanbevelingssystemen te benutten.

We pakken de hoge dimensionaliteit van eigenschap interacties aan via een nieuw Bayesiaans variabel selecteringsmodel. Verder stellen we een Bayesiaans gepersonaliseerde eigenschaps-interactie-selectering-methode (BP-FIS) voor, dit vormt een methodologie voor het selecteren van gepersonaliseerde eigenschaps-interacties voor FMs.