University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Su, Yixin

Title:
Explicit Feature Interaction Modeling for Recommender Systems

Date:
2022

Persistent Link:
https://hdl.handle.net/11343/315702

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.

Explicit Feature Interaction Modeling for Recommender Systems

by

Yixin Su

ORCID: 0000-0003-0553-9163

A thesis submitted in total fulfillment for the
degree of Doctor of Philosophy

in the
School of Computing and Information Systems
Melbourne School of Engineering
THE UNIVERSITY OF MELBOURNE

September 2022

Abstract

Recommender systems play a key role in addressing information overload issues in many Web applications, such as e-commerce, social media platforms, and lifestyle apps. For example, the Amazon shopping website lists items that a user might be interested in at the top of the website. The core function of recommender systems is to predict how likely a user will select an item (e.g., purchase, click). Typical recommender systems leverage users' historical selected items (i.e., user-item interactions) to infer users' interests. For example, matrix factorization (MF), one of the most common recommendation algorithms, learns user and item representations by factorizing the user-item interaction matrix as the multiplication of a user matrix, and an item matrix. Besides user-item interactions, studies show that features (e.g., user/item attributes, contexts), especially feature interactions (e.g., the co-occurrences of features), are important side information that can significantly enhance the recommendation accuracy.

Deep neural networks (DNNs) have achieved great success in many recent studies due to their powerful information analysis ability. As a result, recent recommendation models seek to leverage DNNs for better feature interaction modeling. However, they model feature interactions *implicitly* (i.e., model all feature interactions together in a black-box DNN model without knowing which interactions are modeled or how they are modeled). Recent studies have shown that implicit interaction learning is less effective in extracting useful information about feature interaction for accurate recommendations.

In this thesis, we explore how to effectively leverage feature interactions and improve the performance of recommender systems. Specifically, we focus on modeling feature interactions in an *explicit* manner. Unlike the implicit modeling methods, explicit methods model each feature interaction individually, allowing to choose which feature interactions to model, and decide how to model each feature interaction for more accurate recommendations. Then, we focus on three challenging research questions in improving recommender systems through explicit feature interaction modeling.

The first research question explores how to improve the performance of MF by enabling it to explicitly model feature interactions. MF learns user and item representations from user-item interactions, which has a drawback: it cannot consider feature interactions. This limits MF's potential to perform fine-grained analysis for more accurate predictions. Meanwhile, MF may encounter the cold-start problem (i.e., an item has too few historical user-item interactions to conduct an effective analysis). Focusing on this drawback, instead of factorizing the user-item interaction matrix, we propose to factorize multiple user-attribute interaction matrices to learn attribute representations. The final prediction is an aggregation of all the ratings predicted in the user-attribute matrices. Our

proposed method achieves higher accuracy than MF-based methods, while resolving the cold-start problem.

The second research question explores *which* feature interactions should be modeled in recommender systems. Existing recommendation algorithms consider all feature interactions to generate predictions. However, not all feature interactions are relevant to the recommendation prediction, and capturing irrelevant feature interactions may introduce noise and decrease the prediction accuracy. Therefore, we propose to detect a set of most relevant feature interactions (we formally define them as *beneficial feature interactions* in terms of prediction accuracy) and model only beneficial feature interactions for more accurate recommendation. We propose novel frameworks that leverage the relational reasoning ability of graph neural networks (GNNs) to achieve more effective explicit feature interaction modeling. Under these frameworks, beneficial feature interaction detection and recommendation prediction are achieved via an edge prediction task and a graph classification task, respectively.

The third research question explores *how* to model different feature interactions in recommender systems. Existing recommendation algorithms model each feature interaction equally, neglecting their different impacts while performing the prediction. We explore how feature interactions can be categorized and modeled to fit their roles in recommendation. More specifically, for user attributes and item attributes (e.g., user gender, item color), we define two types of interactions: *inner interactions* for profile learning and *cross interactions* for preference matching. We propose a neural graph matching method, which is based on our proposed GNN-based interaction modeling framework, to model the two types of interactions so that can better analyze their impacts on recommendation predictions. For context features (e.g., weather, time), inspired by psychology, we leverage them to learn *intrinsic factors* and *extrinsic factors* that jointly influence users selection. Contrastive learning and disentanglement learning algorithms are leveraged to learn these factors.

In summary, this thesis has made several contributions to explicit feature interaction modeling for improving recommender systems through: enabling feature interaction modeling in matrix factorization, detecting beneficial feature interactions, and categorizing and modeling different types of feature interactions to fit their roles in recommendation. All works are supported by theoretical analysis and empirical results.

Declaration

This is to certify that:

- the thesis comprises only my original work towards the degree of Doctor of Philosophy,
- due acknowledgement has been made in the text to all other material used,
- the thesis is fewer than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signed: Yixin Su

Date: September 2022

Preface

This thesis has been written at the School of Computing and Information Systems, The University of Melbourne. The major parts of the thesis are the Chapters 3, 4, 5, 6 and 7. They are based on published or submitted papers, and I declare that I am the primary author and have contributed $> 50\%$ in all of the following papers.

1. Yixin, Su, Sarah Erfani, and Rui Zhang. "MMF: Attribute Interpretable Collaborative Filtering". In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. 2019.

2. Yixin, Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. "Detecting Beneficial Feature Interactions for Recommender Systems". In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4357–4365. 2021.

3. Yixin Su, Yunxiang Zhao, Sarah Erfani, Junhao Gan, and Rui Zhang. "Detecting Arbitrary Order Beneficial Feature Interactions for Recommender Systems". In *Proceedings of the Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 2022.

4. Yixin, Su, Rui Zhang, Sarah Erfani, and Junhao Gan. "Neural Graph Matching based Collaborative Filtering". In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 849–858. 2021.

5. Yixin Su, Wei Jiang, Fangquan Lin, Cheng Yang, Sarah Erfani, and Junhao Gan. "Intrinsic-Extrinsic Disentangled Recommendation with Context-aware Contrastive Learning". *Submitted* to the Conference on Neural Information Processing Systems (NeurIPS) 2022.

Acknowledgements

First and foremost, I am deeply grateful to my supervisors, Dr. Sarah Erfani, Dr. Junhao Gan, and Prof. Rui Zhang, for their supervision and guidance throughout my Ph.D. study. This thesis would not have been possible without their supervision. I would also like to thank my advisory committee chair Prof. Lars Kulik. His advice helps me a lot in my research.

I would like to express my sincere gratitude to all my co-authors Zhenghua Xu, Yunxiang Zhao, Wei Jiang, Fangquan Lin, and Cheng Yang. Collaborating and communicating with them inspire me a lot in my research.

I would like to thank my knowledgeable colleagues Xiaojie Wang, Xinting Huang, Jiabo He, Shiquan Yang, Ang Li, Yimeng Dai, Zhuowei Zhao, Yujing Jiang, Lianglu Pan, Zien Wang, Yiqing Zhang, Chenxu Zhao, etc., for their insightful comments and suggestions in my research.

I would like to acknowledge the financial assistance from the Chinese Scholar Council and the University of Melbourne, which allows me to conduct my research and significantly contributes to the thesis.

Finally, I would like to offer my special thanks to my parents for their love and support for me during this Ph.D. journey. Finishing this Ph.D. journey would not be possible without the support of my wife Yuqian Ma, thank you for your unwavering love and support.

Contents

Abstract i

Declaration iii

Preface iv

Acknowledgements v

List of Figures xii

List of Tables xv

1 Introduction 1
- 1.1 Considering Feature Interactions in Matrix Factorization . . . 4
- 1.2 Detecting Beneficial Feature Interactions for Recommendation . . . 5
  - 1.2.1 Detecting Pairwise Beneficial Feature Interactions . . . 5
  - 1.2.2 Detecting Arbitrary Order Beneficial Feature Interactions . . . 6
- 1.3 Modeling Feature Interactions based on Their Roles for Recommendation 7
  - 1.3.1 Modeling User and Item Attribute Interactions for Profile Learning and Preference Matching . . . 7
  - 1.3.2 Modeling Context Feature Interactions for Intrinsic and Extrinsic Factor Learning . . . 8
- 1.4 Thesis Outline . . . 9

2 Literature Review 10
- 2.1 Recommender Systems . . . 10
  - 2.1.1 Content-based Recommender Systems . . . 11
  - 2.1.2 Collaborative Filtering-based Recommender Systems . . . 12
    - 2.1.2.1 Memory-based Collaborative Filtering . . . 13
    - 2.1.2.2 Model-based Collaborative Filtering . . . 13
  - 2.1.3 Hybrid Recommender Systems . . . 15
    - 2.1.3.1 Separate Modeling . . . 15
    - 2.1.3.2 Unified Modeling . . . 16
- 2.2 Modeling Feature Interactions in Recommender Systems . . . 17
  - 2.2.1 Features and Feature Interactions . . . 17
  - 2.2.2 Implicit Feature Interaction Modeling . . . 19

2.2.3 Explicit Feature Interaction Modeling . . . 21
2.2.3.1 Factorization Machine-based Feature Interaction Modeling 21
2.2.3.2 Attention-based Feature Interaction Modeling . . . 23
2.2.3.3 Graph Neural Network-based Feature Interaction Modeling 25
2.3 Graph Neural Networks . . . 28
2.3.1 Graph Neural Networks . . . 28
2.3.2 Neural Graph Matching . . . 30
2.3.3 Other GNN Applications in Recommender Systems . . . 31
2.4 Techniques for Feature Interaction Detection . . . 34
2.4.1 L_0 Regularization . . . 34
2.4.2 Mutual Information Maximization . . . 35
2.5 Summary . . . 37

3 Considering Feature Interactions in Matrix Factorization 38
3.1 Introduction . . . 38
3.2 Preliminaries . . . 40
3.2.1 Matrix Factorization . . . 40
3.3 Our Proposed Method . . . 41
3.3.1 Multi-Matrix Factorization (MMF) . . . 41
3.3.2 Empirical Risk Minimization Function . . . 42
3.4 Discussions . . . 42
3.4.1 Relation to Matrix Factorization . . . 42
3.4.1.1 Personalized Interpretable Recommendation . . . 43
3.4.1.2 Item Cold-Start Problem . . . 43
3.4.1.3 Prediction Accuracy . . . 43
3.4.2 Relation to Factorization Machine . . . 44
3.5 Experiments . . . 44
3.5.1 Data Description . . . 44
3.5.1.1 Datasets . . . 44
3.5.1.2 Attribute Extraction . . . 45
3.5.1.3 Dataset Statistics . . . 45
3.5.2 Baseline Methods . . . 46
3.5.3 Model Performance . . . 47
3.5.3.1 Comparing with Matrix Factorization Methods . . . 47
3.5.3.2 Comparing with Feature Interaction-Based Baselines . . . 48
3.5.4 Different Weight Combinations . . . 48
3.5.5 Interpretability Case Study . . . 49
3.5.5.1 Semantic Dependency . . . 49
3.5.5.2 Interests Consistency . . . 50
3.5.6 Item Cold-Start Problem . . . 52
3.5.7 Evaluation on Different Latent Vector Lengths . . . 53
3.5.8 Evaluation on Different Topic Numbers . . . 53
3.6 Summary . . . 54

4 Detecting Pairwise Beneficial Feature Interactions for Recommender Systems 55
4.1 Introduction . . . 55

4.2 Preliminaries . . . 57
4.2.1 L_0 Regularization . . . 57
4.2.2 Graph Neural Networks . . . 58
4.3 Problem Statement . . . 58
4.4 Definitions . . . 59
4.4.1 Pairwise Beneficial Feature Interactions . . . 59
4.4.2 Statistical Interaction . . . 59
4.5 Our Proposed Method . . . 60
4.5.1 L_0-SIGN . . . 60
4.5.1.1 Model Overview . . . 60
4.5.1.2 L_0 Edge Prediction Model . . . 61
4.5.1.3 SIGN . . . 62
4.5.2 Empirical Risk Minimization Function . . . 64
4.6 Theoretical Analysis . . . 65
4.6.1 Satisfaction of Information Bottleneck (IB) Principle . . . 66
4.6.2 Relation to Spike-and-slab Distribution . . . 70
4.6.3 Statistical Interaction in SIGN . . . 71
4.6.4 Statistical Interaction in L_0-SIGN . . . 71
4.6.5 Time Complexity Analysis . . . 71
4.7 Experiments . . . 72
4.7.1 Experimental Protocol . . . 72
4.7.1.1 Datasets . . . 72
4.7.1.2 Baselines . . . 72
4.7.1.3 Experimental Set-up . . . 73
4.7.2 Overall Performance . . . 73
4.7.3 Comparing SIGN with Other GNNs in Our Model . . . 74
4.7.4 Evaluation of Interaction Detection . . . 76
4.7.4.1 Prediction Accuracy vs. Numbers of Edges . . . 76
4.7.4.2 Predicted Edges vs. Reversed Edges . . . 77
4.7.5 Case Study . . . 77
4.8 Summary . . . 79

5 Detecting Arbitrary Order Beneficial Feature Interactions for Recommender Systems 80
5.1 Introduction . . . 80
5.2 Preliminaries . . . 82
5.2.1 Hypergraph Neural Networks . . . 82
5.2.2 Deep Infomax . . . 83
5.3 Problem Statement . . . 84
5.4 Our Proposed Method . . . 84
5.4.1 Model Overview . . . 84
5.4.1.1 Data Representation with Hypergraphs . . . 84
5.4.1.2 The Overall Structure of HIRS . . . 85
5.4.2 The Recommendation Prediction Component . . . 85
5.4.2.1 Hyperedge Prediction . . . 86
5.4.2.2 IHGNN . . . 87
5.4.3 The s-Infomax and Infomin Component . . . 87

5.4.3.1 Three Properties of Beneficial Feature Interactions 88
5.4.3.2 s-Infomax and Infomin 89
5.4.3.3 The Implementation 89
5.4.4 The Empirical Risk Minimization Function 90
5.5 Discussions 91
5.5.1 IHGNN Satisfying the Statistical Interaction Principle 91
5.5.2 IHGNN Generalizes Existing Methods 92
5.5.2.1 L_0-SIGN as IHGNN 92
5.5.2.2 FM as IHGNN 92
5.5.2.3 DeepFM as IHGNN 92
5.5.3 Interaction Generation Cost Analysis 93
5.6 Experiments 93
5.6.1 Experimental Protocol 93
5.6.1.1 Datasets 93
5.6.1.2 Baselines 94
5.6.1.3 Experimental Setup 95
5.6.2 Overall Performance 95
5.6.3 Effectiveness of the Beneficial Properties 97
5.6.4 Ablation Study on RPC Modules 98
5.7 Running Time Analysis 99
5.7.1 Parameter Study 100
5.7.2 Case Study on the Detected Interactions 101
5.8 Summary 103

6 Modeling User and Item Attribute Interactions for Profile Learning and Preference Matching 104
6.1 Introduction 104
6.2 Problem Statement 106
6.2.1 Problem Statement 106
6.3 Our Proposed Method 107
6.3.1 GMCF Overview 107
6.3.2 User & Item Graph Construction 108
6.3.3 Node Matching based GNN 109
6.3.3.1 Neural Interaction based Message Passing 109
6.3.3.2 Bi-Interaction based Node Matching 109
6.3.3.3 Information Fusing 110
6.3.3.4 Node Representation Aggregation 110
6.3.4 Graph Matching 111
6.3.5 Model Training 111
6.4 Discussions 111
6.4.1 Relation to Existing Work and Discussion 111
6.4.1.1 Relation to L_0-SIGN 111
6.4.1.2 Relation to FM 112
6.4.1.3 When the User or Item Attributes are Unavailable 112
6.4.2 Time Complexity Analyses 112
6.5 Experiments 113
6.5.1 Experimental Protocol 113

6.5.1.1 Datasets . . . 113
6.5.1.2 Baselines . . . 114
6.5.1.3 Experimental Set-Up . . . 114
6.5.2 Overall Performance . . . 115
6.5.3 Study of Neural Graph Matching . . . 116
6.5.3.1 Evaluation of Inner and Cross Interaction modeling . . . 117
6.5.3.2 Evaluation of the Fusing Method . . . 118
6.5.3.3 Evaluation of the User and Item Attributes . . . 120
6.5.4 Parameter study . . . 121
6.5.5 Case Study . . . 123
6.6 Summary . . . 124

7 Modeling Context Feature Interactions for Intrinsic and Extrinsic Factor Learning 125
7.1 Introduction . . . 125
7.2 Preliminaries . . . 127
7.2.1 Variational Contrastive Log-ratio Upper Bound (vCLUB) of Mutual Information . . . 127
7.2.2 Related Work . . . 127
7.2.2.1 Contrastive Learning . . . 127
7.2.2.2 Factor Disentanglement for Recommender Systems . . . 128
7.3 Problem Statement and Definitions . . . 128
7.4 Our Proposed Method . . . 129
7.4.1 Recommendation Prediction Module . . . 129
7.4.2 Contrastive Intrinsic-Extrinsic Disentangling Module . . . 130
7.4.2.1 Context-Invariant Contrastive Learning Component . . . 131
7.4.2.2 Disentangling Component . . . 131
7.4.3 Multi-task Training . . . 132
7.5 Discussions . . . 133
7.5.1 Context-invariant Contrastive Learning in Information Theory . . . 133
7.5.2 Preventing a Trivial Solution in CIED . . . 133
7.5.3 Potential Problems of the Asymmetric vCLUB Method . . . 134
7.6 Experiments . . . 135
7.6.1 Experimental Protocol . . . 135
7.6.2 Overall Performance . . . 137
7.6.3 Effectiveness of Model Components . . . 138
7.6.3.1 Ablation Study of Contrastive Intrinsic-Extrinsic Disentangling Module . . . 139
7.6.4 Different Negative Context Generation Methods . . . 139
7.6.5 Falling Into Trivial Solutions . . . 140
7.6.5.1 Other Feature Modeling Methods . . . 141
7.6.5.2 Disentangling Component Evaluation . . . 142
7.6.6 Disentanglement Verification . . . 143
7.6.6.1 Visualization of Intrinsic and Extrinsic Representations . . . 143
7.6.6.2 Case Study . . . 144
7.6.7 Comparing the Impact of Different Contrastive Learning Variations 145
7.6.8 Effectiveness of Model Hyperparameters . . . 146

7.6.8.1 Embedding Dimension 146
7.6.8.2 The Number of Negative sample and Loss Weight 147
7.7 Summary 147

8 Conclusion and Future Works **149**
8.1 Conclusion 149
8.2 Future Works 151

A Proof of Theorem 4.3 **155**

B Proof of Corollary 4.4 **158**

C Proof of Proposition 5.1 **159**

D Proof of Proposition 5.3 **160**

E Proof of Theorem 7.2 **161**

Bibliography **164**

List of Figures

1.1 An illustration of features and feature interactions. User attributes, item attributes, and contexts are all features that can be leveraged to improve the performance of recommendation systems. Feature interactions are the features' co-occurrence in each data sample (a user-item pair). 2

2.1 The technique structure chart from recommender systems to our focused research techniques. We focus on improving explicit feature interaction modeling methods (the grey box) for recommender systems. Most of our works are focus on leveraging GNN-based technique (in red) for feature interaction modeling. 11
2.2 An example of collaborative filtering-based recommender systems on movie recommendation. 12
2.3 How matrix factorization algorithm learn user and item dense matrices of dimension 2 from the sparse user-item interaction matrix. 14
2.4 (a) The implicit feature interaction modeling framework models all feature interactions directly in a DNN. (b) The explicit feature interaction modeling framework model each feature interaction separately and aggregate the modeling results. 20
2.5 (a) A graph with nodes and edges. (b) How GNNs generate updated node embeddings (nodes in green) by the message passing technique. 25

3.1 An illustration of MMF's prediction procedure on a movie recommendation example. 40
3.2 The overall structure of the MMF model. 41
3.3 A visualization of attribute latent vectors in one movie (The latent vectors have been compressed into 3 dimensions using t-SNE). 50
3.4 The preference of user A (inner loop) and user B (outer loop). 51
3.5 The correlation between predicted attribute proportion and AAD. 52
3.6 The prediction results on item cold-start situation. 52
3.7 The test results of different latent vector length on MMF and baseline methods. 53

4.1 An overview of L_0-SIGN. 61
4.2 The interaction analysis results $\boldsymbol{s}$ from L_0-SIGN are like the relevant part (bottleneck) in the IB principle. 66
4.3 Linear Approximation vs. Bernoulli KL Divergence on different π values. 69
4.4 The comparison of improvement via the L_0 edge prediction technique to using heuristic edges in Table 4.3. 75
4.5 The changes in prediction accuracy and number of edges (in percentage) while training. 76

4.6 Evaluating different number of edges. "Pred" is the predicted edges and "Rev" is the reversed edges. 77
4.7 (a) The highest interaction values with *Gmail*. The numbers are city indexes (e.g., 1014). (b) The prediction that the user *u349* will use *Gmail* as the prediction value is $1.313 > 0$. Darker edges are higher interaction values. 78

5.1 *Left*: existing studies detect beneficial feature interactions of limited orders by examining all the possible feature interactions. *Right*: we detect arbitrary order beneficial feature interactions by directly generating them with our deep-infomax-based methods (s-Infomax and Infomin). 81
5.2 An Overview of HIRS. The part *above* the red dotted line is the recommendation prediction component. The part *below* the red dotted line is the s-Infomax and Infomin component and the L_0 activation regularization, which is only conducted during training. 85
5.3 The Venn diagram of the three properties. The *green* part is the information that our generated interactions to retain, and the *gray* part is to discard. 89
5.4 The distribution of the generated feature interaction orders from HIRS, HIRS without MI, and HIRS without L_0 regularization. 99
5.5 Comparing the performance of HIRS, HIRS without s-Infomax and Infomin, and HIRS without L_0 regularization. 100
5.6 The recommendation performance of HIRS, HIRS without hyperedge prediction, HIRS without non-linear interaction modeling, and HIRS without both. 100
5.7 The recommendation performance of HIRS with different hyperedge numbers. 101
5.8 The performance of different weight values of L_0 activation regularization, s-Infomax, and Infomin. 102
5.9 The detected feature interactions (in an incidence matrix) of a data sample in the MovieLens 1M dataset. Feature interactions are sorted by their orders and columns with the same color represent the interactions of the same order. 102

6.1 An illustration of the differences between GMCF (left) and existing work (right). GMCF treats attribute interactions differently in a structure of graph matching, while existing work treats all attribute interactions equally. 105
6.2 An overview of the GMCF model. 107
6.3 The comparison of using different inner interaction and cross interaction modeling combinations. 119
6.4 The comparison when using different attributes. 120
6.5 The performance of GMCF and selected baselines in different node (attribute) embedding dimensions. 121
6.6 The case studies on the MovieLens 1M dataset: *Left*: the embedding similarities between user age groups. *Right*: the embedding similarities between movie genres. 123
6.7 The case studies on the MovieLens 1M dataset: the node matching (preference) results between the ages groups and the genres. 123

7.1 An Overview of IEDR. It is a symmetric structure on the user side and the item side. The middle part (the black arrows) is the recommendation prediction (RP) module (Section 7.4.1). It generates factor representations ($\boldsymbol{o}_{in}$ and $\boldsymbol{o}_{ex}$) for producing the recommendation prediction y'. The side parts are two contrastive intrinsic-extrinsic disentanglement (CIED) modules. Each CIED includes a context-invariant contrastive learning component (the red arrows, Section 7.4.2.1), and a disentangling component (the blue arrows, Section 7.4.2.2) to ensure the success of the factor learning. The losses generated through the modules ($\mathcal{L}_{RP}, \mathcal{L}_{CICL}, \mathcal{L}_{bi\text{-}appr}, \mathcal{L}_{Dis}$) will be optimized as a two-step multi-task training (Section 7.4.3). 130
7.2 An illustrative example demonstrating the potential problem of the asymmetric learning in vCLUB. The blue circles are intrinsic representations and the red circles are extrinsic representations. The dotted arrows are the directions that *vCLUB* will push $\boldsymbol{o}^u_{in}$ and $\boldsymbol{o}^u_{ex}$ to move toward in their space. 135
7.3 Ablation studies with different component(s) removed. 139
7.4 The performance of different dropout rates for the method 2 (*NegGen2*). . 140
7.5 Visualizations of f_{ie} weights for the *Linear* and *Nonlinear* models. 141
7.6 Comparing the performance of the *Linear* and *Nonlinear* models on different datasets. 141
7.7 The model performance when equipped with different feature modeling methods. 142
7.8 The performance and variance statistics of vCLUB and BiDis. 142
7.9 Comparing of a user's top 100 intrinsic and extrinsic scores in two different contexts. 144
7.10 Hyperparameter study: different embedding dimensions d. 146
7.11 The performance of different numbers of negative samples and the loss weights in the risk minimization function for the contrastive learning component (left) and the disentangling component (right), respectively. 147
7.12 The complete intrinsic-extrinsic disentanglement visualizations in t-SNE. The blue dots are intrinsic representations, and the red dots are extrinsic representations. 148

List of Tables

2.1 Comparison of selective feature interaction modeling methods in several properties. . . . 19

3.1 Datasets statistics. . . . 45
3.2 The attribute details of datasets. . . . 46
3.3 The performance comparing with MF and variations. . . . 47
3.4 The performance comparing with attribute-based methods. . . . 48
3.5 The RMSE of different weight combinations in MMF. . . . 48
3.6 The performance of different topic numbers. . . . 53

4.1 Dataset statistics. All datasets are denoted in graph form. Twitter and DBLP datasets are used for question (ii). . . . 72
4.2 Summary of results in comparison with baselines. . . . 74
4.3 The results in comparison with existing GNNs. The model names without "L_0-" use heuristic edges, and those with "L_0-" automatically detect edges via our L_0 edge prediction technique. . . . 75

5.1 Dataset statistics. . . . 94
5.2 Summaries of whether the model can consider high-order feature interactions and perform beneficial feature interaction detection. . . . 95
5.3 Comparing the prediction performance of HIRS with the baselines in the MovieLens 1M dataset. . . . 96
5.4 Comparing the prediction performance of HIRS with the baselines in the Book-Crossing dataset. . . . 97
5.5 Comparing the prediction performance of HIRS with the baselines in the MovieLens 25M dataset. . . . 98
5.6 Comparing the time (in seconds) of different models for one epoch on the MovieLens 1M dataset. All the models are run on the same machine equipped with a CPU: Intel(R) Core(TM) i7-9700K @ 3.60GHz, and a GPU: GeForce RTX 2080Ti. 101

6.1 Dataset statistics. The *attr.* refers to "attributes". . . . 114
6.2 Summary of the performance in comparison with baselines on the MovieLens 1M dataset. . . . 116
6.3 Summary of the performance in comparison with baselines on the Book-Crossing dataset. . . . 116
6.4 Summary of the performance in comparison with baselines on the Taobao dataset. . . . 117
6.5 The performance of using different fusing algorithms. . . . 119
6.6 The performance of GMCF on using different MLP depths. . . . 122

7.1 Dataset statistics. "Count" refers to the number of users/items, "Features" represent the number of different features (for "User" and "Item", the number of features excludes the user/item ids). 136
7.2 Comparing the prediction performance (in percentage) with the baselines. The *Improv* and *p-value* rows show the relative improvements and the statistical significance of IEDR over the best performed baselines, respectively. N@k refers to NDCG@k and H@k refers to HR@k. 138
7.3 Comparing the performance of IEDR using different negative context generating methods (for the contrastive learning component). 140
7.4 The items (in category) of the highest intrinsic scores for different users. . 144
7.5 Comparing the performance of IEDR$_{sp}$ with different dropout rates (for *NegGen2*). 145

Chapter 1

Introduction

Since the advance of the world wide web (WWW), the information available to people has grown exponentially every day. In order to obtain useful information from the WWW, *recommender systems* are invented. Recommender systems are used to filter a small set of information that a user most needs and demonstrate the filtered information to the user. For example, a user may want to watch a movie, but there are vast number of movies available on the WWW. A recommender system infers the user's interests (e.g., by creating user profiles) and filters the most relevant movies based on the interests. Then, the filtered movies are demonstrated to the user.

Recommender systems provide personalized recommendation that can effectively improve users' experience in online surfing. Specifically, first, recommender systems can provide personalized information to users based on their profiles. For example, to pass the weekend, a user may want to watch a movie that meets her/his interest. A recommender system may recommend different movies to two users—documentary movies for an elderly user and action or adventure movies for a teenager. Second, recommender systems can capture the user requirements that are not easy to describe. Sometimes, a user only have a vague idea of which movie he/she wants to watch. These potential requirements can be captured by recommender systems when inferring users' interests. Therefore, recommender systems may provide the results that fit such requirements. Third, recommender systems can provide information to users even before they realize their needs, or be aware of existing/new information. For example, for a user with a tight working schedule, a recommender system can push a notification to him/her suggesting a relaxing movie to regain strength after work. As a result, recommender systems have attracted more attention in both academia and industry in recent years.

The goal of a recommender system is to predict whether a user will select an item (e.g., click, purchase). The historical selected items by users (i.e., user-item interactions)

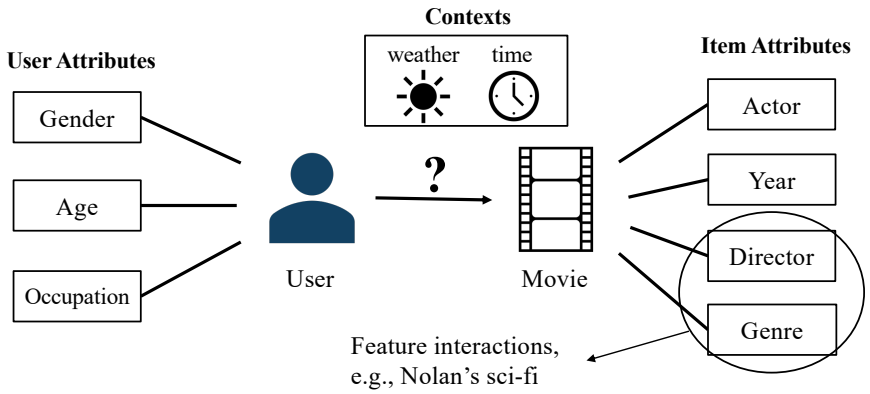


FIGURE 1.1: An illustration of features and feature interactions. User attributes, item attributes, and contexts are all features that can be leveraged to improve the performance of recommendation systems. Feature interactions are the features' co-occurrence in each data sample (a user-item pair).

are important information to predict new selections. For example, matrix factorization (MF) [1] is one of the most popular recommendation algorithms. It factorizes a sparse user-item (historical) interaction matrix into the multiplication of user and item latent vectors. Then the potential new selections can be predicted by multiplying corresponding user and item vectors.

Besides user-item interactions, features are essential side information to facilitate the prediction [2]. Standard features contain user/item attributes (e.g., user gender, item color) and context information (e.g., weather, time). Considering features in recommender systems can effectively improve prediction accuracy by providing fine-grained analysis and addressing the cold-start problem [3]. The cold-start problem can occur when no or very few historical user-item interactions are available. With features, the problem can be mitigated by matching items or users w.r.t. their features (e.g., the users of similar ages may prefer similar genres of movies). While leveraging features in recommender systems, considering the features' co-occurrence (i.e., *feature interactions*) has been proven essential in inferring useful information for more accurate predictions [4, 5]. For example, in a movie recommendation system, the director *Nolan* is a master of producing *sci-fi* movies. In this scenario, considering the feature interaction $\langle$*Nolan*, *sci-fi*$\rangle$ is more effective than considering the two features separately. Figure 1.1 shows examples of the standard features and feature interactions. While considering feature interactions, the final prediction can be the aggregation of all the feature interaction modeling results. Meanwhile, feature interactions inherently provide interpretability of the recommendation predictions [6]. The interaction modeling results show how much each feature interaction contributes to the prediction, which is widely accepted as an interpretation in regression tasks [7, 8].

However, with the development of deep neural networks (DNNs), recent methods model all feature interactions in a DNN (called *implicit* feature interaction modeling) [9, 10]. For example, DeepFM [5] concatenates the vector representations of the features (i.e., feature embeddings) into a large vector. The large vector is then inputted into a DNN, and the output of the DNN is the modeling results mixing the information of all feature interactions. Due to the black-box nature of DNNs, these methods cannot control which feature interactions to model and how to model each feature interaction. Meanwhile, modeling all feature interactions in a DNN undermines the interpretability of feature interactions. Therefore, this implicit way has recently considered to be not effective in extracting useful information from feature interactions in recommender systems [11].

To realize the full potential of feature interaction modeling for recommendation, we devote to performing *explicit* feature interaction modeling. The explicit way of modeling analyzes each feature interaction individually and aggregates the modeling results as the final prediction. Therefore, it provides flexibility in choosing which feature interactions to model and the methods to model them. In this way, it gives potential to deliver better feature interaction modeling and preserve the interpretability of feature interactions.

In this thesis, we explore novel techniques in explicit feature interaction modeling to improve recommender systems. Specifically, we focus on three important research questions:

- **RQ1**. Matrix Factorization (MF) is one of the most popular traditional recommendation algorithms. However, it does not consider feature interactions. How to *enable feature interaction modeling in matrix factorization*?
- **RQ2**. There are vast possible feature interactions that can be considered to produce recommendation predictions. However, only part of them are relevant to the predictions. How to *detect these relevant feature interactions* for recommendation predictions?
- **RQ3**. Feature interactions serve as different roles in recommender systems. How to categorize and model these feature interactions so they *fits their roles* for more accurate recommendation predictions?

While investigating the second the and third research questions, we explore the possibility of using graph neural networks (GNNs) [12, 13], a recently proposed DNN framework that is powerful in relational reasoning, for better explicit feature interaction modeling. GNNs have proven effective in many research tasks but have not been explored to facilitate feature interaction modeling in recommender systems. We find that the procedure of graph learning using GNNs is very similar to that of leveraging feature interactions

for recommendation. Specifically, the message passing procedure of GNNs learns the neighborhood information (node relations) for each node, which is similar to modeling feature interactions (a.k.a. feature relations); the node aggregation procedure of GNNs learns graph representations based on the information gathered through message passing, which is similar to aggregating the feature interactions modeling results for prediction.

To effectively incorporate GNNs for feature interaction modeling, we propose a novel way to represent feature interactions in a graph structure. Specifically, each feature represents a node in a graph, and each feature interaction represents an edge that links to the corresponding nodes. In this way, a data sample with feature information is converted to a graph structure containing features and feature interactions. With the graph structure, we effectively leverages the relational reasoning ability of GNNs for explicit feature interaction modeling. The results show that it has great potential to provide accurate and interpretable feature interaction modeling for recommendation.

In the following subsections, we introduce how we address each research question.

1.1 Considering Feature Interactions in Matrix Factorization

Matrix factorization is one of the most popular algorithms in the design of recommendation systems. It has been widely used by academia and industry. However, since MF cannot consider feature interactions, it suffers from two drawbacks: lack of interpretability, and the cold-start problem. The two drawbacks limit its reliability and practicability. In Chapter 3, we investigate the first research question and explore how to enable matrix factorization to consider feature interactions. We propose a recommendation model called *Multi-Matrix Factorization (MMF)*, which overcomes these two limitations and achieves state-of-the-art prediction accuracy by exploiting the interactions between item attributes and users. MMF provides more fine-grained analyses than MF in the following ways: the final predictions are the weighted sum of interaction modeling results, which indicate how much each attribute contributes to the recommendation and hence provide interpretability; the common attributes can act as a link between the existing and new items, which resolves the item cold-start problem.

The advantage of our work is that MMF simultaneously achieves the following goals:

- Interpretability: MMF analyzes the contribution of each user-attribute interaction to the final predictions, such as which attribute in an item may attract a user most. We

conduct comprehensive interpretability evaluations to show the effectiveness of the interpretations.

- Resolving the cold-start problem: MMF provides reasonable predictions in item cold-start cases. Items may share attributes. This connection allows the transfer of user-attribute ratings from existing items to a new item.
- Better prediction accuracy: MMF provides a novel way to learn fine-grained differences between items through user-attribute interactions. As a result, MMF achieves better prediction accuracy than state-of-the-art matrix factorization models and feature interaction-based recommendation models.

1.2 Detecting Beneficial Feature Interactions for Recommendation

Many existing studies take into account all possible feature interactions. However, this is suboptimal because some feature interactions may not be relevant to the recommendation result (i.e., unbeneficial feature interactions), and taking them into account can introduce noise that decreases accuracy and efficiency [14, 15]. For example, a user may enjoy recently released movies or old movies, regardless of gender. Therefore, considering the interactions between gender features and movie-year features is unnecessary and may cause less accurate recommendations.

In Chapter 4 and Chapter 5, we conduct two works to study the second research question and explore how to detect beneficial feature interactions for recommendation. We explore to detect beneficial feature interactions as a link prediction task in a graph representation, then model and aggregate the detected feature interactions for recommendation prediction as a graph classification task using GNNs.

We introduce the two works below.

1.2.1 Detecting Pairwise Beneficial Feature Interactions

To identify the most influential pairwise feature interactions, we propose a novel technique to automatically detect those pairwise feature interactions that are beneficial for providing accurate predictions. Then, we propose a graph neural network method to model the detected feature interactions for recommendation. The automatic feature interaction detection is achieved via edge prediction with an L_0 activation regularization. Our proposed model has proven to be effective through the information bottleneck principle and statistical interaction theory.

We summarize the main contributions of this work as follows:

- This is the first work to formulate the concept of beneficial feature interactions, and propose the problem of detecting beneficial feature interactions for recommender systems.
- We propose a model, L_0-SIGN, to detect beneficial feature interactions via a graph neural network approach and L_0 regularization.
- We theoretically prove the effectiveness of L_0-SIGN through the information bottleneck principle and statistical interaction theory.
- We have conducted extensive experimental studies. The results show that (i) L_0-SIGN outperforms existing baselines in terms of accuracy; (ii) L_0-SIGN effectively identifies beneficial feature interactions, which result in the superior prediction accuracy of our model.

1.2.2 Detecting Arbitrary Order Beneficial Feature Interactions

Recent researches prove that more sophisticated forms of feature interactions (i.e., high-order interactions) help achieve more accurate recommendation predictions [16, 17]. In the previous work, our proposed L_0-SIGN detects pairwise beneficial feature interactions by examining all possible pairwise feature interactions. However, when it comes to higher-order feature interactions, the time complexity of examining all the possible higher-order feature interactions is prohibitive (exponentially growing with the order increasing). In this work, we propose a HIRS model to directly generate beneficial feature interactions of arbitrary orders. It uses a hypergraph (an extension of a standard graph that each edge, i.e., hyperedge, can link to any number of nodes) to represent the generated beneficial feature interactions of arbitrary order. Then, HIRS makes recommendation predictions via hypergraph classification. The complexity of HIRS is linear to the number of beneficial interactions to be detected; this number may be specified to be one that is much smaller than the number of all the possible interactions and hence incurs a much lower time complexity. To achieve an effective algorithm, we exploit three properties of beneficial feature interactions (sufficiency, low redundancy, and disentanglement). Then, we propose deep-infomax-based methods based on the three properties to guide the interaction generation.

We summarize the main contributions of this work as follows:

- We propose a hypergraph neural network-based model, HIRS, which directly generates a set of beneficial feature interactions of arbitrary orders, and then performs recommendation prediction via hypergraph classification.
- We define three properties of beneficial feature interactions and propose s-Infomax and Infomin methods for beneficial feature interaction generation. We further propose an interaction-based hypergraph neural network (IHGNN), which effectively learns from the generated interactions for recommendation.
- Experimental results show that HIRS outperforms state-of-the-art feature interaction-based models for recommendation. Further evaluations are conducted to show our model's beneficial feature interaction generation ability.

1.3 Modeling Feature Interactions based on Their Roles for Recommendation

In Chapter 6 and Chapter 7, we conduct two works to explore the third research question of how to categorize feature interactions to enhance the recommendations. We focus on two scenarios. In the first scenario (Chapter 6), we categorize user and item attributes into two types: one is for learning user/item profiles, and the other is for matching users' preference to items. Then we propose a neural graph matching-based algorithm to effectively model the two types of feature interactions for recommendation. In the second scenario (Chapter 7), we focus on leveraging context features to distinguish two kinds of factors that jointly impact users' decision-making in recommendation: intrinsic factor and extrinsic factor. We propose an intrinsic-extrinsic disentanglement method that can effectively learn the two kinds of factors to infer users' decisions more accurately.

We introduce the two works below.

1.3.1 Modeling User and Item Attribute Interactions for Profile Learning and Preference Matching

User and item attributes are essential features; capturing their interactions can significantly enhance prediction accuracy. However, existing models do not distinguish the impacts of attribute interactions in a recommendation, which may not be the most effective way to exploit the information carried by the interactions. To address this drawback, we identify two types of attribute interactions, *inner interactions* and *cross interactions*: inner interactions include interactions between only user attributes or only item attributes; cross interactions include iterations between user attributes and

item attributes. We propose a neural Graph Matching based Collaborative Filtering (GMCF) model, that effectively captures these attribute interactions through modeling and aggregates attribute interactions in a graph matching structure. In our model, the two essential recommendation procedures, profile learning and preference matching, are explicitly conducted through graph learning (based on inner interactions) and node matching (based on cross interactions), respectively.

We summarize the main contributions of this work as follows:

- We highlight the importance of distinguishing the impacts of attribute interactions on recommendation predictions, and categorize them into inner interactions and cross interactions. These interactions should be treated differently when they play different roles in the final predictions.
- We propose a novel neural graph matching-based model, GMCF, that effectively leverages inner interactions for user/item profile learning (through graph learning), and exploits cross interactions for preference matching (through graph matching). To our best knowledge, we are the first to represent each user and each item as an attribute graph, and model and aggregate attribute interactions in a graph matching structure.
- We conduct extensive experiments. Experimental results show that (i) our model outperforms the state-of-the-art baselines in terms of accuracy; (ii) GMCF can effectively model the two types of attribute interactions and achieves accurate predictions.

1.3.2 Modeling Context Feature Interactions for Intrinsic and Extrinsic Factor Learning

Context features (e.g., time, weather) are also important in recommender systems, but they are not considered in GMCF. In this work, we explore how to leverage context features for more accurate recommendation.

Inspired by psychology, we find that context feature interactions are perfect in distinguishing *intrinsic* and *extrinsic* factors that mainly influence users' decision-making. Intrinsic factors reveal users' real interest that is invariant in different contexts, whereas extrinsic factors change within different contexts. Accordingly, we propose an Intrinsic-Extrinsic Disentangled Recommendation (IEDR) model that explicitly learns intrinsic and extrinsic factors, and leverages them to provide recommendations. Similar to GMCF, we leverage graph neural networks to conduct explicit feature interaction modeling for user, item, and context representation learning. Then, the intrinsic and extrinsic factors are inferred based on the learned representations. Meanwhile, we propose two

components—a context-invariant contrastive learning component, and a mutual information minimization-based disentangling component—to ensure the success of the factor inferring collaboratively.

We summarize the main contributions of this work as follows:

- To the best of our knowledge, this is the first work to define intrinsic and extrinsic factors for recommender systems with respect to context. The two types of factors are also generated from the perspectives of both users and items for accurate recommendation.
- We propose IEDR to explicitly learn disentangled intrinsic and extrinsic factor representations. A context-invariant contrastive learning component and a mutual information minimization-based disentangling component are proposed to ensure the success of factor inferring.
- Experimental results show that IEDR significantly outperforms state-of-the-art methods in terms of recommendation accuracy through learning intrinsic and extrinsic factor representations.
- We conduct theoretical analyses to demonstrate the connection of context-invariant contrastive learning with information theory.

1.4 Thesis Outline

The thesis is organized as follows. In Chapter 2, we provide a literature review on related work. Our review includes different types of recommender systems, how existing methods model feature interactions for recommendation, and graph neural networks with their application in recommender systems. Additionally, we will review other techniques that are related to our proposed works, including L_0 regularization and mutual information maximization. From Chapter 3 to Chapter 7, we detail our works, respectively. Each chapter contains a detailed introduction to our corresponding work, a complete description of the proposed method, and extensive experimental results to prove the effectiveness of our work. In Chapter 8, we provide a summary of the thesis. In addition, we demonstrate some potential future work.

Chapter 2

Literature Review

This chapter reviews existing studies that are relevant to our work. We review these studies in the order from broad recommendation strategies to our focused techniques. Specifically, we first introduce three broad types of recommender systems, and show that the feature interaction-based methods belong to one of them (hybrid recommender systems). Then, we give a detailed discussion about modeling feature interactions in recommender systems. We review both implicit and explicit feature interaction modeling methods. Focusing on the explicit feature interaction modeling, we discuss studies on three promising strategies: factorization machine-based strategies, attention-based strategies, and graph neural network-based strategies. Next, we review graph neural networks in various domains such as neural language processing and chemical structure search. We also review the recommender systems leveraging graph neural networks for other purposes such as social graphs learning and knowledge graphs learning. Finally, we discuss techniques we used for feature interaction detection, including L_0 regularization and mutual information maximization.

2.1 Recommender Systems

With the development of the World Wild Web, the available information to people has grown exponentially. The excess of information available to a person, usually called the information overload problem, results in difficulty for people to obtain useful information for decision-making [18, 19]. Recommender systems, therefore, were proposed as a critical solution to mitigate this challenge. A recommender system is a technique that provides suggestions for information (usually are items) to be of use to a user [20–22]. A good recommender system can save the effort of users from searching and filtering vast

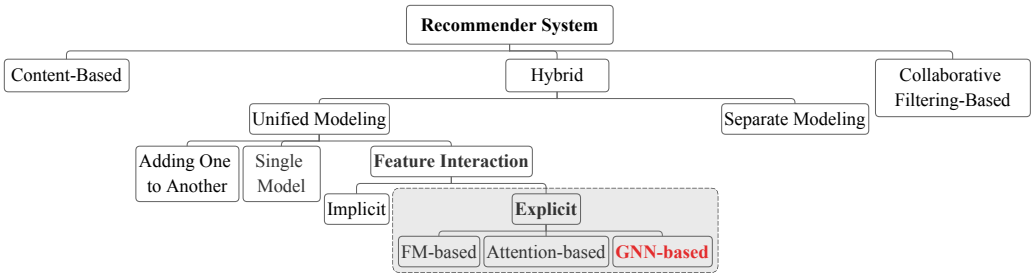


FIGURE 2.1: The technique structure chart from recommender systems to our focused research techniques. We focus on improving explicit feature interaction modeling methods (the grey box) for recommender systems. Most of our works are focus on leveraging GNN-based technique (in red) for feature interaction modeling.

amount of information from the whole Web. As a result, recommender systems have gain increasing interest from both academia and industry.

There are various strategies to design a recommender system. Figure 2.1 illustrates the technique structure chart of recommender systems. We focus on feature interaction-based recommender systems (the dotted box). Specifically, we mainly try to leverage graph neural network (GNN)-based techniques (in red) to learn feature interactions. In the following sections, we will introduce related works based on the techniques in the figure in order.

2.1.1 Content-based Recommender Systems

A content-based recommender system leverages feature information of an item (a user) to construct the profile of the item (the user). Then, the system will recommend an item to a user if this item has a similar profile to the items that have been rated high by the user previously [23]. For example, in a movie recommendation task, there are two movies, *Interstellar* and *Inception*, that may be considered similar in a content-based recommender system (they are both adventure and sci-fi movies directed by Christopher Nolan). If a user watched Interstellar, the system may also recommend Inception to this user.

In recent studies on content-based recommender systems, the weight of features is considered to differentiate the feature's importance in computing the similarities between the profiles of items (users) [24, 25]. Content-based recommendation methods have close relationships with information retrieval and information filtering research [26–29]. Therefore, the techniques of information retrieval and information filtering can be used to improve recommendation accuracy, such as TF-IDF [30–33] and n-gram [34–36]. These approaches are commonly used in various recommendation applications like web page recommendation and book recommendation. For example, Fab system [37] and Syskill

& Webert system [38] select a list of important keywords in web pages and construct the profile of the web pages based on these keywords. Some studies combine the Semantic Web with content-based recommendations and use the feature relationships in the Semantic Web to provide users with recommendations [39, 40]. Other studies learn the content in the news and propose content-based new recommender systems [41–43]. The content-based recommendation can be applied in many other applications as long as the item or user features are available, such as the rhyme and mood in a piece of music for music recommendation [44, 45], the genres, casts and directors of movies for movie recommendation [25, 46], and the product categories and style in e-commerce recommendation [47–49].

Content-based recommender systems learn users' preferences effectively through considering features. However, these recommender systems do not capture the opinions of other users on the same item. Also, if there is a feature that a user has never interacted with before, the system does not know whether the user likes the feature or not.

2.1.2 Collaborative Filtering-based Recommender Systems

Unlike content-based recommender systems, collaborative filtering-based recommender systems produce recommendations based on the assumption that similar users will have shared preferences on similar items (i.e., collaborative signals), which has been proven effective for generating accurate recommendations [26, 50]. In Figure 2.2 as an example, two users have watched similar movies (the gray and blue ones). Thus, the two users are considered similar users. Since the female user watched another movie (the orange one), the system believes that the male user will also like that movie.

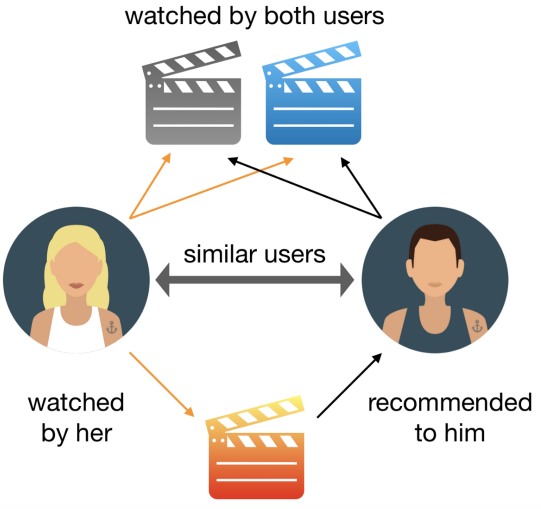


FIGURE 2.2: An example of collaborative filtering-based recommender systems on movie recommendation.

There are two categories of collaborative filtering frameworks: memory-based collaborative filtering and model-based collaborative filtering [51]. We discuss them in the following subsections.

2.1.2.1 Memory-based Collaborative Filtering

Memory-based collaborative filtering methods leverage the historical collection of user-item interactions (e.g., purchase history, ratings) to make predictions [50, 52]. New items will be recommended to a user u_i by aggregating the interactions of similar users on this item. There are different methods perform the aggregation. The simplest way is to average all the corresponding interactions [26]. However, this method supposes that all similar users have the same weight on the final predictions. This is unreasonable since some users may be more similar to user u_i, which should be taken more weights while performing the aggregation. A more reasonable aggregation method is to considers user similarity as a weight and it has been proven can achieves better recommendation performance [51]. Therefore, similarity measurements are vital for memory-based collaborative filtering methods as they are used for searching for similar users and used for calculating the weights while performing the aggregation [53]. While calculating the similarity, many measurements are applied in recommender systems, such as cosine similarity [54–57], correlation-based similarity [58–62], and conditional probability-based similarity [63, 64]. More recently, machine learning algorithms are shown to be effective, and model-based collaborative filtering has become more popular than memory-based collaborative filtering in recommender systems.

2.1.2.2 Model-based Collaborative Filtering

In contrast to memory-based methods, model-based collaborative filtering learns a model based on a collection of ratings. Then the learned model can make recommendation predictions. Different types of models can be learned to perform the recommendation predictions.For example, Bayesian network-based collaborative filtering methods leverage the Bayesian rule to calculate the conditional probability of a user interacting with an item given that they have already rated another item. [65–67]. Clustering-based collaborative filtering methods leverage clustering algorithms to group similar users (or items) so that the similarity of interests can be explored inside each cluster [68–70]. Latent semantic-based collaborative filtering methods such as principal component analysis (PCA) based recommender systems reduce the dimension of the rating matrix to better capture similarities between users (items) [56, 71], and singular value decomposition

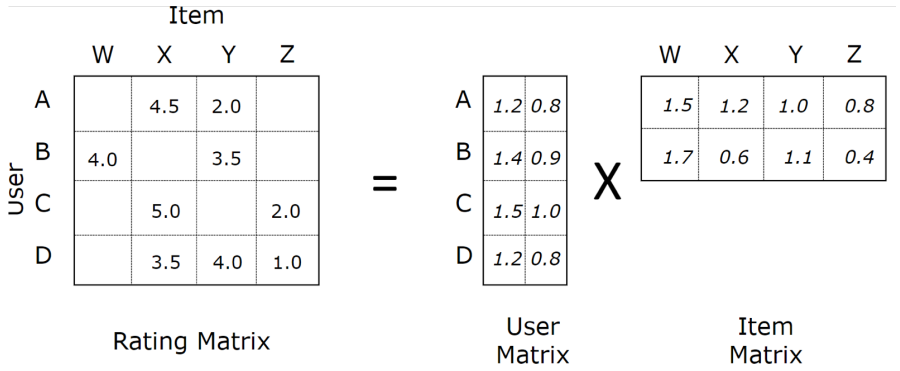


FIGURE 2.3: How matrix factorization algorithm learn user and item dense matrices of dimension 2 from the sparse user-item interaction matrix.

(SVD) based recommender systems decompose the sparse rating matrix into the product of several low-rank matrices so that unseen interactions in the user-item interaction matrix can be restored [72–74].

Similar to SVD based methods, the matrix factorization (MF) model factorizes the rating matrix into the product of a user matrix and an item matrix. Each user (item) is represented as a latent vector in the matrix. Then, whether a user will interact with an item can be predicted via performing a dot product between the user vector and the item vector. Figure 2.3 visualizes the learning procedure of MF. This simple and effective algorithm got the top performance on Netflix price competition in 2007 [1]. Since then, MF has attracted much attention from academia and industry, and is now one of the most popular collaborative filtering algorithms.

Many extensions of MF have been proposed and achieved better performance. For example, non-negative matrix factorization forces the values of user and item matrices to be non-negative. As a result, the meaning of the vectors is easier to inspect, and the recommendation performance is improved [75–77]. Mnih and Salakhutdinov [78] propose probabilistic matrix factorization (PMF) that represents user and item matrices by probabilistic models, which achieves great performance on the large, sparse, and very imbalanced Netflix dataset. Salakhutdinov and Mnih [79] further improve the recommendation performance by using a Bayesian PMF treatment that integrates all model parameters and hyperparameters, which can be efficiently trained through Markov chain Monte Carlo methods (MCMC). To avoid manually choosing the best dimension of the user and item matrices, Srebro et al. [80] propose Maximum Margin Matrix Factorization (MMMF) that can extend the matrices to infinite dimension through kernel methods. Rennie and Srebro [81] analyze the property of the kernel method and propose a Fast MMMF to improve the efficiency. MMMF-based methods have been successfully applied in various recommender systems [82–84].

Recently, with the development of neural networks, He et al. [85] propose neural collaborative filtering (NCF) that combines the results of a neural network and a general MF modeling user and item interactions, respectively. Xue et al. [86] unified the MF model and neural networks. They proposed a deep matrix factorization (DMF) that directly uses the rating matrix as the input of a neural network and it outputs the latent representations of users and items. Then DMF multiplies the user latent representations and item latent representations to generate the predictions. He et al. [87] perform an outer product between user and item matrices to generate an interaction map and leverage convolutional neural networks to learn high-order correlations among the interaction map to generate recommendation predictions. Liang et al. [88] perform matrix factorization in a variational autoencoder (VAE), a non-linear probabilistic model, to enable matrix factorization to perform non-linear modeling. Wang et al. [89] consider the data distribution inconsistency issue in training and test dataset and propose a doubly robust approach to obtain unbiased performance estimation. This approach performs well in various matrix factorization-based models.

However, collaborative filtering-based recommendation algorithms only use the rating matrix, which cannot leverage the feature information. Meanwhile, it may raise the cold-start problem. The problem appears when a user rate few items or an item is rated by few user. As a result, this user/item does not have enough collaborative signal to be well trained. Therefore, the hybrid recommender systems are invented to effectively incorporate the advantages of content-based methods and collaborative filtering-based methods.

2.1.3 Hybrid Recommender Systems

Hybrid recommender systems seek to combine the content-based recommendation methods and collaborative filtering methods to leverage the advantages of both the content-based methods (e.g., can handle the cold-start problem) and collaborative filtering methods (e.g., leverage the collective intelligence). We categorize the hybrid recommender systems into two categories. The first is to combine them via *separate* modeling, and the second via *unified* modeling.

2.1.3.1 Separate Modeling

The most intuitive approach to combining the two types of methods is to run recommendation algorithms of different methods separately and then fuse their results as the final prediction. For example, Miranda et al. [90] aggregate the scores returned by a content-based recommender system and a collaborative filtering-based recommender systems.

Billsus and Pazzani [91] perform a switch strategy that first use a content-based recommender system to generate a prediction. If the prediction confidence is not high, it will then use a collaborative filtering-based recommender system to generate the prediction. Smyth and Cotter [92] recommend a list of items that is generated by mixing several set of items predicted by different recommender systems. However, the separate modeling is suboptimal since the predictions do not combine the advantages of the two types of methods.

2.1.3.2 Unified Modeling

Another approach is to propose a unified model that contains the characteristics of both content-based methods and collaborative filtering-based methods.

Some studies based on this approach can *incorporate one method as the component of another method*. There are basically two strategies. The first strategy is to incorporate content-based characteristics to collaborative filtering-based models. Pazzani [93] propose a "collaboration via content" approach that leverage the content information (e.g., user attributes) to calculate the similarity between users. Good et al. [94] design various agents to perform content analysis additional participants in collaborative filtering models. The second strategy is to incorporate collaborative filtering-based characteristics to content-based models. Soboroff and Nicholas [95] learn user latent vector representations via leveraging the latent semantic indexing technique to create the collaborative views of users. The learned representations are then leveraged to perform pure content-based recommendations.

Other studies focus on designing a *single* model to combine both the two methods. Basu et al. [96] propose a single rule-based classifier that can learn both content-based and collaborative signals. Ansari et al. [97] and Condliff et al. [98] explore leveraging Bayesian-based regression models to model content information for user and item profile learning, and leveraging their interactions as the final recommendation prediction. Schein et al. [99] use the probabilistic latent semantic analysis to combine the characteristics of the two types of methods in a probabilistic model.

In recent research, *feature interaction modeling* has been proven to be a promising approach to combining the advantages of content-based recommendation and collaborative filtering-based recommendation [2, 100]. This is because feature interaction modeling learns the relations between features, which simultaneously captures the content information and collaborative signals. Modeling feature interactions can somehow consider as combining both the two methods into a single model. However, due to its uniqueness in combining the content-based and collaborative filtering-based characteristics, we

consider it a new category of unified modeling approaches. Feature interaction-based recommender systems can be categorized in to two frameworks: *implicit* feature interaction modeling [5, 10, 101, 102] and *explicit* feature interaction modeling [4, 11, 16, 103]. In the next section, we explain how feature interactions can be leveraged in recommender systems.

2.2 Modeling Feature Interactions in Recommender Systems

As discussed in the last section, modeling feature interactions is a promising technique for combining content-based and collaborative filtering methods. In this section, we first give an introduction to feature interactions in recommender systems. Then, we discuss two frameworks of modeling feature interactions: implicit modeling framework and explicit modeling framework.

2.2.1 Features and Feature Interactions

Features. In machine learning, features are critical elements in many algorithms for classification and regression tasks [104]. In recommender systems, standard features can be categorized as: (1) user/item features such as user attributes (e.g., ages, genders) and item attributes (e.g., genres or directors of movies), and (2) context features (e.g., weather, daytime). For example, to recommend a movie, a recommender system leverages the user/item attributes, such as the movie's popularity, to a certain age of users. In addition, if the system requires to suggest a timing to watch the movie in a theater, context features can help since fewer people would like to go to a theater on a rainy day or workday.

Feature interactions. Feature interactions are the co-occurrence of features in a data sample input that jointly influence the final output. Considering feature interactions is proven more effective than considering features independently. In the above example, people may not go to a theater only because it is a weekend or because the weather is sunny. Therefore, considering the time feature *weekend* and the weather feature *sunny* together can provide more useful information for the recommendation prediction.

The data to train recommender systems considering feature interactions have the input-output format (X, y), where $X = \{x_i\}_{i=1:p}$ (i.e., a data sample) is a set of features that will be inputted into the system, p is the number of features in the input, and y is the output (e.g., whether the user purchase the item).

Considering feature interactions resolves the cold-start problem and sparsity problem in recommender systems. The cold-start problem is resolved by linking the users/items. For example, a feature interaction may appear in multiple items. Therefore, a recommender system can capture the knowledge that these items may share some common characteristics. As a result, the system can generate reasonable predictions based on the knowledge even if one of the items has not been seen by any user (cold-start situation). The sparsity problem is that there may be millions of inputs in a dataset, while a feature may only appear in a few inputs, let alone the feature interactions. It can be resolved by modeling feature interactions via capturing the similarities between different features. For example, a dataset contains features A, B, C, D. The feature interactions $\langle A, C\rangle, \langle A, D\rangle$ appear in two data samples, respectively, and the two data samples have similar outputs. Then, after training, the features C and D will be considered similar. Another combination, $\langle B, C\rangle$, also appears in the training set and is well trained. While testing, even if the feature combination $\langle B, D\rangle$ has never been seen in the training set, the model can produce well analysis: the feature interaction modeling result of $\langle B, D\rangle$ is similar to $\langle B, C\rangle$ since feature C is similar to D.

Another advantage of considering feature interactions is that it can produce potential interpretability of the prediction results. Broadly speaking, a machine learning model is interpretable if it can reveal relevant knowledge about domain relationships contained in data [8, 105]. For example, linear models (e.g., linear regression) are widely accepted as having interpretability [106, 107]. A linear regression model weightily aggregates features to get the final prediction, and the weight of each feature is considered to indicate the contributions of this feature to the prediction. The interpretability of feature interactions in recommender systems follows the same idea as the linear regression models. If a recommender system considering feature interactions can obtain the modeling result (e.g., a scalar value) of each feature interaction, these modeling results can be regarded as the contributions of these feature interactions.

In the following subsections, we introduce representative feature interaction-based recommender systems. There are two basic types of feature interaction modeling frameworks: implicit feature interaction modeling and explicit feature interaction modeling. In Table 2.1, we give a comparison of selective feature interaction modeling methods (including our proposed five models in this thesis) w.r.t. different properties, including (1) the interaction modeling frameworks, (2)detailed feature interaction modeling approaches, (3) order of feature interactions (higher-order or pairwise), and (4) types of feature interaction modeling (explicit or implicit).

From Table 2.1, we can see that most existing studies model feature interactions based

Mthods	Framework	Modeling Method	Orders	Explicit
DLRM [102]	DNN	MLP	Arbitrary	
DIN [10]	DNN	MLP	Arbitrary	
W&D [9]	DNN	MLP/linear model	Arbitrary	
DeepFM [5]	DNN	MLP/FM	Arbitrary	
FM [2]	FM	dot product	Pairwise	√
AFM [4]	FM	dot product+attention	Pairwise	√
NFM [108]	FM	Bi-linear+MLP	Pairwise	
xDeepFM [16]	FM	dot product/CrossNet	Limited	√
AutoFIS [17]	FM	dot product+gate	Limited	√
AFN [109]	FM	logarithmic transformation	Limited	√
AutoInt [11]	Attention	self-attention	Limited	√
Fi-GNN [103]	GNN/Attention	self-attention	Pairwise	√
MMF (Chapter 3)	MF	dot product	Pairwise	√
L_0-SIGN (Chapter 4)	GNN	MLP	Pairwise	√
HIRS (Chapter 5)	HGNN	MLP + Infomax	Arbitrary	√
GMCF (Chapter 6)	Graph Matching GNN	Bi-linear/MLP	Pairwise	√
IEDR (Chapter 7)	GNN	MLP	Pairwise	√

TABLE 2.1: Comparison of selective feature interaction modeling methods in several properties.

on the DNNs and factorization machine (FM, we will detail it in Section 2.2.3) frameworks. Our works (except MMF) are based on the more effective GNN framework. Meanwhile, existing works can only leverage DNNs to implicitly model arbitrary order feature interactions (DLRM, DIN, DeepFM, etc.). In contrast, our HIRS model is the first to model arbitrary order feature interactions explicitly. We discuss these methods in the following subsections in whether they are implicit or explicit feature interaction modeling.

2.2.2 Implicit Feature Interaction Modeling

With the development of deep neural networks (DNNs), there is an increasing demand to leverage DNNs for feature interaction modeling. Implicit feature interaction modeling is a straightforward framework that models feature interactions using DNNs. Figure 2.4 (a) illustrates how implicit feature interaction modeling works. Specifically, a recommender system modeling feature interactions implicitly first concatenates all the input features together. For example, each categorical feature can be represented as a latent vector of d dimensions (i.e., feature embedding). A numerical feature can be represented as a scalar value. Then, all the feature representations are concatenated into a large vector as the input of a DNN (e.g., a multilayer perceptron). The feature interaction modeling will be conducted in the DNN and output a result (a scalar value or a vector) indicating the modeling results of all the feature interactions [5]. It is called implicit modeling because of the black-box nature of DNNs. We neither know which feature interactions are modeled nor how they are modeled inside the DNN. In the following, we introduce some implicit feature interaction modeling-based recommender systems.

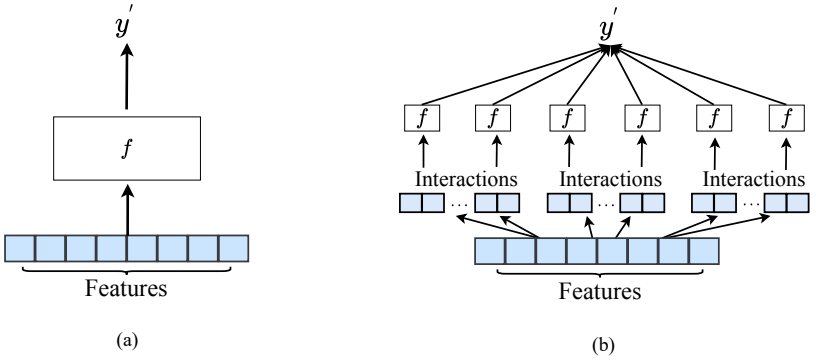


FIGURE 2.4: (a) The implicit feature interaction modeling framework models all feature interactions directly in a DNN. (b) The explicit feature interaction modeling framework model each feature interaction separately and aggregate the modeling results.

The Wide & Deep algorithm [9] is a hybrid framework that simultaneously considers the linear relationships (wide) between features through a linear regression model and the non-linear relationships (deep) between features through deep neural networks. It combines the memorization ability of linear regression models and the generalization ability of deep neural networks to achieve accurate recommendation predictions. Guo et al. [5] propose DeepFM based on the finding that a multilayer perceptron (MLP) can implicitly model any order feature interactions [110].

Implicit feature interaction modeling also attracts attention in the industry due to its simplicity and effectiveness. These models have similar structures of modeling feature interactions—concatenate feature information and input it into a multilayer perceptron (MLP) to directly output the final prediction. The differences are the choice of features and the methods to construct the MLP input. Covington et al. [101] propose a deep candidate generation model to recommend videos to users on YouTube. The model concatenates users' historical features, such as historical watched videos and searched histories, and user/item features, such as geographic information and user gender, and inputs all the features into an MLP. Naumov et al. [102] propose DLRM for Facebook. It concatenates features and pairwise feature interactions modeling results using the FM as the input of an MLP. Zhou et al. [10] propose DIN for Taobao's item prediction. Besides the common features such as user attributes and item attributes, it proposes a deep interest network to learn user behaviors as part of the input of MLP to capture users' interests. Following extensions of DIN such as DIEN [111] and BST [112] focus on performing a better user behavior learning, while their frameworks of learning feature interactions are the same.

Although the implicit feature interaction modeling achieves success in recommender systems, the black-box nature of DNNs diminishes the interpretability and flexibility of the feature interactions, limiting its potential for further improving the recommendation

accuracy. Meanwhile, later researches have experimentally proved the ineffectiveness of the implicit way in extracting useful information from feature interactions. [11, 16, 113, 114].

2.2.3 Explicit Feature Interaction Modeling

Unlike the implicit method, the explicit feature interaction modeling method analyzes each feature interaction individually, then aggregates the results as the final predictions. Figure 2.4 (b) illustrates how explicit feature interaction modeling works.

In this subsection, we introduce three types of promising explicit feature interaction modeling frameworks in recommender systems: (1) factorization machine-based feature interaction modeling, (2) attention-based feature interaction modeling, and (3) graph neural network-based feature interaction modeling.

2.2.3.1 Factorization Machine-based Feature Interaction Modeling

Factorization machine (FM) [2] is one of the most popular algorithms for modeling feature interactions for recommendation. FM regards each feature as a pair (a, w), where a is the name of the feature and w is the corresponding feature value in each data sample. Note that each feature may have different feature values in different inputs. For example, the age of a user can be regarded as a continuous feature and is represented as $(age, 0.35)$, where 0.35 is the normalized age of that user. For categorical features, FM regards them as having the value 1 to indicate that the feature appears in a data sample. For example, a female user will have the gender feature "female" and can be regarded as $(female, 1)$. FM represents each feature as a feature embedding of latent space (a vector of d dimensions) via an embedding function $\boldsymbol{v} = f_{emb}(a)$. Therefore, each feature shares the same embedding in different data samples. This is crucial for FM to effectively consider feature interactions in recommender systems where the features are usually sparse.

FM models feature interactions through the dot product of corresponding features. Considering pairwise feature interactions as an example, FM models the interaction of feature i and feature j (i.e., $\langle i, j \rangle$) as $u_{ij} = \boldsymbol{v}_i \odot \boldsymbol{v}_j w_i w_j$, where $\odot$ is the dot product of two vectors and u_{ij} is the result of the feature interaction modeling. Then, the recommendation predictions are generated via aggregating the modeling results of all possible pairwise feature interactions and trainable bias values of all features in the data sample. Equation (2.1) shows the prediction function of FM.

$$y' = \theta_0 + \sum_{i=1}^{p} \theta_i w_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} \boldsymbol{v}_i \odot \boldsymbol{v}_j w_i w_j, \tag{2.1}$$

where θ are parameters, p is the number of features in the input and y' is the predicted output.

With the success of FM, many recommender systems based on FM have been developed. Xiao et al. [4] investigate one of the main limitations of FM that assigns the same weight to different features, while different feature interactions may have different levels of contribution to the final predictions. Accordingly, they propose an attentional factorization machine (AFM) that assigns each feature interaction a different weight, which is calculated through an attention mechanism [115]. Another limitation of FM is that the aggregation of interaction modeling results in a linear process, which limits the potential of FM to learn complicated (non-linear) relations between feature interactions in jointly determining the final predictions. To address this, He and Chua [108] propose a neural factorization machine (NFM) that replaces the linear aggregation method (sum) of FM with a deep neural network (DNN). NFM combines FM and deep neural networks, which achieves state-of-the-art recommendation accuracy. Chen et al. [116] investigate the suboptimal nature of the negative sampling strategy, such as the bias issue and the highly sensitive to the sampling distribution issue. They propose a non-negative sampling-based FM algorithm that is more efficient and effective in modeling feature interactions than FM. Yu et al. [117] study the impact of the same feature in different situations (different data samples) and propose an input-aware neural network upon a factorization machine model so that the same feature will have different impacts in different data samples. Lu et al. [118] propose a dual input-aware neural network that learns feature representations at bit-wise and vector-wise levels simultaneously for fine-grained interaction analysis. Then the reweighted feature representations are used for recommendation prediction by FM. Chen et al. [119] introduce two types of features: static features that may not change with time (e.g., user genders), and dynamic features that may change with time (e.g., user interacted items). They especially investigate the effect of sequential dependencies between dynamic features to extend factorization machines to sequential recommendation tasks.

Theoretically, FM can capture feature interactions higher than 2nd orders (i.e., high-order feature interactions). However, its time complexity grows exponentially with the order increase since it needs to model all possible feature interactions. Therefore, it is prohibitive to model all the high-order feature interactions in FM. To address the problem, HOFM [120] combines FM and a ANOVA kernel [121] for efficiently computing feature interactions in FM. HOFM can efficiently consider arbitrary-order feature interactions while maintaining similar accuracy to FM. Liu et al. [17] calculate a gate for each

high-order and low-order feature interaction to decide which feature interactions are important to be eligible for participating in the final prediction. Cheng et al. [109] propose AFN that leverages a logarithmic transformation layer to perform adaptive learning on arbitrary-order feature interactions. In the last section, we discussed the DeepFM model that performs implicit feature interaction modeling for higher-order feature interaction. To model these feature interactions explicitly, Lian et al. [16] extend the DeepFM model by leveraging a cross network [113] to explicitly model bounded-order feature interactions (e.g., up to 4-order) instead of a DNN in DeepFM. Thus, the cross network keeps the benefit of DNNs, while it is more effective in learning the high-order feature interactions of specific orders. Zhang et al. [122] consider filed information of features and propose explicitly learning filed relations in the DeepFM framework for more accurate recommendation.

FM-based models leverage the dot product as the interaction modeling function. However, feature interactions are inherently complicated. Attention-based methods explore to leverage the attention mechanism (in the next section) to better capture feature interaction information.

2.2.3.2 Attention-based Feature Interaction Modeling

Based on psychology, the attention mechanism is a behavioral and cognitive process where people concentrate on some information while ignoring other perceivable information [123]. This mechanism has been borrowed by the machine learning field and indicates that a machine learning model gives more weight (pay more attention) to some important parts of the input to perform predictions. It has been first proposed by Bahdanau et al. [115] to improve the long short-term memory model (LSTM) in neural language processing tasks and achieves surprisingly good performance. An attention model has been proposed to capture the relative importance of input entities (tokens) to the output entities. Due to the success of the attention mechanism, it has rapidly been applied in various neural language processing tasks [124–128] and computer vision tasks [129–132].

The attention model has great potential to learn feature interactions since the feature interaction modeling also tries to analyze the relations between features. The general framework of attention-based feature interaction modeling first learns the attention weight between each pair of features. Then the weight will be normalized as the contributions of each interaction to the final predictions. Equation (2.2) shows how the attention model learn feature interactions.

$$
\begin{aligned}
a_{ij}^{'} &= f_{attn}(\boldsymbol{v}_i x_i, \boldsymbol{v}_j x_j), \\
a_{ij} &= \frac{\exp(a_{ij}^{'})}{\sum_{i \neq j} \exp(a_{ij}^{'})},
\end{aligned}
\tag{2.2}
$$

where f_{attn} learns the unnormalized attention weight between each pair of features and a_{ij} is the normalized attention value. The attention values are not used for feature interaction modeling for the first attempt but are still used as weights while aggregating the feature interaction modeling results. Xiao et al. [4], Guo et al. [133] leverage the attention values as the weights of feature interactions in factorization machines. Zhang et al. [122] propose a field attention algorithm for calculating the importance of the same features in different data samples. Tao et al. [134] extend AFM by leveraging the attention algorithm from calculating the weight of pairwise feature interactions to the weight of high-order feature interactions.

With the progress of the Transformer algorithm [135], the self-attention mechanism solely as the relational reasoning method achieves great success [136–139]. Yun et al. [140] apply self-attention to model feature interactions and then use a vanilla attention method to aggregate the feature interaction modeling results at the final step of the prediction. Song et al. [11] leverage a multi-head self-attentive neural network [135] for pairwise feature interaction modeling. They claim that high-order feature interactions can be modeled by stacking the self-attentive neural network layers. Li et al. [103] also leverage the multi-head self-attentive neural network for feature interaction modeling but apply it in graphs that can also be regarded as edge modeling. Xie et al. [141] consider both explicit/implicit positive/negative feedback and leverage multiple multi-head self-attentive neural networks to model feature interactions in each kind of feedback, respectively. Then all the modeling results are aggregated through vanilla attention models. Xiao et al. [142] explore the multiple interests of users and propose DMIN that contains two multi-head self-attentive neural networks. One self-attentive neural network is used for feature interaction modeling, and another for extracting users' multiple interests. On top of the multi-head self-attentive neural networks, Li et al. [143] efficiently model high-order feature interactions through a proposed attentional aggregation strategy that simultaneously provides interpretability.

Compared to factorization machine-based feature interaction modeling methods that model feature interactions via dot product, attention-based methods use more sophisticated self-attention mechanisms to better capture the feature interaction information. In practice, such sophisticated interaction modeling is proven more effective and has achieved state-of-the-art performance in recent years.

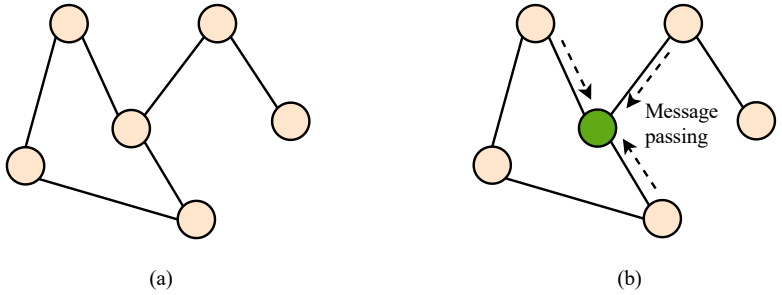


FIGURE 2.5: (a) A graph with nodes and edges. (b) How GNNs generate updated node embeddings (nodes in green) by the message passing technique.

2.2.3.3 Graph Neural Network-based Feature Interaction Modeling

Graph neural networks (GNNs) facilitate learning about entities and their relations [12, 13, 144, 145]. In this part, we give a brief introduction to how GNNs work and focus on their applications in feature interaction modeling. In the next section, we give a more detailed discussion of GNNs and their related applications.

A GNN is applied in the data form of a graph $G\langle\mathcal{V},\mathcal{E}\rangle$ consisting of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. Figure 2.5 (a) illustrates a graph representation. A node will first be represented as a node embedding, which is a vector that each element is either the attributes of the node or a randomly initialized parameter. The core of GNNs is the message passing technique, which aggregates the neighbor information for each node to update the node embedding. Figure 2.5 (b) illustrates how the message passing technique works to update node embeddings. Specifically, for each node $v \in \mathcal{V}$, a GNN first aggregates the neighbor information:

$$b_v = f_{aggr}(\{h_u : u \in \mathcal{N}(v)\}), \tag{2.3}$$

where $\mathcal{N}(v)$ are the neighbors of node v, h_u is the node u's embedding, $f_{aggr}(\cdot)$ is an aggregation function (e.g., element-wise mean).

Then, the embedding of node v will be updated though a update function f_{update}:

$$h_v^{'} = f_{update}(h_v, b_v), \tag{2.4}$$

where $h_v^{'}$ is the updated embedding of node v.

Leveraging GNNs for feature interaction modeling is a new but promising research direction in recommender systems. Generally, GNN-based methods regard each data sample with features as a graph: each feature being a node in the graph and each feature interaction being the edge between the corresponding nodes. Therefore, each data sample can

be regarded as a graph (i.e., a feature graph). In this way, the advantages of GNNs can be leveraged for feature interaction modeling by learning the feature graph, as discussed below.

First, the graph is a flexible structure and does not constraint data representation. For example, in neural language processing tasks, tokens are stored in vectors, e.g., each sentence is a vector, with each token represented as an ID and takes one element of a vector. On the other hand, in computer vision tasks, images are stored as a matrix, where each element is a pixel. As a result, the fixed-size vector or matrix representation can limit the flexibility of the data formatting. For instance, sentences may have different lengths. To represent sentences of different lengths uniformly, researchers have to predefine the largest length (e.g., 100) of tokens for each sentence. Then, the sentences containing more than 100 tokens have to be truncated, and those containing less than 100 tokens have to be filled up with placeholder symbols. Images may also have different sizes. The larger images may be truncated to a fixed size to be stored as matrices format uniformly. Inevitably, this alteration will result in information loss or introduce some noise.

A graph is a data structure in which each graph can contain different numbers of nodes and edges. Therefore, even if the input data samples have a different format, they can be easily represented in graph form, e.g., each image is a graph where each node is a pixel, and neighbor pixels are linked by an edge. For recommender systems considering feature and feature interactions, each data sample in recommender systems may contain different numbers of features. For example, a movie may contain different numbers of casts, genres, and directors. A graph, therefore, is a more suitable data format to represent such data samples containing different numbers of features by representing each feature as a node in a graph.

Second, the graph is a perfect data structure to explicitly represent features and feature interactions. Previous data representation methods such as vectors can only represent feature information. Feature interactions must be constructed in the recommender systems to model feature interactions. For example, FM-based models construct all the feature interactions and then model them. Graphs provide a uniform structure to represent feature interactions explicitly. The feature interactions to be modeled are already stored in a feature graph. Researchers can design new feature interaction-based recommender systems by simply proposing their variations of GNNs without worrying about feature interaction generation.

The third and most important advantage of using GNNs for feature interaction modeling is that GNNs are powerful in modeling the relations between entities (nodes) [12, 13].

This is the main reason behind the success and great adoption of GNNs in many problems, such as machine learning-based molecular property prediction [146, 147], and object relation learning physical systems [148, 149]. Presenting feature interactions as edges perfectly leverage the relational reasoning ability of GNNs for feature interaction modeling and aggregation. Specifically, the message passing procedure in GNNs can be used as the feature interaction modeling procedure. The updated node embeddings contain the interaction modeling results and can be leveraged for recommendation prediction.

However, how to effectively leverage GNNs for feature interaction modeling is still an open question. In existing studies, Li et al. [103] is the first work to leverage GNNs for feature interaction modeling in recommender systems. They propose Fi-GNN, which represents each feature as a node, and all pairwise feature interactions as edges. Therefore, each data sample forms a complete graph. Then, they combine the GNNs and the self-attention mechanism and modify the message passing function as a multi-head attentive neural network. Fi-GNN achieves better recommendation performance than existing competitive baselines. However, Fi-GNN has a drawback: they construct a complete graph for each data sample. This is suboptimal solution because some feature interactions may not be relevant to the recommendation result, and taking them into account may introduce noise and decrease the accuracy [150, 151]. We propose L_0-SIGN that leverages a link prediction technique with an L_0 activation regularization to detect beneficial feature interactions for recommendation. We demonstrate this work in Chapter 4.

Li et al. [152] propose a feature interaction detection method under the graph neural network framework as a link prediction. They first leverage a neural network to capture the link information (the probability of having a link) between every pair of features in a graph. Then, they propose a sampling strategy that samples the same number of edges for each graph as a set of beneficial feature interactions based on the probabilities. Xie et al. [153] propose an inductive feature interaction searching strategy to detect beneficial high-order feature interactions. Their search strategy is based on the intuition that useful high-order feature interactions come from useful low-order feature interactions. Then they represent features as feature graphs, and perform the useful feature interaction detection as an edge search task. To find useful high-order feature interactions, they predict multiple edge adjacency matrix (i.e., adjacency tensor), with each newly predicted adjacency matrix (higher-order feature interactions) detected based on previously predicted adjacency matrices (lower-order feature interactions). Finally, the detected feature interactions are modeled through a stacked GNN.

The above proposed GNN-based feature interaction modeling methods cannot capture

high-order feature interactions, impose significant computational and memory complexity (e.g., the detected adjacency tensor can be very sparse). In this thesis, we leverage the hypergraph form representations (a generalized graph form representation that each edge can link to more than two nodes) to represent high-order feature interactions more efficiently and more straightforward. Then we propose a method to directly generate high-order feature interactions, which is more time efficiency founded by using the deep mutual information maximization [154] technique. We will detail this method in Chapter 5.

2.3 Graph Neural Networks

In the last section, we discuss studies that leverage graph neural networks for feature interaction modeling in recommender systems. This section provides a more detailed discussion about studies of graph neural networks, including different graph neural network algorithms, and a brief introduction to their applications in other machine learning domains. Then, we will talk about neural graph matching, a specific GNN application domain that closely relates to our work in Chapter 6. After that, we will talk about the applications of GNNs in recommender systems other than for feature interaction modeling.

2.3.1 Graph Neural Networks

In recent years, graph neural networks (GNNs) are invented to model the rapid growth of graph data, such as social networks, molecules, and traffic [155]. GNNs can be divided into two categories: spectral models [12, 156] and spatial models [157–159]. Their graph learning strategies are based on the same principle: each node in a graph aggregates neighborhood information to capture the neighbor relations and leverage the information to update the node representation. Specifically, each node will have a node representation (e.g., node embedding). Then, each node aggregates the vector representations of its neighbors (e.g., element-wise mean) and updates the node representation through a function considering the aggregated representation of the neighbors and the node's original representation. The updated node representations are called node patch representations. This procedure can be stacked multiple times to catch distant neighbor information. If the graph representation is required (e.g., in graph classification tasks), the node patch representations will be further aggregated into a single vector as the graph representation for further downstream tasks.

GNNs have been successfully applied in many research domains. In chemistry, Duvenaud et al. [146] generalize hand-crafted molecular features by operating directly on molecular fingerprints using graph convolutional networks. Gilmer et al. [147] propose the Message Passing Neural Networks (MPNNs), a general GNN-based molecule learning framework for molecular property prediction. In physical system research, the numbers and positions of objects are flexible. Therefore, a physical system is suitable to be represented as a graph. Chang et al. [148] represent a physical system as a graph and use GNNs to learn object relations in dynamic physical systems. Battaglia et al. [149] propose an interaction network that regards a physical system as a graph. In the graph, nodes are objects, and edges represent their relations. The interaction network can model complex systems interacting in dynamic situations.

In computer vision, an important task is to extract objects and their relations in images (the extracted representation is a scene graph). Xu et al. [160] leverage the message passing strategy to iteratively model the generated scene graphs. Yang et al. [161] detect objects and their relations in an image through a proposed relational proposal network and learn the related information through a proposed attentional GNN to generate scene graphs. Li et al. [162] focus on the efficiency of scene graph generation. They first generate a set of subgraphs through a bottom-up clustering method. Then, they propose a spatial-weighted message passing method to efficiently model the relations within each subgraph while maintaining their spatial relations in the entire scene graph. Action recognition in videos is another computer vision task that can be effectively handled by GNNs. Yan et al. [163] propose a spatial-temporal GNN, a general framework to learn both the spatial and temporal patterns in video data for action recognition, which is more expressive than traditional non-GNN methods. It does not rely on hand-craft parts or traversal rules. Shi et al. [164] propose a two-stream adaptive GNN that automatically learns the topology of the skeleton graph and can further capture the second-order information (e.g., directions of bones) that has not been considered in previous methods.

In neural language processing, the most intuitive application is text classification. The whole text dataset can be regarded as a graph. For example, in a citation network, each text is a node, with the tokens in texts being node features, and their citation relationships are edges [165]. Then, GNNs can be applied to perform text classification as a graph node classification task [12, 157, 158]. An alternative option is to consider each text as a graph. In the graph, nodes are tokens, and edges are their relations. Then, the text classification predictions are the graph classification results. Marcheggiani and Titov [166] combine the syntactic dependency trees of texts and text graphs that consider the dependency relations as edges. They propose a syntactic GNN to model the text graph with syntactic dependency-based edges to get the latent text representations. Bastings et al. [167] use a similar idea and apply syntactic GNN to the

neural language translation task. Graph-to-sequence learning is to generate text through graph structure semantic information (i.e., abstract meaning representation). Therefore, it is also a promising neural language processing task to apply GNNs. Song et al. [168] leverage GNNs to model the graph-structure abstract meaning representation to form a GNN-LSTM framework for graph-to-sequence learning, which is more effective in capturing the graph-based abstract meaning than the previous LSTM-LSTM framework. Beck et al. [169] investigate the issue that semantic graphs always have long dependencies between nodes. They levearege the Gated Graph Neural Network [170] to capture longer distance semantic relations in graph-to-sequence learning. The above GNN-based methods achieve state-of-the-art performance in their corresponding research domains, proving that GNN is a promising framework for modeling various relations.

2.3.2 Neural Graph Matching

Graph matching aims to discover the similarity between two graph representations, and it is a long-standing research topic in computer science, such as similarity search in process model repositories [171] and chemical structure search [172]. We leverage the neural graph matching technique in Chapter 6 for matching users and items in recommender systems. In this section, we introduce studies related to neural graph matching.

Traditional graph matching algorithms are based on heuristic rules, such as minimal graph edit distance [173, 174], or graph kernel methods, such as random walks inside graphs [175, 176] and graph sub-structures [177, 178].

With the development of GNNs, recent studies explore neural-based graph matching. Bai et al. [179] leverage a GNN to model two graphs and generate graph-level embeddings, respectively. Then, they devise a pairwise node comparison method to perform the graph matching with fine-grained node-level information. Li et al. [180] capture node-level information for graph matching, and simultaneously perform graph learning and node-level graph matching in a fused model in a Siamese network framework. Bromley et al. [181] model two graphs using the same GNN, and perform node-level matching at the message passing stage of graph learning. In this way, the graph learning procedure learns both the graph information, and the node-level matching information for the final graph matching.

Recently, neural graph matching methods have been used for image matching in the computer vision domain. Sarlin et al. [182] extract image features to construct a feature graph. They propose SuperGlue to match two feature graphs by a two-stage procedure.

In the first stage, SuperGlue maps graphs into graph vectors containing keypoint information through an attentional graph neural network. In the second stage, it learns a score matrix showing the feature matching results through multiple proposed optimal matching layers. As a result, SuperGlue achieves end-to-end image feature matching compared to traditional hand-designed heuristic matching methods. Jiang et al. [183] focus on the problem that the smoothing-based graph convolutional layers in traditional GNNs may lose discriminatory information of graph nodes. They propose a Laplacian sharpening convolutional module and a constraint regularized loss for more discriminative node embedding learning for better image feature matching using GNNs.

However, the neural graph matching methods have not yet been fully explored in recommender systems. In this thesis, we find that the graph matching procedure perfectly fits the preference matching procedure in recommender systems. Specifically, let's consider each user (or item) as a graph constructed by their corresponding attributes. The preference matching procedure is exactly the matching procedure between a user graph and an item graph. The difference is that the traditional procedure matches whether two graphs have a similar topology structure, while the graph matching in our scenario matches users' interests in item attributes. Besides, the graph structure representations of users and items enable better user/item profile learning by GNN models. We will detail this work in Chapter 6. This is the first work to represent each user and item in a graph form and leverage the framework of neural graph matching for preference matching.

2.3.3 Other GNN Applications in Recommender Systems

In Recommender Systems, GNNs have been applied in many other graph analysis tasks in addition to modeling feature interactions to facilitate recommendation predictions. Here we focus on several common scenarios where existing works leverage GNNs in recommender systems. In this section, we discuss how GNNs can be used for user-item graph learning [184–188], social relation learning [189–193], and knowledge graph learning [194–198, 198, 199].

User-item interactions are one of the most key relations in recommender systems. Therefore, it is natural to consider leveraging the relational reasoning ability of GNNs to infer the user-item interactions. One intuitive method is constructing a user-item bipartite graph where an edge exists between a user and an item. van den Berg et al. [184] construct a user-item bipartite graph where a user and an item are linked by an edge only if the user interacted with the item. Therefore, the recommendation predictions are conducted via link prediction. They propose a GNN-based auto-encoder framework to infer

the link information. However, the bipartite graph only consists of direct interaction information, but omits the remote interaction information (i.e., high-order connectivity information). Therefore, one method is to construct a user-item bipartite graph, where an edge exists between users and items. Wang et al. [185] use a similar graph construction strategy but regard the graph as a typical graph. By staking the message passing procedure of GNNs, their proposed algorithm can capture a better collaborative signal to achieve better prediction performance.

Wang et al. [186] focus on the different intents of user-item interactions (e.g., interests, shop for others). They propose a disentangled graph collaborative filtering methods that learn disentangled representations for different intents. As a result, interactions of different types of intents will be treated differently, which yields more fine-grained user-item interaction analysis. He et al. [187] pay attention to designing a GNN framework that is more suitable for user-item interaction modeling for recommendation. They propose LightGCN, which simplifies the typical GNN by removing the transformation and the nonlinearity components. As a result, LightGCN is more efficient in learning user-item graphs and achieves better recommendation prediction performance than traditional GNNs. On top of LightGCN, Wu et al. [188] leverage self-supervised contrastive learning methods to achieve more robust predictions, further improving the recommendation prediction accuracy.

Social networks reveal the relationships between users (e.g., friends, fans). In social recommendation, social networks are crucial information for finding common preferences of users for recommendation prediction [200, 201]. For example, we can assume users have similar interests if they are friends. Then, in collaborative filtering, these users can be considered similar users and perform recommendation predictions based on the user similarity.

A common way to represent a social network is to represent it as a graph. Users in the network are nodes, and their relationships are edges. If user side-information (e.g., age, gender) is known, initial user representation can be users' features. Then, node (user) patch representations can be learned through standard GNNs. Fan et al. [189] consider the connection and the differences between users involved in the social network graph and the user-item interaction graph, and propose a GNN-based framework, GraphRec, for social recommendation that coherently models two graphs. Wu et al. [190] explore the recursive diffusion process (close neighbors may influence each other, and they will further diffuse this influence to other neighbors) in social networks and propose a deep influence propagation model that captures the recursive social diffusion process for social recommendation. Wu et al. [191] also jointly learn a social network graph and a user-item graph, but focus on the different weights that different friends can

influence and the dynamic pattern of social effects. They propose dual graph attention networks to tackle these problems and achieve significant performance improvement in social recommendation. Huang et al. [192] leverage GNNs to model an item-item graph that involves inter-dependent knowledge across items to facilitate the item representation learning. Yu et al. [193] focus on the fact that user relations may not be pairwise. They propose to leverage hypergraphs to represent the high-order user relations (e.g., a group of people all have similar interests). Then, they integrate self-supervised mutual information maximization methods into GNN learning to effectively learn the different types of high-order relations.

Another graph-based information usually used in recommender systems is the knowledge graph (KG). A knowledge graph is a general graph-based representation of different attributes and relations of data. For example, a movie information knowledge graph may contain a node set that each node is one of the attributes of movies (e.g., a kind of genre, a director, a cast), and an edge set that each edge is a kind of relations between the nodes (e.g., is the director of, is the gene of). By learning the knowledge graph, useful attribute relation information among all movies can be extracted and stored in the learned node patch representations and be used for recommendation prediction.

In existing KG-based recommender systems, Wang et al. [194] construct items and their corresponding side-information as a knowledge graph. They propose RippleNet, an end-to-end model that performs knowledge graph learning and recommendation prediction simultaneously. The prediction is the multiplication of the representations (vectors) of a user and an item. Wang et al. [195] explore whether the same relation in the knowledge graph may have different importance to different users. Therefore, each user is first assigned a real-valued weight to the relations through a function considering user information. Then they learn the knowledge graph through a knowledge-aware GNN. Wang et al. [196] construct the knowledge graph by considering users, items, and item attributes as nodes. User-item interaction and the relationship between items and item attributes are edges (the collaborative knowledge graph). Then the node embeddings are learned through a proposed GNN-based attentive embedding propagation layer. Yang and Dong [197] propose hierarchical attention that captures high-order connectivity between nodes in a heterogeneous knowledge graph. Chen et al. [198] propose semi-supervised graph attention networks that incorporate rich unsupervised information in the knowledge graph for node embedding learning. Cao et al. [199] focus on the drawbacks of knowledge graphs that some relations may not always be correct (e.g., some relations and entities may be missing). They propose a unified model that jointly performs knowledge graph completion and recommendation model learning to simultaneously achieve more accurate recommendation prediction and get in a better knowledge

graph completion result. The knowledge graph merely contains attribute-attribute relations so that all the knowledge graph-based recommender systems do not consider feature interactions.

2.4 Techniques for Feature Interaction Detection

Our second research question explores detecting beneficial feature interactions for recommendation. In this section, we discuss the related techniques conducted in our works, L_0 regularization and mutual information maximization, for the detection. Note that these techniques do not aim for feature interaction detection in other works. We perform the detection by leveraging these techniques in novel ways.

2.4.1 L_0 Regularization

L_0 regularization is an essential technique in sparse approximations [202] that penalizes non-zero parameters. Formally, the L_0 regularization has the form $\|\boldsymbol{\theta}\|_0 = \sum_{j=1}^{|\boldsymbol{\theta}|} \mathbb{I}[\theta_j \neq 0]$, where θ are the parameters. However, optimizing an objective function with an L_0 regularization is considered an NP-hard optimization problem [203]. Various studies explore to solve the problem. Greedy methods [204] are fast in performing the optimization but have no guarantee of the optimizing results. Relaxation methods [205, 206] have better performance but are more time-consuming. Iterative thresholding methods [202, 207, 208] guarantee the convergence of the optimization procedure and achieve less cost.

Due to the non-differentiable nature of L_0 regularization, it does not attract attention previously in machine learning domains until Louizos et al. [151] solved the non-differentiable problem. They leverage the reparameterization trick to represent each parameter as the product of a real value (parameter value) and a binary value (a gate value). Then, the L_0 regularization is used to minimize the number of gates having a value of 1. Finally, they propose a hard concrete distribution to approximate the discrete binary gate values by continuous values that are differentiable and can be optimized using gradient descent algorithms. Based on this work, some studies have been proposed to compress neural networks using L_0 regularization.

Tsang et al. [7] apply L_0 regularization on fully connected multilayer neural networks to sparsify the connections between layers. In this way, the sparse neural networks are more time and space efficient and provide the potential for interpretability. Shi et al. [209] leverage L_0 regularization to prune the dimension of the latent representation

vectors of the variational autoencoders [210] to find the most suitable dimension of the latent representation vectors. We utilize L_0 regularization to detect beneficial feature interactions and leverage only the beneficial feature interactions for recommendation. Besides, we are the first work to theoretically prove the ability of the L_0 regularization to achieve feature interaction detection founded on the information bottleneck principle. We will detail this work in Chapter 4.

It is worth noting that L_1 regularization (or Lasso) is another regularization method that is also able to sparsify models [104]. Unlike L_0 regularization in recent machine learning studies that minimizes the number of parameter gates, L_1 regulation directly minimizes parameters. Many studies leverage L_1 regularization to achieve model sparsity [211–214]. However, the direct parameter minimization makes a mix of parameter estimation and parameter sparsifying [151]. This may result in a challenging situation for the model to simultaneously achieve good parameter estimation and sparsifying during training. For example, a parameter (or a feature) shows a strong correlation (important) to the true output. This parameter should be significant in parameter estimation, while it should be small in the point of view of parameter sparsifying by the L_1 regularization. L_0 regularization solves this problem by achieving the sparsity goal by a gate parameter for each learnable parameter.

2.4.2 Mutual Information Maximization

In information theory, mutual information (MI) is fundamental in measuring the relations of two random variables. The estimation of mutual information is notorious because of the high computational complexity in a continuously high-dimensional space. Therefore, MI has not been widely adopted in deep learning until Belghazi et al. [215] proposed a efficient mutual information neural estimation (MINE) method that leverages a neural network to distinguish the joint distribution of two variables from their marginal distribution. As a result, MI can be efficiently estimated through training a neural network-based classifier (or discriminator).

Mutual information maximization (Infomax) [216, 217], which maximizes the MI between input and output, is one of the most critical techniques in unsupervised feature representation learning. Infomax heavily relies on mutual information estimation, which has not caught much attention in deep learning. With the proposal of MINE, Hjelm et al. [154] successfully perform image representation learning using the Infomax technique by proposing Deep Infomax (DIM) based on MINE. Specifically, DIM maximizes the mutual information (MI) between an image's local and global representations. DIM learns image representations in an unsupervised setting but gains similar (even better)

performance in downstream tasks (e.g., image classification) compared to supervised learning. After that, Veličković et al. [218] leverage DIM to perform unsupervised graph representation learning and propose the deep graph Infomax method (DGI) to maximize the MI between node representations and graph representations. In addition to node representation, Sun et al. [219] perform graph representation learning that considers substructure information from graphs (e.g., edges, triangles) for better representation learning results. Li et al. [220] use DGI to learn graph information and summarize it into a compact form (e.g., graph pooling) to achieve hierarchical graph representation learning. They propose a vertex Infomax pooling method that is used for both representation learning and important node searching. Pang et al. [145] focus on better graph pooling performance, and propose a coarsened graph Infomax pooling method to efficiently capture the global dependencies between graphs before and after pooling (which previous methods are unable due to the high computational complexity).

Recommender systems also require representation learning (e.g., feature representation learning, item/user representation learning). For the first time, we perform arbitrary order beneficial feature interaction detection through Infomax algorithms for recommendation. Our work achieves several unprecedented explorations. First, we are the first to achieve feature interaction detection using Infomax algorithms. Second, we are the first to perform Infomax on hypergraphs with more complex relationships between nodes and different graph learning strategies. Third, since recommender systems usually contain label information (e.g., explicit or implicit feedback), we perform Infomax in a supervised setting to leverage the label information to achieve better Infomax performance. Forth, our proposed Infomax method is theoretically guided by our proposed properties of beneficial arbitrary order feature interactions in information theory. We will detail this work in Chapter 5.

Contrastive learning is a popular research direction recently that is close to DIM. The procedure of contrastive learning can be regarded as maximizing the mutual information between two views of the same variable (e.g., image) [221–223]. The view of the variable is obtained via data augmentation algorithms (e.g., rotation, color-changing). By doing so, the essential information that is invariant to the augmentation (usually being regarded as beneficial for downstream tasks) will be captured in the learned representations, and other irrelevant information will be discarded. For example, given an image of a cat, a person can recognize that it is a cat even if the image is rotated or the color becomes black and white. There are some contrastive learning works that are close to our fourth work (Chapter 5). Tian et al. [224] define the properties of good views in contrastive learning. They hypothesize that each view contains all the information of an input variable about the output. In Chapter 5, we define the properties of beneficial feature interactions. However, the hypothesis in [224] is not suitable for our scenarios.

Each beneficial feature interactions in recommender systems usually contain only a tiny part of the output information, and the final predictions are made based on all the interaction information. Therefore, our proposed properties of beneficial feature interactions are based on the hypothesis that each interaction only contains part of the information about the output. Khosla et al. [225] extend contrastive learning to a supervised setting that maximizes the mutual information between the views of the same variable and the views of all the variables of the same label class. Our s-Infomax is similar to this work but applies the supervised learning between local and global representations.

2.5 Summary

In this chapter, we review existing works along with the techniques from three broad types of recommender systems to our focused feature interaction modeling-based recommender systems. Then, we demonstrate how feature interactions can be leveraged for recommendation. We discuss three techniques for explicit feature interaction modeling, including our focused graph neural network-based feature interaction modeling. Next, we talk about general graph neural network techniques and how they are leveraged in recommender systems besides feature interaction modeling. Finally, we review existing works about L_0 regularization and mutual information maximization, which are essential techniques in our works for beneficial feature interaction detection.

Chapter 3

Considering Feature Interactions in Matrix Factorization

This chapter is based on the paper that has been published in the International Joint Conference on Neural Networks (IJCNN) 2019: ***MMF: Attribute Interpretable Collaborative Filtering****. Yixin Su, Sarah Erfani, Rui Zhang.*

3.1 Introduction

In this chapter, we investigate the first research question of how to perform explicit feature interaction modeling in matrix factorization (MF) to improve its performance in the recommendation. As discussed in Chapter 2, MF is one of the most common collaborative filtering techniques, whose main idea is to learn user latent vectors and item latent vectors, so that the inner product of the two vectors can approximate the original matrix with the minimal approximation error.

MF has advantages of simplicity and performing well in recommender systems [226, 227]. However, MF cannot consider feature interactions. As a result, it suffers from two limitations. Firstly, its predictions do not provide interpretability, which is showed to be crucial for recommendation systems because interpretations can significantly improve users' acceptance rate [6]. Secondly, if only few ratings are available for some items (a.k.a. item cold-start problem), the prediction accuracy of MF is unsatisfactory [3].

There are other techniques such as knowledge graph-based recommender systems that consider features and feature interactions naturally [196, 199]. However, these methods require the knowledge graphs to be available. MF is a more general framework that can be embedded in many recommendation techniques. For example, in sequence or session

based recommender systems, the final user representations and item representations will be modeled as a MF (or deep MF [85]) to get the final prediction [10, 111, 228]. Enabling MF the ability to model feature interactions provides the potential for these models to easily consider feature interactions by simply replacing MF in the final step with an MF extension that can consider feature interactions (e.g., our proposed model).

Existing work on enhancing MF mainly focuses on a single aspect, such as improving prediction accuracy or one of the limitations described above. For example, Fuzheng [229] leveraged knowledge information in items to improve the prediction accuracy, but neglect interpretability. Hong [230] utilized topic modeling methods to enable matrix factorization interpretability, but the interpretations are limited and the prediction accuracy improvement is not significant.

In order to overcome the two limitations and improve the prediction accuracy simultaneously, we propose a novel recommendation model called *Multi-Matrix Factorization* (MMF). Our model aggregates the *attribute* ratings, where the attributes can be regarded as shared features or characters in items, into item ratings. Specifically, the attribute ratings can be calculated by the inner product of the user latent vector and the attribute latent vector learned through MMF. Further, we integrate two weights to strengthen MMF's performance on both the prediction accuracy and the interpretability, which are the user's preference weight on different attributes and the attribute performance weight on different items.

Figure 3.1 illustrates the prediction procedure of MMF on a movie recommendation example. We can see that after training, MMF generates several attribute rating matrices that each matrix represents one type of attribute. Each value in the matrices is an attribute rating a user may give to an attribute. In the prediction procedure, we extract attributes from the movie and find the corresponding attribute ratings. Then, MMF combines the attribute ratings with the user preference and the attribute performance weight together to get the weighted attribute ratings. Finally, the movie rating can be regarded as the sum of all weighted attribute ratings.

The contribution of MMF is that it simultaneously achieves the following goals:

- Interpretability: MMF analyzes the contribution of each user-attribute interaction to the final predictions, such as which attribute in an item may attract a user most. We conduct comprehensive interpretability evaluations to show the effectiveness of the interpretations.
- Resolving the cold-start problem: MMF provides reasonable predictions in item cold-start cases. Items may share attributes. This connection allows the transfer of user-attribute ratings from existing items to a new item.

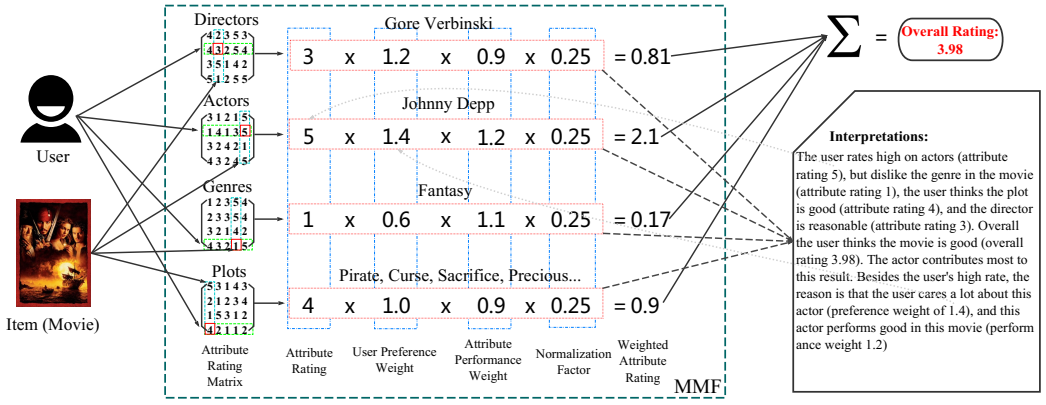


FIGURE 3.1: An illustration of MMF's prediction procedure on a movie recommendation example.

- Better prediction accuracy: MMF provides a novel way to learn fine-grained differences between items through user-attribute interactions. As a result, MMF achieves better prediction accuracy than state-of-the-art matrix factorization models and feature interaction-based recommendation models.

3.2 Preliminaries

3.2.1 Matrix Factorization

Suppose there is a rating matrix $\boldsymbol{Y} \in \mathbb{R}^{n^u \times n^v}$, where n^u is the number of users and n^v is the number of items, which stores the interaction history of uses and items. For example, in movie recommendations, $Y_{ij} = 5$ if the user i rated 5 to the movie j. The target of matrix factorization is to learn user latent vectors $\boldsymbol{U} \in \mathbb{R}^{n^u \times d}$, where d is the dimension of each vector, and item latent vectors $\boldsymbol{V} \in \mathbb{R}^{n^v \times d}$, so that their inner product can restore the values in $\boldsymbol{Y}$ through multiplying $\boldsymbol{U}$ and $\boldsymbol{V}$.

$$\boldsymbol{Y}^{'} = \boldsymbol{U}\boldsymbol{V}^{\top}, \tag{3.1}$$

where $\boldsymbol{Y}^{'}$ is the restored (predicted) rating matrix.

In the empirical risk minimization function, we use the mean squared error (MSE) to minimize the differences between the predicted rating matrix and the true rating matrix:

$$\mathcal{R}_{MF} = \mathcal{L}(\boldsymbol{Y}, \boldsymbol{Y}^{'}) + \lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2), \tag{3.2}$$

where $||\cdot||_F^2$ is the second normal form regularization function, $\mathcal{L}$ is the MSE loss function, and λ is the weight value of the regularization.

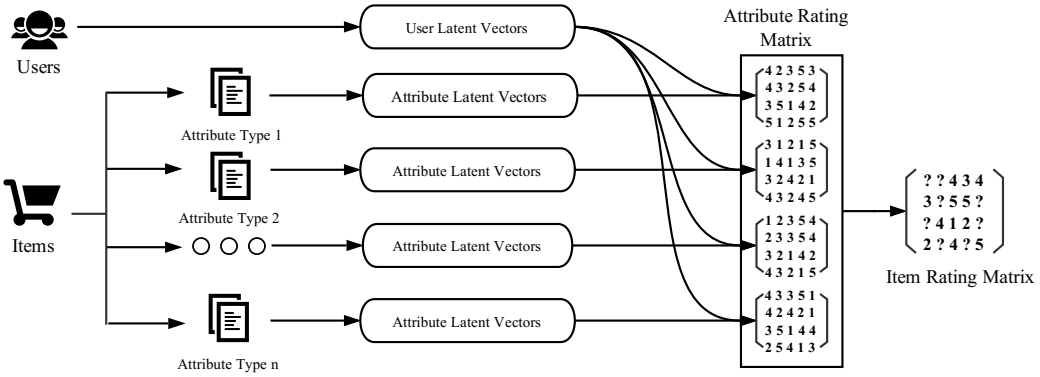


FIGURE 3.2: The overall structure of the MMF model.

3.3 Our Proposed Method

We first illustrate the structure of MMF and the learning procedure. Then, we discuss why MMF is superior to matrix factorization and factorization machine [2].

3.3.1 Multi-Matrix Factorization (MMF)

The main idea of MMF is to exploit the common attributes in items. Instead of linking items by user-item interactions, we build fine grained connections between items using attributes. The MMF model merges several matrix factorization models into a single one. Each matrix factorization can model the interactions between users and one type of attributes, indicating the personalized ratings of users on the attributes.

Figure 3.2 shows the overall structure of the MMF model. Specifically, MMF learns a user latent matrix $\boldsymbol{U}$ and several item attribute latent vectors $\{\boldsymbol{A}_k\}_{k=1:n}$ of n types of item attributes (e.g., directors, genres), where $\boldsymbol{A}_k \in \mathbb{R}^{a^k \times d}$ and a^k is the number of type k's attributes, through multiple matrix factorization models, with each model learns one type of attribute latent vectors:

$$\boldsymbol{R}_k = \boldsymbol{U}\boldsymbol{A}_k^\top, \tag{3.3}$$

where $\boldsymbol{R}_k$ is the attribute rating matrix, which stores the interaction modeling results between uses and the type k's attributes.

Next, because the same type of attributes is trained in a single matrix factorization model in MMF, it is inevitable that the attributes with the same type will affect each other, which is unfavourable for the prediction accuracy and the interpretation. For example, if a user likes most genres of movies but strongly dislikes *Thriller*, MMF may not give Thriller a very low attribute rating (which is expected), because the average attribute ratings of all the genres are high. We address this issue by giving each user a

preference weight ω on each attribute to show user's preference on different attributes. Higher ω value means higher user preference.

Finally, we cannot assume that one attribute performs equally in different items. To better differentiate the quality of one attribute in different items, we assign different weights θ, for attributes in different items, which means that one user may give different ratings for an attribute in different items.

The predicted ratings can be calculated by combining all components discussed above together. In addition, the ratings will be normalized by dividing with the number of attributes in the item to balance inequality of attribute number. The calculation formula is:

$$Y_{ij}^{'} = \frac{1}{|M_j|} \sum_{(k,m) \in M_j} \omega_{ikm} \theta_{jkm} (R_k)_{im}, \tag{3.4}$$

where M_j stores which attributes (the m's attribute of k's type) belong to the item j.

3.3.2 Empirical Risk Minimization Function

Although the MMF seems to be more complex than matrix factorization, the training process is not difficult. Similar to MF, we use the MSE loss function to measure the differences between predicted ratings and user's ratings. The empirical risk minimization function is:

$$\mathcal{R}_{MMF} = \mathcal{L}(\boldsymbol{Y}, \boldsymbol{Y}^{'}) + \lambda(||\boldsymbol{U}||_F^2 + \sum_{k=1}^{n} ||\boldsymbol{A}||_F^2). \tag{3.5}$$

3.4 Discussions

In this section, we discuss the relations of our MMF model with MF and factorization machine (FM) [2].

3.4.1 Relation to Matrix Factorization

MMF improves over matrix factorization from three aspects: 1) provide interpretable recommendation, 2) relief item cold-start problem, and 3) provide more accurate predictions.

3.4.1.1 Personalized Interpretable Recommendation

MMF provides personalized interpretations from two aspects: (i) users' judgment on attributes ($\boldsymbol{U}\boldsymbol{A}^\top$), (ii) users' personalized preference on each attribute (ω). As Figure 3.2 shown, the item latent vectors can be regarded as the weighted sum of attribute latent vectors. Therefore, items can be divided into several components with explicit meanings. Specifically, $\boldsymbol{U}\boldsymbol{A}^\top$ can be interpreted as general ratings showing how a user may like a type of attribute, which is similar to matrix factorization on items. ω will further differentiate the preference of users between different types of attributes. Detailed evaluation will be demonstrated in Section 3.5.5.

3.4.1.2 Item Cold-Start Problem

The main reason that matrix factorization method performs unsatisfactory on item cold-start situation is that if there is no rating history (e.g., a newly added item), the model cannot link it to previous items. MMF overcomes this defect efficiently. An item in MMF is represented as the weighted sum of the attribute latent vectors, who may appear in multiple items so that they can act as the connections. For example, when a new item is added into the system, the item may already contain attributes that appear in previous items. Attribute ratings that users give to previous items can be used as a vital clue to perform personalized item rating predictions. Therefore, as long as the new item contains attributes that have appeared in existing items, MMF is able to infer well item rating.

3.4.1.3 Prediction Accuracy

In addition to the above advantages, attributes extracted from items show the ability on improving prediction accuracy as well. Comparing to matrix factorization, the MMF model analyzes items in a fine grained point of view. It learns the similarities between the attributes rather than the items, which enables the model to learn more delicate differences between items. This helps improve the prediction accuracy. In extreme situations, if there is no common attribute between any two items, MMF is identical to matrix factorization. From Equation (3.4), we can see that if there is no common attribute, every item latent vector is composed by weights and attribute latent vectors without overlap, which reduces MMF to matrix factorization.

3.4.2 Relation to Factorization Machine

Factorization Machine [2] is a popular algorithm for feature interaction modeling. FM analyzes attributes by performing inner product among attributes, and the final predictions are the sum of all analysis results, which is similar to MMF. However, instead of aggregating attribute interactions in equal weight, MMF gives attributes interpretable weights (attribute performance weights and user preference weight) to improve both prediction accuracy and interpretability because that the weights can both be parameters to further differentiate attributes' contribution, and can act as informative interpretations (e.g., how attributes perform in different items). We experimentally prove it in Section 3.5.3.2. Meanwhile, MMF simultaneously learns item embeddings (weighted sum of all corresponding latent attribute vectors) during training, while FM does not. This makes MMF easy to further perform item level analysis directly, such as implementing neural collaborative filtering on users and items [108].

3.5 Experiments

We conduct empirical analysis to evaluate MMF. Then, we demonstrate the interpretability of MMF through a case study. Finally, performance of MMF on item cold-start situation and the influence of hyper parameters are evaluated.

3.5.1 Data Description

We use movie recommendation as a representative application in our experimental study due to two reasons. First, there are a few publicly available sources for movie data. Second, movie recommendation has been the focus of previous studies on collaborative filtering such as [231, 232]. We use MovieLens [233] and Netflix Prize [234] to evaluate our model, which are commonly used in movie recommendation researches.

3.5.1.1 Datasets

We use two public datasets: (i) MovieLens 100k (**100k**), (ii) MovieLens latest-small (**20m Light**). Meanwhile, we analyze the performance of MMF on different dataset densities, including rating data density and attribute density (will be illustrated in Section 3.5.1.3). Since the density in public datasets is uncontrollable, we further use three subsets from MovieLens and Netflix Prize Data sources. They are (iii) top movies from MovieLens 20m ranked by box-office (**Boxoffice**) with high rating data density, (iv)

Netflix 500, and (v) **Netflix 1000** datasets that randomly chose 500 and 1000 movies from Netflix Prize data source respectively, which both have high attribute densities.

In addition, to evaluate our model in item cold-start situation, we randomly pick 10% movies from all datasets, then regard ratings of the picked movies as test data and the rest as training data, which simulates the item cold-start situation.

3.5.1.2 Attribute Extraction

Four types of attributes are chosen: directors, casts, plots and genres. All attributes are extracted from IMDB website, which is already mapped into the MovieLens data source. We also aggregate Netflix Prize data source to the IMDB database by matching the name and year of the movies. In addition, we use topic modelling method, LDA [235], to transform poorly structured plot attribute into well-structured form called topics. Each movie contains one or several topics with the same structure as other attributes.

3.5.1.3 Dataset Statistics

Table 3.1 shows the statistic information of all datasets. The *Rate* column demonstrates the rating data density in datasets. The lower the rating data density, the less average rating of users. It is calculated by:

$$Rate = \frac{N_r}{N_p \times N_u}, \tag{3.6}$$

where N_r is the number of ratings; N_p is the number of movies and N_u is the number of users.

TABLE 3.1: Datasets statistics.

Dataset	Movies	Users	Ratings	
Name	Number	Number	Count	Rate
100k	1683	944	100000	0.063
20m Light	8788	671	98688	0.017
Boxoffice	206	3550	99326	0.136
Netflix 500	539	5893	100181	0.031
Netflix 1000	1061	17185	308544	0.017

TABLE 3.2: The attribute details of datasets.

Dataset	Directors	Casts	Genres	Dense Rate
100k	1138	1988	24	0.049
20m Light	3976	11632	25	0.083
Boxoffice	194	474	20	0.050
Netflix 500	28	1065	24	0.103
Netflix 1000	64	1837	27	0.094

Table 3.2 demonstrates the detailed attribute information of all datasets. Meanwhile, our experiments generate 50 topics for all datasets. Evaluation on different topic numbers will be demonstrated in Section 3.5.8.

Moreover, we present an attribute density (*Dense Rate*) to indicate the repetitive degree of attributes in all items. The Dense Rate calculation formula is:

$$DenseRate = \frac{1}{n} \sum_{i \in \{1..n\}} \frac{Ave_i}{N_i}, \tag{3.7}$$

where n is the number of attribute types (in our datasets, n is 4); Ave_i is the average number of i^{th} types of attribute that appear in one item; N_i is the total number of i^{th} types of attribute in datasets.

The dense rate is capable of representing the probability that each attribute appears in different items. The higher the attribute dense rate, the higher probability of two items in the dataset share common attributes.

3.5.2 Baseline Methods

Matrix factorization models and State-of-the-art attribute based factorization models are compared, including:

MF The matrix factorization model that has been discussed before.

PMF [78]: It uses probabilistic method to describe latent vectors. PMF is one of the most popular variations and usually get better result than MF on large, sparse and imbalance datasets.

BPMF [79]: It uses Bayesian treatment to train PMF with the help of Markov Chain Monte Carlo methods.

FM [2]: The factorization machine model that we have described in Section 2.2.3.1.

DeepCrossing [236]: It leverages residual deep neural network that automatically combines attributes. We implement this model that stacks three residual units with hidden dimensions 128, 128, 64, respectively.

3.5.3 Model Performance

We evaluate the perforamnce of our model using the root mean square error (RMSE). Note that the lower the RMSE value, the better the performance. In our experiments, the proportion of training set and test set are 8:2 for all datasets.

3.5.3.1 Comparing with Matrix Factorization Methods

Firstly, MMF are compared with MF and its variations, which are all developed from MF model. The results are shown in Table 3.3. The best result for each dataset is highlighted in boldface.

TABLE 3.3: The performance comparing with MF and variations.

Dataset	MF	PMF	BPMF	MMF
100k	0.953	0.948	0.949	**0.927**
20m Light	0.939	0.937	1.001	**0.876**
Boxoffice	0.761	0.760	0.794	**0.740**
Netflix 500	0.971	0.967	1.043	**0.904**
Netflix 1000	1.061	0.986	1.008	**0.919**

From the table, we observe that:

- MMF consistently achieves the best performance over all the datasets. This result suggests that the structure of MMF provides a better way to utilize attribute information and differentiate similar users and items.
- We can see that in 20m Light and two Netflix datasets, MMF has more significant accuracy increment than other datasets. According to Table 3.2, we find that 20m Light and two Netflix datasets have higher attribute density than others. It further indicates that attributes have helped the MMF model to learn fine grained differences between items.
- 20m Light and two Netflix datasets have lower rating density than other two datasets according to Table 3.1, but the accuracy is not influenced by the sparsity of data, which shows that MMF is capable of performing well in sparse datasets.

3.5.3.2 Comparing with Feature Interaction-Based Baselines

We now compare MMF with the attribute-based methods FM and DeepCrossing.

TABLE 3.4: The performance comparing with attribute-based methods.

Dataset	FM	DeepCrossing	MMF
100k	0.938	0.935	**0.927**
20m Light	0.889	0.880	**0.876**
Boxoffice	0.749	0.743	**0.740**
Netflix 500	0.909	0.908	**0.904**
Netflix 1000	0.925	0.921	**0.919**

Table 3.4 shows the RMSE for MMF and baseline methods. From the table, we can see that our MMF model is superior in prediction accuracy comparing to the FM and DeepCrossing on all datasets. It shows the ability of MMF in leveraging user-attribute interactions for accurate predictions.

3.5.4 Different Weight Combinations

In this subsection, we will analyze how the two weights in MMF (the user preference weight ω, and the item performance weight θ) influence the prediction accuracy. We evaluate 4 combinations: base $\boldsymbol{U}\boldsymbol{A}^\top$, adding ω only, adding θ only, and adding both ω and θ (the proposed MMF model).

TABLE 3.5: The RMSE of different weight combinations in MMF.

Dataset	Base	Base+ω	Base+θ	Base+ω+θ(MMF)
100k	0.944	0.932	0.947	**0.927**
20m Light	0.960	0.924	0.881	**0.876**
Boxoffice	0.759	0.757	0.747	**0.740**
Netflix 500	0.912	0.914	0.904	**0.904**
Netflix 1000	0.928	0.928	0.920	**0.919**

The prediction results of four combinations are shown in Table 3.5. We can see that:

- The proposed MMF model performs best among all combinations, which shows that the ω and the θ together have positive influence on improving accuracy.

- In some situations, such as in 100k and two Netflix datasets, adding one weight cannot guarantee to improve prediction accuracy. This demonstrates that the three components do not work individually in the model, but work together to get the best results.

3.5.5 Interpretability Case Study

It is necessary to exam the interpretability of MMF model. In this subsection, we give a case study to validate it in two steps. First, we demonstrate that the MMF model can capture semantic dependencies between attributes. Then, we give an example to show how MMF can provide interpretable recommendations and analyze whether the interpretations are consistent with users' personal interests.

Specifically, we choose the movie "*Guardians of the Galaxy*" from Boxoffice dataset, which containing 8 attributes: 1 director, 1 topic, 3 casts and 3 genres. The topic is the 7^{th} topic generated by LDA, which contains the following keywords that indicate the elements in the movie: *superhero, orb, galaxy, musketeer, etc.*. The latent vectors for the 8 attributes will be visualized to show their semantic dependencies, and we will analyze the preference of the 2 users on this movie.

3.5.5.1 Semantic Dependency

In this part, we visualize the learned attribute latent vectors in the movie "*Guardians of the Galaxy*". To draw the latent vectors, we utilize the t-SNE technique [237] to reduce the high dimension latent vectors into 3 dimensions while preserving the spatial relationship. Figure 3.3 shows the latent attribute vectors belonging to the movie in a 3-dimension form. Specifically, the attributes are "James Gunn" (Director), "Topic 7", "Chris Pratt" (Cast1), "Zoe Saldana" (Cast2), "Dave Bautista" (Cast3), "Action" (Genre1). "Adventure" (Genre2) and "Sci-Fi" (Genre3).

From the figure, we can see that the attribute latent vectors learned by MMF show strong links between spatial and semantic relationship. For example, attributes "Action" (Genre1) and "Adventure" (Genre2) are very close to each other in spatial. Semantically, movies belonging to the genres of both action and adventure account for 23% of all movies, which is much higher than other genre combinations. Meanwhile, "Topic 7" is closer to "Sci-Fi" (Genre3) than another two genres. Intuitively, elements like "superhero", "orb", "galaxy" appears often in movies of genre "Sci-Fi" than those of genre "action" and "adventure", which is constant with our claims.

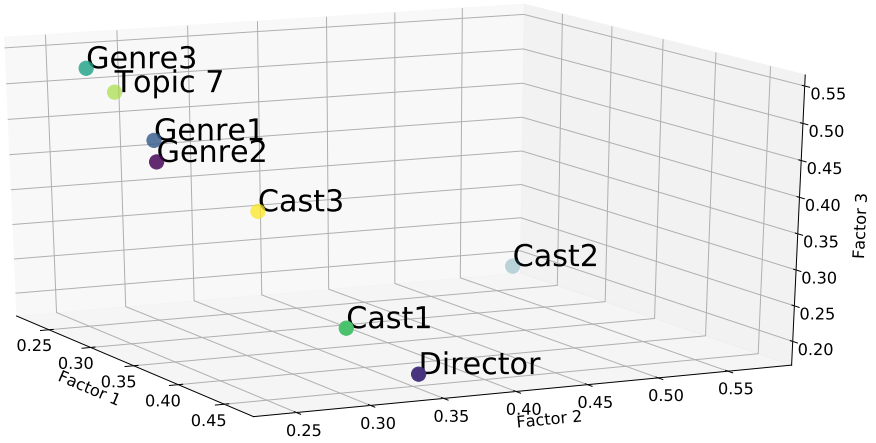


FIGURE 3.3: A visualization of attribute latent vectors in one movie (The latent vectors have been compressed into 3 dimensions using t-SNE).

3.5.5.2 Interests Consistency

In the previous section, we have stated that the MMF model have the ability to recommend items to users with predicted personal attribute preference. It is crucial to show whether the preference predictions are consistent with users' real interests. In this part, we first demonstrate how our model can recommend the movie "*Guardians of the Galaxy*" to two users with different reasons, and then analyze whether the predictions show the two users' real preferences about this movie based on their rating histories.

In our example, the MMF model predicts that user A and user B will give similar ratings to the movie. In addition, it also predicts that the attribute preference of the two users are different. Figure 3.4 shows the percentage of ratings that compose the overall rating of the movie. The inner loop of the pie chart belongs to user A and the outer loop belongs to user B. We can see that although two users may give similar ratings, the reasons are different.

From the pie chart, we can see that the rating composition of the two users are different. For example, for user A, the cast Dave Bautista (17%) may be the most important motivation to watch the movie, while for user B, the cast Zoe Saldana (19%) and the genre Adventure (18%) are the attractions. Therefore, MMF can recommend the movie to different users with personalized reasons based on the attribute proportion.

However, our interpretations will be convincing only if they truly capture users' interests. We evaluate it based on an assumption about users' interests: *a user will give higher ratings to movies that contain the attributes the user loves.* This assumption is quantified by calculating an *attribute aware rating difference* (AAD) on each attribute for users. Then we explore whether they are constant with our predictions.

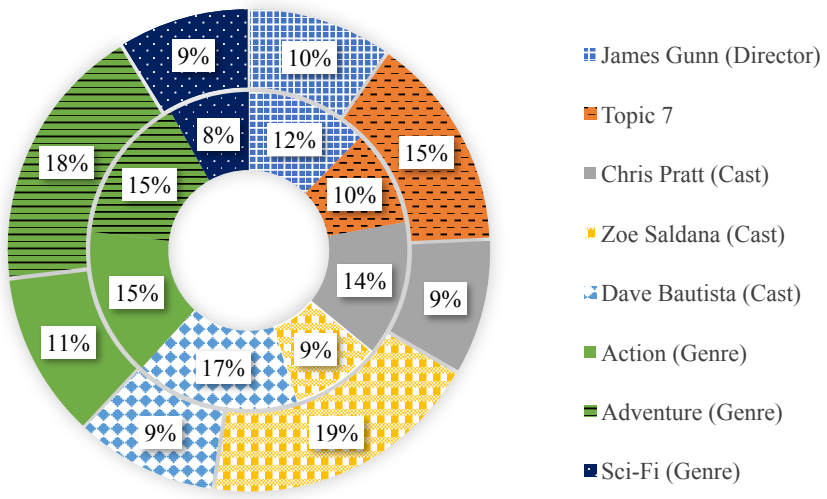


FIGURE 3.4: The preference of user A (inner loop) and user B (outer loop).

To calculate the AAD of an attribute for a user, we first extract k attributes (in our case study, $k = 5$) that are most similar (shortest Euclidean distance) to that attribute, then B_k is a set containing all movies that contain these $k + 1$ attribute (including the original one) and are also rated by the user. Then the AAD is:

$$AAD = \frac{1}{|B_k|} \sum_{j \in B_k} r_j - \frac{1}{|B_n|} \sum_{i \in B_n} r_i, \tag{3.8}$$

where B_n is a set contains all movies that the user has rated.

We can see that the first term of Equation (3.8) is the average rating of movies that a user has rated on an attribute (and similar attributes), and the second term is the average ratings of all movies the user has rated. According to the assumption above, if a user really likes an attribute, the AAD of this attribute will be positive, and vise versa. In addition, if a user loves attribute a_1 more than attribute a_2, the AAD of a_1 is supposed to be greater than a_2 for this user. Although AAD may not always be accurate due to that a user only rates a small amount of movies, the trend should roughly hold, which means that AAD can reflect users' interests to a certain extent.

Figure 3.5 shows the relationship between AAD and the predicted proportion. We find a strong positive correlation between them, which shows that the predicted proportions are consistent with users' interests. Therefore, we can conclude that our recommendations can capture users' interests and the interpretations of the MMF model are convincing.

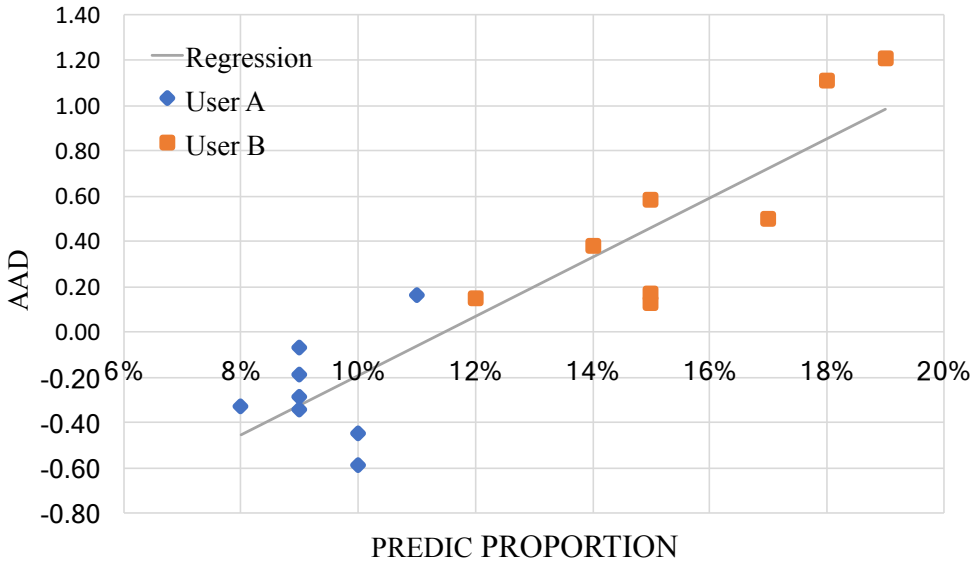


FIGURE 3.5: The correlation between predicted attribute proportion and AAD.

3.5.6 Item Cold-Start Problem

To explore the ability of the MMF model in item cold-start situations, we test the MMF model and MF model on the datasets that simulate item cold-start situation. We use RMSE to measure the performance as well.

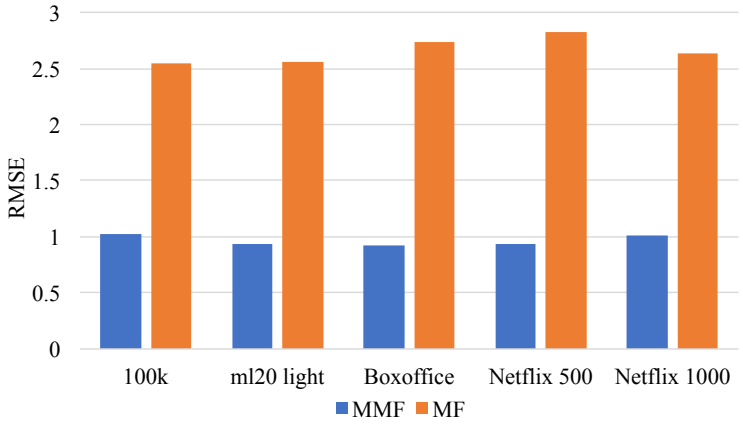


FIGURE 3.6: The prediction results on item cold-start situation.

Figure 3.6 shows the RMSE results of two models on item cold-start situation. We can see that the MMF model has significant improvement on accuracy comparing to the MF model. It indicates that the MMF model successfully leverages attributes in items and gives accurate recommendation predictions on the movies without rating history. The MF model, however, provides nearly random predictions. Since the item cold-start problem happens frequently in practice, the MMF model shows greater value on practical applications.

3.5.7 Evaluation on Different Latent Vector Lengths

In this subsection, we evaluate the performance of state-of-the-art matrix factorization models and MMF on different latent vector length. We test 4 different latent vector length (5, 10 ,15, 20) in this experiment.

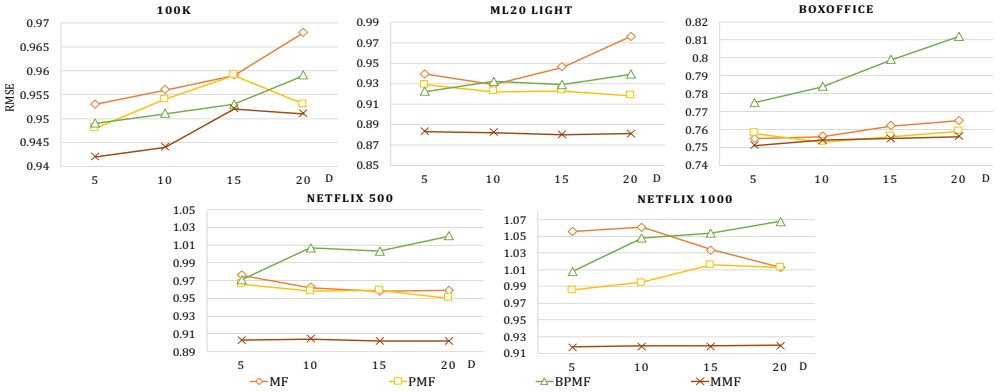


FIGURE 3.7: The test results of different latent vector length on MMF and baseline methods.

Figure 3.7 shows the test results. From the figure, we can see that for all datasets, MMF has better RMSE in all length of latent vectors. Moreover, the performance of MMF is more consistent than other methods when vector length changes. This indicates that our has more robust performance in different lengths of latent vectors.

3.5.8 Evaluation on Different Topic Numbers

In previous experiments, we fix the topic number to 50 for all datasets. In this part, we evaluate our model's prediction accuracy on different topic numbers. We set 5 different topic numbers for all datasets, they are 0 (ignore topic attribute), 10, 20, 50 and 100. The results are shown in Table 3.6.

TABLE 3.6: The performance of different topic numbers.

Dataset	0	10	20	50	100
100k	0.942	0.938	0.938	0.936	0.935
20m light	0.889	0.877	0.880	0.876	0.875
Boxoffice	0.757	0.751	0.753	0.750	0.751
Netflix 500	0.908	0.903	0.904	0.902	0.903
Netflix 1000	0.923	0.921	0.919	0.919	0.920

From the table, we find that considering topic attribute (topic number is larger than 0) improves the performance than ignoring it. It demonstrate that the topic information

is beneficial to the MMF model's performance improvement. However, among the last four columns, there are not much difference on the performance when topic number changes. That may because that the topic attribute are generated from LDA model, which may not be accurate. Adding topic number may not provide with more useful information. Moreover, it also proves that MMF is robust: even the attributes are not accurate enough, the MMF model still performs firm and steady.

3.6 Summary

In this chapter, we investigate how to enable MF the ability to consider feature interactions. To this end, we propose a novel attribute interpretable collaborative filtering model called Multi-Matrix Factorization (MMF). It offers great attribute-level interpretability in explaining how each attribute contributes to the overall predicted rating. At the same time, it solves the item cold-start problem, a critical limitation of the matrix factorization model. Experimental results show that MMF outperforms state-of-the-art baselines in terms of accuracy.

In the next chapter, we will discuss the second research question we focus on: how to detect beneficial feature interactions for recommendation. Also, from the next chapter, all of our works are based on the graph neural network-based explicit feature interaction modeling.

Chapter 4

Detecting Pairwise Beneficial Feature Interactions for Recommender Systems

This chapter is based on the paper that has been published in the Conference on Artificial Intelligence (AAAI) 2021: ***Detecting Beneficial Feature Interactions for Recommender Systems****. Yixin Su, Rui Zhang, Sarah Erfani and Zhenghua Xu.*

4.1 Introduction

In this chapter, we start to investigate the second research question of detecting beneficial feature interactions to improve the recommendation performance. In recent years, many research efforts have been put in modeling the feature interactions [11, 16, 108]. Our motivation of this research question is that these models take into account the interaction between *every* pair of features. However, in practice, not all feature interactions are relevant to the recommendation result [14, 15]. Modeling the feature interactions that provide little useful information may introduce noise and cause overfitting, and hence decrease the prediction accuracy [150, 151]. For example, a user may use *Gmail* on a workday no matter what weather it is. However, if the interaction of workday and specific weather is taking into account in the model and due to the bias in the training set (e.g., the weather of the days when the Gmail usage data are collected happens to be cloudy), the interaction $< Monday, Cloudy >$ might be picked up by the model and make less accurate recommendations on Gmail usage.

The most intuitive method to find the useful feature interactions is to rely on the expert knowledge, i,e., invite domain experts to identify which feature interactions are important. However, this method has some clear drawbacks. First, the quality of the identified feature interactions heavily depends on the experts. If the experts are not fully qualified, the performance of the final prediction may be decreased. Second, there are a vast number of feature interactions. Identifying the usefulness of each feature interaction manually is time and cost consuming. Third, experts are domain specific and the expert knowledge is hard to adapt to different recommendation tasks. That is to say, we need to find related experts to do the interaction identification for each recommendation task.

Recent years, with the development of machine learning techniques, data driven methods to identify useful feature interactions emerge [4, 238]. For example, some work considers each feature interaction's importance through the attention mechanism [4, 11]. However, these methods still take all feature interactions into account. Moreover, they capture each *individual* feature interaction's contribution to the recommendation prediction, failing to capture the *holistic* contribution of a set of feature interactions together.

In this work, we focus on identifying the set of feature interactions that together produce the best recommendation performance. To formulate this idea, we propose the novel problem of *detecting beneficial feature interactions*, which is defined as identifying the set of feature interactions that contribute most to the recommendation accuracy (see Definition 4.1 for details). Then, we propose a novel graph neural network (GNN)-based recommendation model, L_0-SIGN, that detects the most beneficial feature interactions and utilizes only the beneficial feature interactions for recommendation, where each data sample is treated as a graph, features as nodes and feature interactions as edges. Specifically, our model consists of two components. One component is an L_0 edge prediction model, which detects the most beneficial feature interactions by predicting the existence of edges between nodes. To ensure the success of the detection, an L_0 activation regularization is proposed to encourage unbeneficial edges (i.e. feature interactions) to have the value of 0, which means that edge does not exist. Another component is a graph classification model, called **S**tatistical **I**nteraction **G**raph neural **N**etwork (SIGN). SIGN takes nodes (i.e., features) and detected edges (i.e., beneficial feature interactions) as the input graph, and outputs recommendation predictions by effectively modeling and aggregating the node pairs that are linked by an edge.

Theoretical analyses are further conducted to verify the effectiveness of our model. First, the most beneficial feature interactions are guaranteed to be detected in L_0-SIGN. This is proved by showing that the empirical risk minimization procedure of L_0-SIGN is a variational approximation of the Information Bottleneck (IB) principle, which is a golden

criterion to find the most relevant information correlating to target outcomes from inputs [239]. Specifically, only the most beneficial feature interactions will be retained in L_0-SIGN. It is because, in the training stage, our model simultaneously minimizes the number of detected feature interactions by the L_0 activation regularization, and maximizes the recommendation accuracy with the detected feature interactions. Second, we further show that the modeling of the detected feature interactions in SIGN is very effective. By accurately leveraging the relational reasoning ability of GNN, *iff* a feature interaction is detected to be beneficial in the L_0 edge prediction component, it will be modeled in SIGN as a statistical interaction (an interaction is called statistical interaction if the joint effects of variables are modeled correctly).

We summarize our contributions as follows:

- This is the first work to formulate the concept of beneficial feature interactions, and propose the problem of detecting beneficial feature interactions for recommender systems.
- We propose a model, L_0-SIGN, to detect beneficial feature interactions via a graph neural network approach and L_0 regularization.
- We theoretically prove the effectiveness of L_0-SIGN through the information bottleneck principle and statistical interaction theory.
- We have conducted extensive experimental studies. The results show that (i) L_0-SIGN outperforms existing baselines in terms of accuracy; (ii) L_0-SIGN effectively identifies beneficial feature interactions, which result in the superior prediction accuracy of our model.

4.2 Preliminaries

4.2.1 L_0 Regularization

L_0 regularization encourages the regularized parameters $\boldsymbol{\theta}$ to be exactly zero by setting an L_0 term:

$$\|\boldsymbol{\theta}\|_0 = \sum_{j=1}^{|\boldsymbol{\theta}|} \mathbb{I}[\theta_j \neq 0], \tag{4.1}$$

where $\mathbb{I}$ is 1 if $\theta_j \neq 0$, and 0 otherwise.

For a dataset D, an empirical risk minimization procedure is used with L_0 regularization on the parameters $\boldsymbol{\theta}$ of a hypothesis $\mathcal{H}(\cdot;\boldsymbol{\theta})$, which can be any objective function

involving parameters, such as neural networks. Then, using reparameterization of $\boldsymbol{\theta}$, we set $\theta_j = \tilde{\theta}_j z_j$, where $\tilde{\theta}_j \neq 0$ and z_j is a binary gate with Bernoulli distribution $Bern(\pi_j)$ [151]. The procedure is represented as:

$$\begin{aligned} \mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}) &= \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{\pi})} \frac{1}{N} (\sum_{n=1}^{N} \mathcal{L}(\mathcal{H}(\boldsymbol{x}_n; \tilde{\boldsymbol{\theta}} \odot \boldsymbol{z}), y_n)) + \lambda \sum_{j=1}^{|\boldsymbol{\theta}|} \pi_j, \\ \tilde{\boldsymbol{\theta}}^*, \boldsymbol{\pi}^* &= \arg\min_{\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}} \mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}), \end{aligned} \tag{4.2}$$

where $p(z_j|\pi_j) = Bern(\pi_j)$, N is the number of samples in D, $\odot$ is element-wise production, $\mathcal{L}(\cdot)$ is a loss function and λ is the weighting factor of the L_0 regularization.

4.2.2 Graph Neural Networks

Consider a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \{i\}_{1 \leq i \leq k}$ is the set of k nodes, each of which is represented by a vector representation $\boldsymbol{v}_i$, and $\mathcal{E}$ is the set of edges which indicate the neighborhood information between nodes: two nodes are neighbors if they are linked by an edge. A graph neural network (GNN) learns the vector representation of each node by message passing, a procedure of aggregating neighborhood information. Specifically, the message passing procedure for node i first aggregates the vector representations of all its neighbors. Then, it gets the fused representation of node i by fusing $\boldsymbol{v}_i$ and the aggregated vector representation. Formally, the fused vector representation of node i, $\boldsymbol{v}'_i$, through the graph modeling is:

$$\boldsymbol{v}'_i = f_{fuse}(\boldsymbol{v}_i, \text{AGGREGATE}_v(\{\boldsymbol{v}_j\}_{j \in N(i)})), \tag{4.3}$$

where f_{fuse} is the fusing function, $\text{AGGREGATE}_v(\cdot)$ is an aggregation function that aggregates the neighborhood embeddings into a fixed dimension representation (e.g., element-wise sum) and $N(i)$ is the set of all the neighbors of node i.

If it is necessary, the graph representation can be computed as the aggregation of the vector embeddings of all the nodes: $\boldsymbol{v}_G = \text{AGGREGATE}_G(\{\boldsymbol{v}'_i\}_{i \in \mathcal{V}})$, where $\boldsymbol{v}_G$ is the graph representation and $\text{AGGREGATE}_G(\cdot)$ is a node aggregation function that is similar to the one for aggregating nodes' neighbors.

4.3 Problem Statement

Denote by $\mathcal{J}$ the *universe* of features. A *feature-value pair* is a name-value pair, denoted by (a, w), where a is the name of the feature and w is the value of this feature. For

example, $(Male, 1)$ and $(Female, 1)$ mean the categorical features of Male and Female, respectively, where $Male \in \mathcal{J}$ and $Female \in \mathcal{J}$ are considered as two different features. Let $\mathcal{D} : \mathcal{X} \times \mathcal{Y}$ be an input-output domain and $D = \{(\boldsymbol{x}_n, y_n)\}_{i=1}^{N}$ is a set of N input-output training pairs sampled from $\mathcal{D}$.

- $\boldsymbol{x}_n = \{(a, w)\}_{a \in J_n}$ is called a *data sample* consisting of a set of features, with respect to a set of features $J_n \subseteq \mathcal{J}$.
- $y_n \in \{0, 1\}$ is the *implicit feedback* (e.g., purchased, clicked) of the user on the item.

Our goal is to design a model $F(\boldsymbol{x}_n)$ that detects the most beneficial *pairwise*[1] feature interactions and utilizes only the detected feature interactions to predict the true output y_n.

4.4 Definitions

4.4.1 Pairwise Beneficial Feature Interactions

Inspired by the definition of relevant feature by usefulness [14, 240], we formally define the beneficial feature interactions in Definition 4.1.

Definition 4.1. (**Beneficial Feature Interactions**) Given a data sample $\boldsymbol{x} = \{x_i\}_{1 \leq i \leq k}$ of k features whose corresponding full pairwise feature interaction set is $I = \{(x_i, x_j)\}_{1 \leq i,j \leq k}$, a set of pairwise feature interactions $I_1 \subseteq I$ is more beneficial than another set of pairwise feature interactions $I_2 \subseteq I$ to a model with $\boldsymbol{x}$ as input if the accuracy of the predictions that the model produces using I_1 is higher than the accuracy achieved using I_2.

The above definition formalizes our detection goal: find and retain only a part of feature interactions that together produce the highest prediction accuracy by our model.

4.4.2 Statistical Interaction

Statistical interaction, or non-additive interaction, ensures a joint influence of several variables on an output variable is not additive [7]. Sorokina et al. [238] formally define the pairwise statistical interaction:

[1] We focus on pairwise feature interactions in this work, and we leave high-order feature interactions in Chapter 5.

Definition 4.2. (**Pairwise Statistical Interaction**) Function $F(\boldsymbol{x})$, where $\boldsymbol{x} = \{x_i\}_{1 \leq i \leq k}$ has k variables, shows **no** pairwise statistical interaction between variables x_i and x_j if $F(\boldsymbol{x})$ can be expressed as the sum of two functions $f_{\setminus i}$ and $f_{\setminus j}$, where $f_{\setminus i}$ does not depend on x_i and $f_{\setminus j}$ does not depend on x_j:

$$\begin{aligned} F(\boldsymbol{x}) =& f_{\setminus i}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_k) \\ &+ f_{\setminus j}(x_1, ..., x_{j-1}, x_{j+1}, ..., x_k). \end{aligned} \tag{4.4}$$

More generally, if using $\boldsymbol{v}_i \in \mathbb{R}^d$ to describe the i-th variable with d factors [2], e.g., variable embedding, each variable can be described in a vector form $\boldsymbol{u}_i = x_i \boldsymbol{v}_i$. Then, we define the pairwise statistical interaction in variable factor form by changing the Equation (4.4) into:

$$\begin{aligned} F(\boldsymbol{x}) =& f_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_k) \\ &+ f_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_k). \end{aligned} \tag{4.5}$$

The definition indicates that the interaction information of variables (features) x_i and x_j will not be captured by $F(\boldsymbol{x})$ if it can be expressed as the above equations. Therefore, to correctly capture the interaction information, our model should not be expressed as the above equations. In this work, we theoretically prove that the interaction modeling in our model strictly follow the definition of pairwise statistical interaction, which ensures our model to correctly capture interaction information.

4.5 Our Proposed Method

In this section, we formally describe L_0-SIGN's overview structure and the two components in detail. Then, we provide theoretical analyses of our model.

4.5.1 L_0-SIGN

4.5.1.1 Model Overview

Each input of L_0-SIGN is represented as a graph (without edge information), where its features are nodes and their interactions are edges. More specifically, the n's data sample is represented as a graph $G_n\langle \mathcal{V}_n, \mathcal{E}_n \rangle$, where $\mathcal{V}_n = \boldsymbol{x}_n$ and $\mathcal{E}_n = \{(e_n)_{ij}\}_{i,j \in \boldsymbol{x}_n}$ is a set of edge/interaction values [2], where $(e_n)_{ij} \in \{1, 0\}$, 1 indicates that there is an edge

[2] In this work, nodes and features are used interchangeably, and the same as edges and feature interactions.

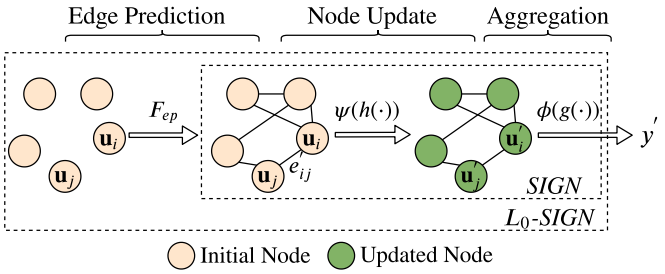


FIGURE 4.1: An overview of L_0-SIGN.

(beneficial feature interaction) between nodes i and j, and 0 otherwise. Since we do not know which feature interactions are beneficial at the before training, $\mathcal{E}_n = \emptyset$

While predicting, the L_0 edge prediction component, $F_{ep}(\boldsymbol{x}_n; \boldsymbol{\omega})$, analyzes the existence of edges on each pair of nodes, where $\boldsymbol{\omega}$ are parameters of F_{ep}, and outputs the predicted edge set $\mathcal{E}'_n$. Then, the graph classification component, SIGN, performs predictions based on $G\langle \mathcal{V}_n, \mathcal{E}'_n \rangle$. Specifically, SIGN firstly conducts interaction modeling on each pair of initial node representation that are linked by an edge. Then, each node is updated via aggregating all of the corresponding modeling results. Finally, all updated node representations are aggregated to get the final prediction. The general form of SIGN prediction function is $y'_n = f_S(G_n\langle \mathcal{V}_n, \mathcal{E}'_n \rangle; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is SIGN's parameters and the predicted outcome y'_n is the graph classification result. Therefore, the L_0-SIGN prediction function f_{LS} is:

$$f_{LS}(G_n\langle \mathcal{V}_n, \emptyset \rangle; \boldsymbol{\theta}, \boldsymbol{\omega}) = f_S(G_n\langle \mathcal{V}_n, F_{ep}(\boldsymbol{x}_n; \boldsymbol{\omega}) \rangle; \boldsymbol{\theta}). \tag{4.6}$$

Figure 4.1 shows the structure of L_0-SIGN. Next, we will show the two components in detail. When describing the two components, we focus on one input-output pair, so we omit the index "n" for simplicity.

4.5.1.2 L_0 Edge Prediction Model

A (neural) matrix factorization (MF) based model is used for edge prediction. MF is effective in modeling relations between node pairs by factorizing the adjacency matrix of a graph into node dense embeddings [241]. In L_0-SIGN, since we do not have the ground truth adjacency matrix, the gradients for optimizing this component come from the errors between the outputs of SIGN and the target outcomes.

Specifically, the edge value, $e'_{ij} \in \mathcal{E}'$, is predicted by an edge prediction function $f_{ep}(\boldsymbol{v}^e_i, \boldsymbol{v}^e_j)$: $\mathbb{R}^{2\times b} \rightarrow \{0, 1\}$, which takes a pair of node embeddings for edge prediction with dimension b as input, and output a binary value, indicating whether the two nodes are connected

by an edge. The embedding of node i for the edge prediction ($\boldsymbol{v}_i^e$) is generated via $\boldsymbol{v}_i^e = \text{emb}_e(i)$, where $\text{emb}(\cdot)$ maps the node i into an embedding of d dimension. To ensure that the detection results are identical to the same pair of nodes, f_{ep} should be invariant to the order of its input, i.e., $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) = f_{ep}(\boldsymbol{v}_j^e, \boldsymbol{v}_i^e)$. For example, in our experiments, we use an multilayer neural network (MLP) with the input as the element-wise product result of $\boldsymbol{v}_i^e$ and $\boldsymbol{v}_j^e$ (details are in the section of experiments). Note that $e'_{ii} = 1$ can be regarded as the feature i being beneficial.

While training, an L_0 activation regularization is performed on $\mathcal{E}'$ to minimize the number of detected edges, which will be described later in the section about the empirical risk minimization function of L_0-SIGN.

4.5.1.3 SIGN

In SIGN, each node i is first mapped into an initial node embedding $\boldsymbol{v}_i$ of d dimensions for interaction modeling via a mapping function $\text{emb}(\cdot)$. Note that each node has node embeddings $\boldsymbol{v}_i^e$ and $\boldsymbol{v}_i$ for edge prediction and interaction modeling, respectively, to ensure the best performance of respective tasks. Then, the interaction modeling is performed on each node pair (i, j) that $e'_{ij} = 1$, by a non-additive function $h(\boldsymbol{u}_i, \boldsymbol{u}_j) : \mathbb{R}^{2\times d} \to \mathbb{R}^d$ (e.g., an MLP), where $\boldsymbol{u}_i = w_i \boldsymbol{v}_i$. The output of $h(\boldsymbol{u}_i, \boldsymbol{u}_j)$ is denoted as $\boldsymbol{z}_{ij}$. Similar to f_{ep}, h should also be invariant to the order of its input. The above procedure can be reformulated as $\boldsymbol{s}_{ij} = e'_{ij}\boldsymbol{z}_{ij}$, where $\boldsymbol{s}_{ij} \in \mathbb{R}^d$ is called the statistical interaction analysis result of (i, j).

Next, each node is updated by aggregating all of the analysis results between the node and its neighbors using a linear aggregation function ψ: $\boldsymbol{v}'_i = \psi(\varsigma_i)$, where $\boldsymbol{v}'_i \in \mathbb{R}^d$ is the updated embedding of node i, ς_i is a set of statistical interaction analysis results between node i and its neighbors. Note that ψ should be invariant to input permutations, and be able to take inputs with variant number of elements (e.g., element-wise summation/mean).

Finally, each updated node embedding will be transformed into a scalar value by a linear function $g : \mathbb{R}^d \to \mathbb{R}$, and all scalar values are linearly aggregated as the output of SIGN. That is: $y' = \phi(\nu)$, where $\nu = \{g(\boldsymbol{u}'_i) \mid i \in \boldsymbol{x}\}$, $\boldsymbol{u}'_i = x_i \boldsymbol{v}'_i$ and $\phi : \mathbb{R}^{|\nu|\times 1} \to \mathbb{R}$ is an aggregation function having similar properties to ψ. Therefore, the prediction function of SIGN is:

$$f_S(G; \boldsymbol{\theta}) = \phi(\{g(\psi(\{e'_{ij} h(\boldsymbol{u}_i, \boldsymbol{u}_j)\}_{j\in\boldsymbol{x}}))\}_{i\in\boldsymbol{x}}). \tag{4.7}$$

In summary, we formulate the L_0-SIGN prediction function of Equation (4.6) with the two components in detail:

$$f_{LS}(G;\boldsymbol{\omega},\boldsymbol{\theta}) = \phi(\{g(\psi(\{f_{ep}(\boldsymbol{v}_i^e,\boldsymbol{v}_j^e)h(\boldsymbol{u}_i,\boldsymbol{u}_j)\}_{j\in\boldsymbol{x}}))\}_{i\in\boldsymbol{x}}). \tag{4.8}$$

We formulate the pseudocodes of the SIGN and L_0-SIGN prediction functions:

Algorithm 1 SIGN prediction function f_S

Input: data $G\langle\mathcal{V},\mathcal{E}\rangle$
for each pair of feature $(i,\ j)$ **do**
  if $e_{ij}=1$ **then**
    $\boldsymbol{z_{ij}} = h(w_i\boldsymbol{v}_i, w_j\boldsymbol{v}_j)$
    $\boldsymbol{s}_{ij} = \boldsymbol{z}_{ij}$
  else
    $\boldsymbol{s}_{ij} = \boldsymbol{0}$
  end if
end for
for each feature i **do**
  $\boldsymbol{v}_i' = \psi(\varsigma_i)$
  $\boldsymbol{u}_i' = w_i\boldsymbol{v}_i'$
  $\nu_i = g(\boldsymbol{u}_i')$
end for
$y' = \phi(\nu)$
Return: y'

Algorithm 2 L_0-SIGN prediction function f_{LS}

Input: data $G\langle\mathcal{V},\emptyset\rangle$
for each pair of feature $(i,\ j)$ **do**
  $e_{ij}' = HardConcrete(f_{ep}(\boldsymbol{v}_i^e,\boldsymbol{v}_j^e))$
  $\boldsymbol{z_{ij}} = h(w_i\boldsymbol{v}_i, w_j\boldsymbol{v}_j)$
  $\boldsymbol{s}_{ij} = e_{ij}'\boldsymbol{z}_{ij}$
end for
for each feature i **do**
  $\boldsymbol{v}_i' = \psi(\varsigma_i)$
  $\boldsymbol{u}_i' = x_i\boldsymbol{v}_i'$
  $\nu_i = g(\boldsymbol{u}_i')$
end for
$y' = \phi(\nu)$
Return: y'

4.5.2 Empirical Risk Minimization Function

The empirical risk minimization function of L_0-SIGN minimizes a loss function, a reparameterized L_0 activation regularization on $\mathcal{E}'_n$, and an L_2 activation regularization on $\boldsymbol{z}_n$. Instead of regularizing parameters, activation regularization regularizes the output of models [242]. We leverage activation regularization to link our model with the IB principle to ensures the success of the interaction detection, which will be discussed in the theoretical analyses.

Formally, the function is:

$$\begin{aligned}\mathcal{R}(\boldsymbol{\theta},\boldsymbol{\omega}) = \frac{1}{N}\sum_{n=1}^{N}(&\mathcal{L}(F_{LS}(G_n;\boldsymbol{\omega},\boldsymbol{\theta}),y_n)\\ &+\lambda_1\sum_{i,j\in\boldsymbol{x}_n}(\pi_n)_{ij}+\lambda_2\left\|\boldsymbol{z}_n\right\|_2),\\ \boldsymbol{\theta}^*,\boldsymbol{\omega}^* = &\arg\min_{\boldsymbol{\theta},\boldsymbol{\omega}}\mathcal{R}(\boldsymbol{\theta},\boldsymbol{\omega}),\end{aligned} \tag{4.9}$$

where $(\pi_n)_{ij}$ is the probability of $(e'_n)_{ij}$ being 1 (i.e., $(e'_n)_{ij} = Bern((\pi_n)_{ij})$), $G_n = G_n\langle\mathcal{V}_n,\emptyset\rangle$, λ_1 and λ_2 are weight factors for the regularizations, $\mathcal{L}(\cdot)$ corresponds to a loss function and $\boldsymbol{\theta}^*$, $\boldsymbol{\omega}^*$ are final parameters.

A practical difficulty of performing L_0 regularization is that it is non-differentiable. Inspired by [151], we smooth the L_0 activation regularization by approximating the Bernoulli distribution with a hard concrete distribution so that e'_{ij} is differentiable.

Specifically, let f_{ep} now output continuous values. Then

$$\begin{aligned}&u\sim\mathcal{U}(0,1),\\ &s=\text{Sigmoid}((\log u-\log(1-u)+\log(\alpha_{ij}))/\beta),\\ &\bar{s}=s(\delta-\gamma)+\gamma,\\ &e'_{ij}=min(1,max(0,\bar{s})),\end{aligned} \tag{4.10}$$

where $u\sim\mathcal{U}(0,1)$ is a uniform distribution, $\alpha_{ij}\in\mathbb{R}^+$ is the output of $f_{ep}(\boldsymbol{v}_i^e,\boldsymbol{v}_j^e)$, β is the temperature and (γ,δ) is an interval with $\gamma<0$, $\delta>0$.

Therefore, the L_0 activation regularization is changed to:

$$\sum_{i,j\in\boldsymbol{x}_n}(\pi_n)_{ij}=\sum_{i,j\in\boldsymbol{x}_n}\text{Sigmoid}(\log\alpha_{ij}-\beta\log\frac{-\gamma}{\delta}). \tag{4.11}$$

As a result, e'_{ij} follows a hard concrete distribution through the above approximation, which is differentiable and approximates a binary distribution. Following the recommendations from [243], we set $\gamma = -0.1, \delta = 1.1$ and $\beta = 2/3$ throughout our experiments. We refer interested readers to [151] for details about hard concrete distributions.

We formulate the pseudocodes of the training procedures of SIGN and L_0-SIGN:

Algorithm 3 Training procedure of SIGN

Randomly initialize $\boldsymbol{\theta}$
repeat
  for each input-output pair $(G_n\langle\mathcal{V}_n, \mathcal{E}_n\rangle, y_n)$ **do**
    Get $y'_n, \boldsymbol{v}'_n$ from $f_S(G_n; \boldsymbol{\theta})$
    $\boldsymbol{v}_n = \boldsymbol{v}'_n$
  end for
  $\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}(F_{LS}(y'_n, y_n))$
  Update $\boldsymbol{\theta}$ (exclude $\boldsymbol{v}$) by $\min \mathcal{R}(\boldsymbol{\theta})$
until reach the stop conditions

Algorithm 4 Training procedure of L_0-SIGN

Randomly initialize $\boldsymbol{\theta}, \boldsymbol{\omega}$
repeat
  for each input-output pair $(G_n\langle\mathcal{V}_n, \emptyset\rangle, y_n)$ **do**
    Get $y'_n, \boldsymbol{v}'_n$ from $f_{LS}(G_n; \boldsymbol{\theta}, \boldsymbol{\omega})$
    $\boldsymbol{v}_n = \boldsymbol{v}'_n$
  end for
  Calculate $\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega})$ through Equation (4.9)
  Update $\boldsymbol{\omega}, \boldsymbol{\theta}$(exclude $\boldsymbol{v}$) by $\min \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega})$
until reach the stop conditions

4.6 Theoretical Analysis

We conduct theoretical analyses to verify our model's effectiveness, including how L_0-SIGN satisfies the IB principle to guarantee the success of beneficial interaction detection, the relation of the statistical interaction analysis results with the spike-and-slab distribution (the golden standard in sparsity), and how SIGN and L_0-SIGN strictly follow the statistical interaction theory for effective interaction modeling.

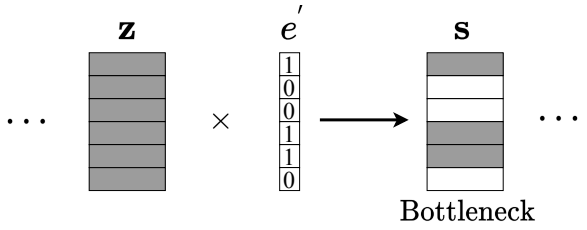


FIGURE 4.2: The interaction analysis results $\boldsymbol{s}$ from L_0-SIGN are like the relevant part (bottleneck) in the IB principle.

4.6.1 Satisfaction of Information Bottleneck (IB) Principle

IB principle [239] aims to extract the most relevant information that input random variables $\boldsymbol{X}$ contains about output variables $\boldsymbol{Y}$ by considering a trade-off between the accuracy and complexity of the process. The relevant part of $\boldsymbol{X}$ over $\boldsymbol{Y}$ denotes $\boldsymbol{S}$. The IB principle can be mathematically represented as:

$$\min \quad (\mathcal{I}(\boldsymbol{X};\boldsymbol{S}) - \iota\mathcal{I}(\boldsymbol{S};\boldsymbol{Y})), \tag{4.12}$$

where $\mathcal{I}(\cdot)$ denotes mutual information between two variables and $\iota > 0$ is a weight factor.

The empirical risk minimization function of L_0-SIGN in Equation (4.9) can be approximately derived from Equation (4.12):

$$\min \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) \approx \min(\mathcal{I}(\boldsymbol{X};\boldsymbol{S}) - \iota\mathcal{I}(\boldsymbol{S};\boldsymbol{Y})). \tag{4.13}$$

Intuitively, the L_0 regularization in Equation (4.9) minimizes a Kullback–Leibler divergence between every e'_{ij} and a Bernoulli distribution $Bern(0)$, and the L_2 regularization minimizes a Kullback–Leibler divergence between every $\boldsymbol{z}_{ij}$ and a multivariate standard distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. As illustrated in Figure 4.2, the statistical interaction analysis results, $\boldsymbol{s}$, can be approximated as the relevant part $\boldsymbol{S}$ in Equation (4.12).

Then, we give a detailed derivation from IB to L_0-SIGN:

Deep variational information bottleneck method [244] performs a variational approximation to the Information Bottleneck principle (Equation (4.12)). Specifically, the function can be approximated by maximizing a lower bound L:

$$\begin{aligned} L \approx & \frac{1}{N}\sum_{n=1}^{N}[\int d\tilde{s}_n \, p(\tilde{s}_n \mid \tilde{x}_n) \log q(\tilde{y}_n \mid \tilde{s}_n) \\ & - \iota p(\tilde{s}_n \mid \tilde{x}_n) \log \frac{p(\tilde{s}_n \mid \tilde{x}_n)}{r(\tilde{s}_n)}], \end{aligned} \tag{4.14}$$

where $\tilde{x}_n, \tilde{y}_n, \tilde{s}_n$ are input, output and the middle state respectively, $r(\tilde{s}_n)$ is the variational approximation to the marginal $p(\tilde{s}_n)$.

Then, maximizing the lower bound L equals to minimizing the J_{IB}:

$$\begin{aligned} J_{IB} = \frac{1}{N}\sum_{n=1}^{N}(&\mathbb{E}_{\tilde{s}_n \sim p(\tilde{s}_n|\tilde{x}_n)}[-\log q(\tilde{y}_n \mid \tilde{s}_n)] \\ &+ \iota KL[p(\tilde{s}_n \mid \tilde{x}_n), r(\tilde{s}_n)]), \end{aligned} \tag{4.15}$$

where $KL[p(\tilde{s}_n \mid \tilde{x}_n), r(\tilde{s}_n)]$ is the Kullback–Leibler divergence between $p(\tilde{s}_n \mid \tilde{x}_n)$ and $r(\tilde{s}_n)$.

In L_0-SIGN, the middle state part between input and output is the statistical interaction analysis result $\boldsymbol{s}_n$, which is the multiplication of predicted edge values $\boldsymbol{e}'_n$ (the vector form of $\mathcal{E}'_n$) and interaction modeling results $\boldsymbol{z}_n$. $(e'_n)_{ij} = Bern((\pi_n)_{ij})$ so that $\boldsymbol{e}'_n$ is a multivariate Bernoulli distribution, denoted as $p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)$. Similarly, $(\boldsymbol{z}_n)_{ij}$ is a multivariate normal distribution $\mathcal{N}((\boldsymbol{z}_n)_{ij}, \Sigma_{ij})$ so that $\boldsymbol{z}_n$ is a multivariate normal distribution, denoted as $p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)$. Therefore, the distribution of $\boldsymbol{s}_n$ (denoted as $p(\boldsymbol{s}_n \mid \boldsymbol{x}_n)$) is represented as:

$$\begin{aligned} p(\boldsymbol{s}_n \mid \boldsymbol{x}_n) &= p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n) p(\boldsymbol{z}_n \mid \boldsymbol{x}_n) \\ &= \|_{i,j \in \boldsymbol{x}_n}[Bern(\pi_{ij})\mathcal{N}((\boldsymbol{z}_n)_{ij}, \Sigma_{ij})], \end{aligned} \tag{4.16}$$

where Σ_{ij} is a covariance matrix and $\|$ is concatenation.

Meanwhile, we set the variational approximation of the $\boldsymbol{s}_n$ being a concatenated multiplication of normal distributions with mean of 0 and variance of 1, and Bernoulli distributions that the probability of being 0 is 1. Then, the variational approximation in the vector form is:

$$r(\boldsymbol{s}_n) = Bern(\boldsymbol{0})\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{4.17}$$

where $\boldsymbol{I}$ is an identity matrix.

Combining Equation (4.16) and Equation (4.17) into Equation (4.15), the minimization function correlating to L_0-SIGN becomes:

$$\begin{aligned} J_{IB} = \frac{1}{N}\sum_{n=1}^{N}(&\mathbb{E}_{\boldsymbol{s}_n \sim p(\boldsymbol{s}_n|\boldsymbol{x}_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\ &+ \iota KL[p(\boldsymbol{s}_n \mid \boldsymbol{x}_n), r(\boldsymbol{s}_n)]). \end{aligned} \tag{4.18}$$

Next, we use the forward Kullback–Leibler divergence $KL[r(\boldsymbol{s}_n), p(\boldsymbol{s}_n \mid \boldsymbol{x}_n)]$ to approximate the reverse Kullback–Leibler divergence in Equation (4.18) to ensure the

Kullback–Leibler divergence can be properly derived into the L_0 and L_2 activation regularization (will be illustrated in Equation (4.21) and Equation (4.22)). We can perform this approximation because when the variational approximation $r(\tilde{z}_n)$ only contains one mode (e.g., Bernoulli distribution, normal distribution), both forward and reverse Kullback–Leibler divergence force $p(\tilde{z}_n \mid \tilde{x}_n)$ to cover the only mode and will have the same effect [245]. Then Equation (4.18) becomes:

$$\begin{aligned}
& J_{IB} \\
\approx & \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|\boldsymbol{x}_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\
& + \iota KL[r(\boldsymbol{s}_n), p(\boldsymbol{s}_n \mid \boldsymbol{x}_n)]) \\
= & \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|\boldsymbol{x}_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\
& + \iota KL[Bern(\mathbf{0})\mathcal{N}(\mathbf{0}, \boldsymbol{I}), p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)]) \\
= & \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|\boldsymbol{x}_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\
& + \iota(dKL[Bern(\mathbf{0}), p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)] \\
& \quad + KL[\mathcal{N}(\mathbf{0}, \boldsymbol{I}), p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)])),
\end{aligned} \tag{4.19}$$

where d is the dimention of each vector $(\boldsymbol{z}_n)_{ij}$.

Minimizing $\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|\boldsymbol{x}_n)}[-\log q(y_n \mid \boldsymbol{s}_n)]$ in Equation (4.19) is equivalent to minimizing $\mathcal{L}(f_{LS}(G_n; \boldsymbol{\omega}, \boldsymbol{\theta}), y_n)$ in Equation (4.9): its the negative log likelihood of the prediction as the loss function of L_0-SIGN, with $p(\boldsymbol{s}_n \mid \boldsymbol{x}_n)$ being the edge prediction procedure and interaction modeling procedure, and $q(y_n \mid \boldsymbol{s}_n)$ being the aggregation procedure from the statistical interaction analysis result to the outcome y_n.

For the part $KL[Bern(\mathbf{0}), p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)]$ in Equation (4.19), $p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n) = Bern(\boldsymbol{\pi}_n)$ is a multivariate Bernoulli distribution, so the KL divergence is:

$$\begin{aligned}
& KL[Bern(\mathbf{0}), p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)] = KL[Bern(\mathbf{0}), Bern(\boldsymbol{\pi}_n)] \\
= & \sum_{i,j\in\boldsymbol{x}_n}(0\log\frac{0}{(\pi_n)_{ij}} + (1-0)\log\frac{1-0}{1-(\pi_n)_{ij}}) \\
= & \sum_{i,j\in\boldsymbol{x}_n}\log\frac{1}{1-(\pi_n)_{ij}}.
\end{aligned} \tag{4.20}$$

Next, we use a linear function $\gamma(\pi_n)_{ij}$ to approximate $\log\frac{1}{1-(\pi_n)_{ij}}$ in Equation (4.20), where $\gamma > 0$ is a scalar constant. Figure 4.3 shows the values (penalization) of the Bernoulli KL divergence and its linear approximation on different π values. It can be seen

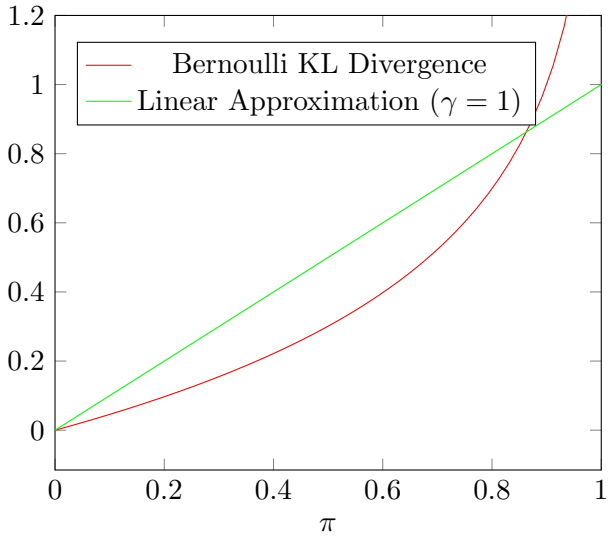


FIGURE 4.3: Linear Approximation vs. Bernoulli KL Divergence on different π values.

that both Bernoulli KL divergence and its approximations are monotonically increasing. In the empirical risk minimization procedure, they will have similar effects on penalizing those $\pi > 0$. In addition, the approximation is more suitable for our model because: (i) it penalizes more than the Bernoulli KL divergence when π is approaching 0 (take more effort on removing unbeneficial feature interactions); and (ii) it gives reasonable (finite) penalization when π is approaching 1 (retrain beneficial feature interactions), while the Bernoulli KL divergence produces infinite penalization when $\pi = 1$.

Then the KL divergence of the multivariate Bernoulli distribution can be approximately calculated by:

$$\begin{aligned} KL[Bern(\mathbf{0}), p(\boldsymbol{e}'_n \mid \boldsymbol{x}_n)] &= \sum_{i,j \in \boldsymbol{x}_n} \log \frac{1}{1 - (\pi_n)_{ij}} \\ &\approx \sum_{i,j \in \boldsymbol{x}_n} \gamma (\pi_n)_{ij}. \end{aligned} \tag{4.21}$$

For the part $KL[\mathcal{N}(\mathbf{0}, \boldsymbol{I}), p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)]$, the distribution $p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)$ is a multivariate normal distribution and is denoted as $\mathcal{N}(\boldsymbol{z}_n, \Sigma)$. If we assume all normal distributions in $p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)$ are i.i.d, and have the same variance (i.e., $\Sigma = diag(\sigma^2, \sigma^2, \ldots, \sigma^2)$ where

σ is a constant), we can reformulate the KL divergence:

$$\begin{aligned}
&KL[\mathcal{N}(\mathbf{0},\boldsymbol{I}),p(\boldsymbol{z}_n \mid \boldsymbol{x}_n)] = KL[\mathcal{N}(\mathbf{0},\boldsymbol{I}),\mathcal{N}(\boldsymbol{z}_n,\Sigma_n)] \\
=&\frac{1}{2}(\mathrm{Tr}(\Sigma_n^{-1}\boldsymbol{I}) + (\boldsymbol{z}_n-\mathbf{0})^T\Sigma_1^{-1}(\boldsymbol{z}_n-\mathbf{0}) + \ln\frac{\det\Sigma_n}{\det\boldsymbol{I}} - d|\boldsymbol{x}_n|) \\
=&\frac{1}{2}\sum_{i,j\in\boldsymbol{x}_n}\sum_{k=1}^{d}(\frac{1}{\sigma^2}+\frac{(z_n)_{ijk}^2}{\sigma^2}+\ln\sigma^2-1) \\
=&\frac{1}{2\sigma^2}\sum_{i,j\in\boldsymbol{x}_n}\sum_{k=1}^{d}((z_n)_{ijk}^2+C_2) \\
\propto&\frac{1}{2\sigma^2}\sum_{i,j\in\boldsymbol{x}_n}\sum_{k=1}^{d}(z_n)_{ijk}^2,
\end{aligned} \tag{4.22}$$

where $C_2 = 1+\sigma^2\ln\sigma^2-\sigma^2$ is a constant and $(z_n)_{ijk}$ is the kth dimension of $(\boldsymbol{z}_n)_{ij}$.

Relating to Equation (4.9), Equation (4.21) is exactly the L_0 activation regularization part and Equation (4.22) is the L_2 activation regularization part in the empirical risk minimization procedure of L_0-SIGN, with $\lambda_1 = d\iota\gamma$ and $\lambda_2 = \frac{\iota}{2\sigma^2}$. Therefore, the empirical risk minimization procedure of L_0-SIGN is proved to be a variational approximation of minimizing the object function of IB (J_{IB}):

$$\min\mathcal{R}(\boldsymbol{\theta},\boldsymbol{\omega}) \approx \min J_{IB}. \tag{4.23}$$

Therefore, through training L_0-SIGN, $\boldsymbol{s}$ is the most compact (beneficial) representation of the interactions for recommendation prediciton. This provides a theoretical guarantee that the most beneficial feature interactions will be detected.

4.6.2 Relation to Spike-and-slab Distribution

The spike-and-slab distribution [246] is the golden standard in sparsity. It is defined as a mixture of a delta spike at zero and a continuous distribution over the real line (e.g., a standard normal):

$$\begin{aligned}
p(a) = Bern(\pi), \qquad p(\theta \mid a=0) = \delta(\theta), \\
p(\theta \mid a=1) = \mathcal{N}(\theta \mid 0,1).
\end{aligned} \tag{4.24}$$

We can regard the spike-and-slab distribution as the product of a continuous distribution and a Bernoulli distribution. In L_0-SIGN, the predicted edge value vector $\boldsymbol{e}'$ (the vector form of $\mathcal{E}'$) is a multivariate Bernoulli distribution and can be regarded as $p(a)$. The interaction modeling result $\boldsymbol{z}$ is a multivariate normal distribution and can be regarded as $p(\theta)$. Therefore, L_0-SIGN's statistical interaction analysis results, $\boldsymbol{s}$, is a multivariate

spike-and-slab distribution that performs edge sparsification by discarding unbeneficial feature interactions. The retained edges in the spike-and-slab distribution are sufficient for L_0-SIGN to provide accurate predictions.

4.6.3 Statistical Interaction in SIGN

The feature interaction modeling in SIGN strictly follows the definition of statistical interaction, which is formally described in Theorem 4.3 (Appendix A gives the proof):

Theorem 4.3. *(**Statistical Interaction in SIGN**) Consider a graph $G\langle\mathcal{V},\mathcal{E}\rangle$, where $\mathcal{V}$ is the node set and $\mathcal{E} = \{e_{ij}\}_{i,j\in\mathcal{V}}$ is the edge set that $e_{ij} \in \{0,1\}, e_{ij} = e_{ji}$. Let $G\langle\mathcal{V},\mathcal{E}\rangle$ be the input of SIGN function $f_S(G)$ in Equation (4.7), then the function flags pairwise statistical interaction between node i and node j if and only if they are linked by an edge in $G\langle\mathcal{V},\mathcal{E}\rangle$, i.e., $e_{ij} = 1$.*

Theorem 4.3 guarantees that SIGN will capture the interaction information from node pairs *iff* they are linked by an edge. This ensures SIGN to accurately leverage the detected beneficial feature interactions for both inferring the target outcome and meanwhile providing useful feedback to the L_0 edge prediction component for better detection.

4.6.4 Statistical Interaction in L_0-SIGN

L_0-SIGN provides the same feature interaction modeling ability as SIGN, since we can simply extend Theorem 4.3 to Corollary 4.4 (Section *B* of Appendix gives the proof):

Corollary 4.4. *(**Statistical Interaction in L_0-SIGN**) Consider a graph G that the edge set is unknown. Let G be the input of L_0-SIGN function $F_{LS}(G)$ in Equation (4.8), the function flags pairwise statistical interaction between node i and node j if and only if they are predicted to be linked by an edge in G by L_0-SIGN, i.e., $e'_{ij} = 1$.*

4.6.5 Time Complexity Analysis

Our model performs interaction detection on each feature interactions. Then, the detected beneficial interactions are modeled. The time complexity is $O(q^2(M^e + M^h))$, where $q = |\boldsymbol{x}_n|$, $O(M^e)$ and $O(M^h)$ are the complexities of interaction detection and interaction modeling on one feature interaction, respectively. The two aggregation procedures ϕ and ψ can be regarded together as summing up all statistical interaction analysis results two times. The complexity is $O(2dq^2)$. Finally, there is an aggregation procedure g on each node. The complexity is $O(qd)$. Therefore, the complexity of our model is $O(q^2(M^e + M^h + 2d) + qd)$.

Dataset	#Features	#Graphs	#Nodes/Graph
FRAPPE	5,382	288,609	10
MOVIELENS	90,445	2,006,859	3
TWITTER	1,323	144,033	4.03
DBLP	41,324	19,456	10.48

TABLE 4.1: Dataset statistics. All datasets are denoted in graph form. Twitter and DBLP datasets are used for question (ii).

4.7 Experiments

We focuses on answering three questions: (i) how L_0-SIGN performs compared to baselines and whether SIGN helps detect more beneficial interactions for better performance? (ii) How is the detection ability of L_0-SIGN? (iii) Can the statistical interaction analysis results provide potential explanations for the recommendations predictions?

4.7.1 Experimental Protocol

4.7.1.1 Datasets

We study two real-world datasets for recommender systems to evaluate our model:

Frappe [247] is a context-aware recommendation dataset that records app usage logs from different users with eight types of contexts (e,g, weather). Each log is a graph (without edges), and nodes are user ID, app ID, or the contexts.

MovieLens-tag [108] focuses on the movie tag recommendation (e.g., "must-see"). Each data instance is a graph, with nodes as user ID, movie ID, and a tag that the user gives to the movie.

To evaluate the question (ii), we further study two datasets for graph classification, which will be discussed later. We demonstrate the statistics of the datasets in Table 4.1.

4.7.1.2 Baselines

We compare our model with recommender system baselines that model all feature interactions. Including **FM** [74] and the following methods:

AFM [4]: Addition to FM, it calculates an attention value for each feature interaction.

NFM [108]: It replaces the dot product procedure of FM by an MLP.

DeepFM [5]: It leverages MLP to perform implicit higher-order feature interaction modeling and FM to perform pairwise feature interaction modeling, respectively.

xDeepFM [16]: It is an extension of DeepFM that models feature interactions in an explicit way.

AutoInt [11]: It explicitly models all feature interactions using a multi-head self-attentive neural network.

We use the same MLP settings in all baselines (if use) as our interaction modeling function h in SIGN for fair comparison.

4.7.1.3 Experimental Set-up

In the experiments, we use element-wise mean as both linear aggregation functions $\psi(\cdot)$ and $\phi(\cdot)$. The linear function $g(\cdot)$ is a weighted sum function (i.e., $g(\boldsymbol{u_i'}) = \boldsymbol{w}_g^T \boldsymbol{u_i'}$, where $\boldsymbol{w}_g \in \mathbb{R}^{d\times 1}$ are the weight parameters). For the interaction modeling function $h(\cdot)$, we use a MLP with one hidden layer after element-wise product: $h(\boldsymbol{u}_i, \boldsymbol{u}_j) = \boldsymbol{W}_2^h \sigma(\boldsymbol{W}_1^h(\boldsymbol{u}_i \odot \boldsymbol{u}_j) + \boldsymbol{b}_1^h) + \boldsymbol{b}_2^h$, where $\boldsymbol{W}_1^h, \boldsymbol{W}_2^h, \boldsymbol{b}_1^h, \boldsymbol{b}_2^h$ are parameters of MLP and $\sigma(\cdot)$ is a Relu activation function. We implement the edge prediction model based on the neural collaborative filtering framework [85], which has a similar form to $h(\cdot)$: $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) = \boldsymbol{W}_2^e \sigma(\boldsymbol{W}_1^e(\boldsymbol{v}_i^e \odot \boldsymbol{v}_j^e) + \boldsymbol{b}_1^e) + \boldsymbol{b}_2^e$. We set node embedding sizes for both interaction modeling and edge prediction to 8 (i.e., $b, d = 8$) and the sizes of hidden layer for both h and f_{ep} to 32. We choose the weighting factors λ_1 and λ_2 from $[1 \times 10^{-5}, 1 \times 10^{-1}]$ that produce the best performance in each dataset.

Each dataset is randomly split into training, validation, and test datasets with a proportion of 70%, 15%, and 15%. We choose the model parameters that produce the best results in validation set when the number of predicted edges being steady. We use accuracy (ACC) and the area under a curve with Riemann sums (AUC) as evaluation metrics.

4.7.2 Overall Performance

Table 4.2 shows the results of comparing our model with recommendation baselines, with the best results for each dataset in bold. The results of SIGN are using all feature interactions as input, i.e., the input is a complete graph.

Through the table, we observe that:

	Frappe		**MovieLens**	
	AUC	ACC	AUC	ACC
FM	0.9263	0.8729	0.9190	0.8694
AFM	0.9361	0.8882	0.9205	0.8711
NFM	0.9413	0.8928	0.9342	0.8903
DeepFM	0.9422	0.8931	0.9339	0.8895
xDeepFM	0.9435	0.8950	0.9347	0.8906
AutoInt	0.9432	0.8947	0.9351	0.8912
SIGN	0.9448	0.8974	0.9354	0.8921
L_0-SIGN	**0.9580**	**0.9174**	**0.9407**	**0.8970**

Table 4.2: Summary of results in comparison with baselines.

- L_0-SIGN outperforms all baselines. It shows L_0-SIGN's ability in providing accurate recommendations. Meanwhile, SIGN solely gains comparable results to competitive baselines, which shows the effectiveness of SIGN in modeling feature interactions for recommendation.
- L_0-SIGN gains significant improvement from SIGN. It shows that more accurate predictions can be delivered by retaining only beneficial feature interactions and effectively modeling them.
- FM and AFM (purely based on dot product to model interactions) gain lower accuracy than other models, which shows the necessity of using sophisticated methods (e.g., MLP) to model feature interactions for better predictions.
- The models that explicitly model feature interactions (xDeepFM, AutoInt, and L_0-SIGN) outperform those that implicitly model feature interactions (NFM, DeepFM). It shows that explicit feature interaction analysis is promising in delivering accurate predictions.

4.7.3 Comparing SIGN with Other GNNs in Our Model

To evaluate whether our L_0 edge prediction technique can be used on other GNNs and whether SIGN is more suitable than other GNNs in our model, we replace SIGN with existing GNNs: GCN [12], Chebyshev filter based GCN (Cheby) [248] and GIN [13]. We run on two datasets for graph classification since they contain heuristic edges (used to compare with the predicted edges):

Twitter [249] is extracted from twitter sentiment classification. Each tweet is a graph with nodes being word tokens and edges being the co-occurrence between two tokens in each tweet.

	Twitter		DBLP	
	AUC	ACC	AUC	ACC
GCN	0.7049	0.6537	0.9719	0.9289
L_0-GCN	0.7053	0.6543	0.9731	0.9301
CHEBY	0.7076	0.6522	0.9717	0.9291
L_0-CHEBY	0.7079	0.6519	0.9719	0.9297
GIN	0.7149	0.6559	0.9764	0.9319
L_0-GIN	0.7159	0.6572	0.9787	0.9328
SIGN	0.7201	0.6615	0.9761	0.9316
L_0-SIGN	**0.7231**	**0.6670**	**0.9836**	**0.9427**

TABLE 4.3: The results in comparison with existing GNNs. The model names without "L_0-" use heuristic edges, and those with "L_0-" automatically detect edges via our L_0 edge prediction technique.

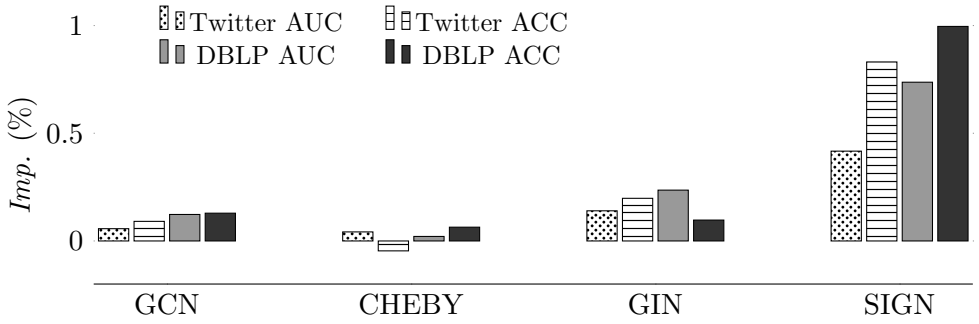


FIGURE 4.4: The comparison of improvement via the L_0 edge prediction technique to using heuristic edges in Table 4.3.

DBLP [250] consists of papers with labels indicating either they are from DBDM or CVPR field. Each paper is a graph with nodes being paper ID or keywords and edges being the citation relationship or keyword relations.

Table 4.3 shows the accuracies of each GNN that runs both on using the given (heuristic) edges (without "L_0" in name) and on predicting edges with our L_0 edge prediction model (with "L_0" in name). We can see that:

- For all GNNs, L_0-GNNs gain competitive results comparing to corresponding GNNs. It shows that our model framework lifts the requirement of domain knowledge in defining edges in order to use GNNs in some situations.
- L_0-SIGN outperforms other GNNs and L_0-GNNs, which shows L_0-SIGN's ability in detecting beneficial feature interactions and leveraging them to perform accurate predictions.

Figure 4.4 shows each GNN's improvement from the results from "GNN" to "L_0-GNN" in Table 4.3. It shows that among all the different GNN based models, SIGN gains the largest improvement from the L_0 edge prediction technique v.s. SIGN without L_0 edge prediction. It proves that SIGN can help the L_0 edge prediction model to better detect beneficial feature interactions for more accurate predictions.

4.7.4 Evaluation of Interaction Detection

We then evaluate the effectiveness of beneficial feature interaction detection in L_0-SIGN.

4.7.4.1 Prediction Accuracy vs. Numbers of Edges

Figure 4.5 shows the changes in prediction accuracy and the number of edges included while training. The accuracy first increases while the number of included edges decreases dramatically. Then, the number of edges becomes steady, and the accuracy reaches a peak at similar epochs. It shows that our model can recognize unbeneficial feature interactions and remove them for better prediction accuracy.

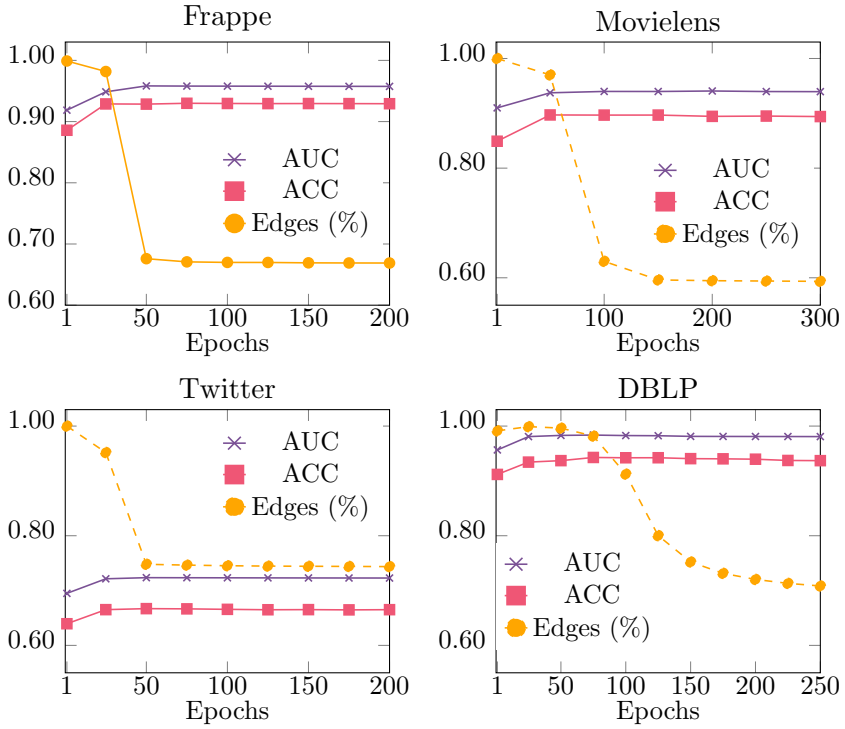


FIGURE 4.5: The changes in prediction accuracy and number of edges (in percentage) while training.

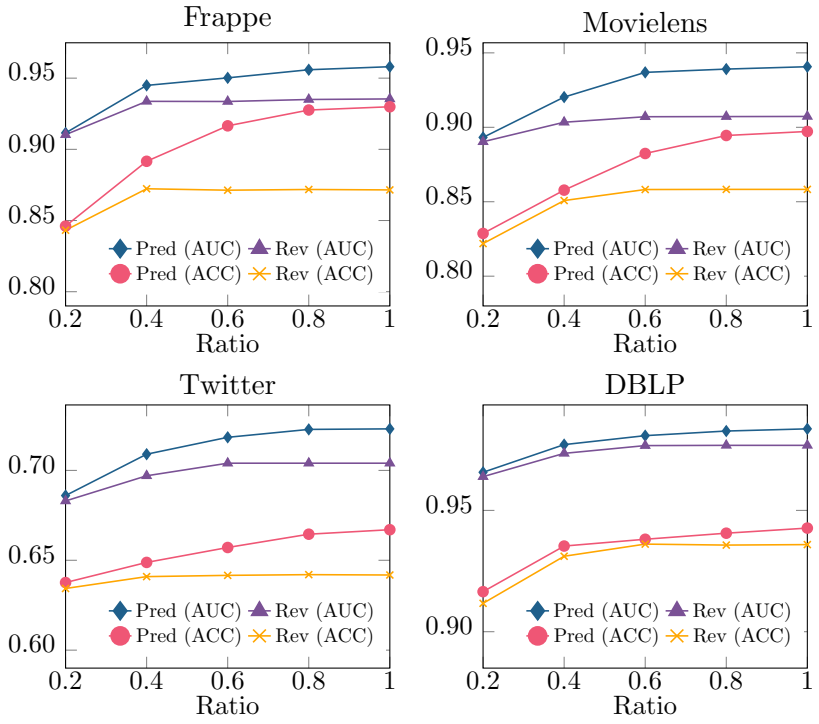


FIGURE 4.6: Evaluating different number of edges. "Pred" is the predicted edges and "Rev" is the reversed edges.

4.7.4.2 Predicted Edges vs. Reversed Edges

We evaluate how *predicted edges* and *reversed edges* (the excluded edges) influence the performance. We generate 5 edge sets with a different number of edges by randomly selecting from 20% predicted edges (ratio 0.2) to 100% predicted edges (ratio 1.0). We generate another 5 edge sets similarly from reversed edges. Then, we run SIGN on each edge set for 5 times, and show the averaged results in Figure 4.6. It shows that using predicted edges always gets higher accuracy than using reversed edges. The accuracy stops increasing when using reversed edges since the ratio 0.6, while it continually improves when using more predicted edges. According to Definition 4.1, the detected edges are proved beneficial since considering more of them can provide further performance gain and the performance is the best when using all predicted edges, while considering the reversed edges cannot. Note that the increment of reversed edges from 0.2 to 0.6 may come from covering more nodes since features solely can provide some useful information for prediction.

4.7.5 Case Study

Another advantage of interaction detection is to automatically discover potential explanations about recommendation predictions. We conduct case studies on the Frappe

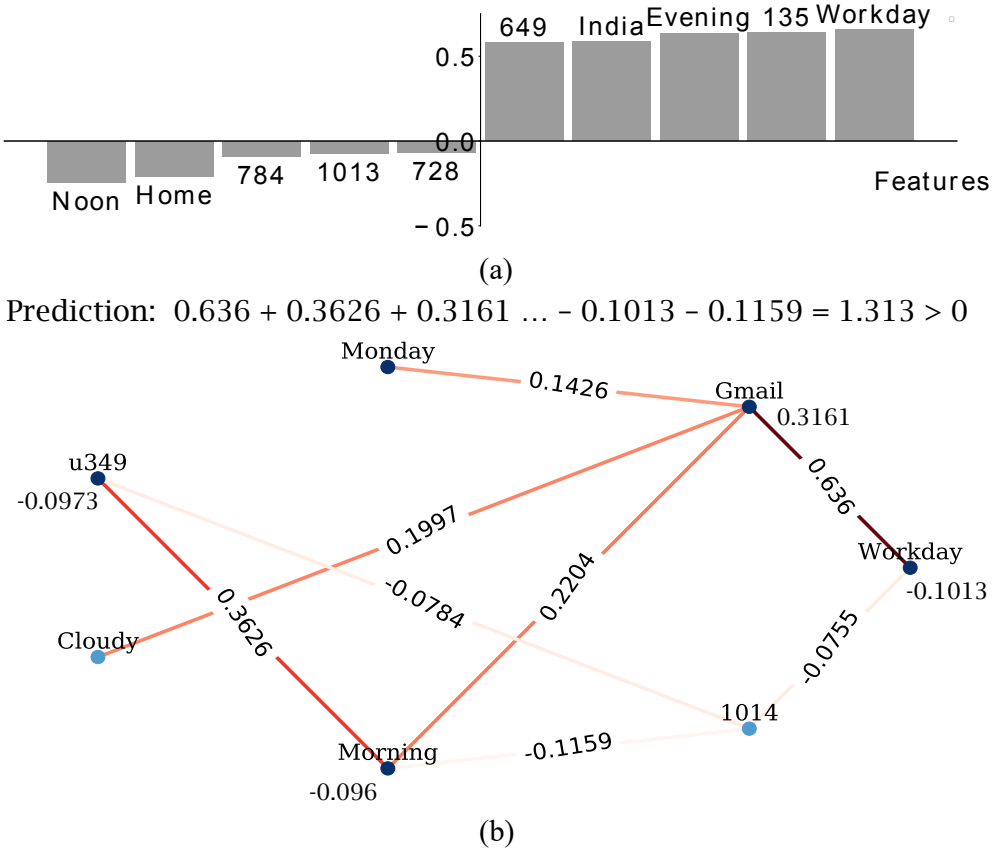


FIGURE 4.7: (a) The highest interaction values with *Gmail*. The numbers are city indexes (e.g., 1014). (b) The prediction that the user *u349* will use *Gmail* as the prediction value is $1.313 > 0$. Darker edges are higher interaction values.

dataset to show how L_0-SIGN provides the explanations.

We first show the beneficial interactions that have the highest interaction values with *Gmail* in Figure 4.7a. We can see that *Workday* has a high positive value, while *Home* has a high negative value. It may show that Gmail is very likely to be used in workdays since people need to use Gmail while working, whereas Gmail is not likely to be used at home since people may not want to dealing with email while resting. We then show how L_0-SIGN provides potential explanations for the prediction on each data sample. Figure 4.7b visualizes a prediction result that a user (*u349*) may use *Gmail*. The graph shows useful information for explanations. Despite the beneficial interactions such as $<$ *Gmail*, *Workday* $>$ that have discussed above, we can also see that *Cloudy* and *Morning* have no beneficial interaction, meaning that whether it is a cloudy morning does not contribute to decide the user's will of using Gmail.

4.8 Summary

In this chapter, we investigate the second research question of how to detect beneficial feature interactions to improve recommender systems. We are the first to propose and formulate the problem of detecting beneficial feature interactions for recommender systems. We propose L_0-SIGN to detect the beneficial feature interactions via a graph neural network approach and L_0 regularization. Theoretical analyses and extensive experiments show the ability of L_0-SIGN to detect and model beneficial feature interactions for accurate recommendations.

However, L_0-SIGN can only detect pairwise beneficial feature interactions. It is intuitive that beneficial feature interactions are not always in pairwise forms, but also may exist in more sophisticated forms (i.e., higher-order feature interactions). In the next chapter, we continue the second research question to explore detecting arbitrary order beneficial feature interactions in recommender systems.

Chapter 5

Detecting Arbitrary Order Beneficial Feature Interactions for Recommender Systems

This chapter is based on the paper that has been accepted in the Conference on Knowledge Discovery and Data Mining (SIGKDD) 2022: ***Detecting Arbitrary Order Beneficial Feature Interactions for Recommender Systems****. Yixin Su, Yunxiang Zhao, Sarah Erfani, Junhao Gan, Rui Zhang.*

5.1 Introduction

In the last chapter, we propose L_0-SIGN that can detect beneficial pairwise feature interactions for recommender systems. Recently, researchers found that more sophisticated forms of feature interactions (i.e., high-order interactions) help achieve more accurate recommendation predictions. [16, 17]. For example, if we observe that male teenagers like Nolan's sci-fi movies, the interactions (order-4) between user gender, user age, movie director, and movie genre provide useful information for recommendation that pairwise feature interactions cannot. In this chapter, we explore to effectively and efficiently detect arbitrary order beneficial feature interactions.

Existing interaction detecting methods (including L_0-SIGN) have to examine all the possible feature interactions to find the most beneficial feature interactions [17]. However, the number of all the possible feature interactions grows exponentially as the order increases. This leaves the complexity issue unsolved and they resort to detecting only limited order (e.g., combinations of up to four features) beneficial feature interactions

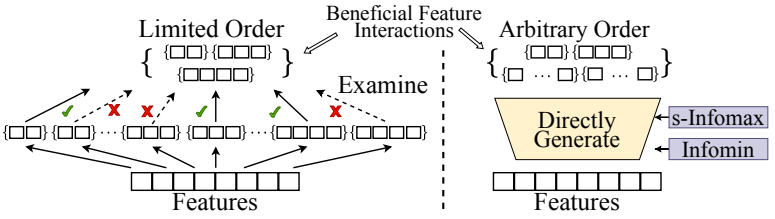


FIGURE 5.1: *Left*: existing studies detect beneficial feature interactions of limited orders by examining all the possible feature interactions. *Right*: we detect arbitrary order beneficial feature interactions by directly generating them with our deep-infomax-based methods (s-Infomax and Infomin).

(Figure 5.1 *left*). Hence, it is still an urgent yet challenging problem to efficiently detect arbitrary order beneficial feature interactions.

In this work, we propose a method that can efficiently detect beneficial feature interactions of arbitrary orders. Our method learns a neural network that directly generates the beneficial feature interactions, with the time complexity being linear to the number of beneficial interactions to be detected (Figure 5.1 *right*). This number may be specified to be one that is much smaller than the number of all the possible interactions and hence incurs a much lower time complexity. Then we achieve recommendation predictions based on the generated feature interactions. Specifically, we propose a hypergraph neural network based model called Hypergraph Infomax Recommender System (HIRS). Our model learns a hypergraph representation for each data sample, with each feature in the data represented as a node and each beneficial feature interaction represented as a hyperedge. The beneficial feature interaction generation is conducted via hyperedge matrix prediction, and the recommendation prediction is conducted via hypergraph classification.

To achieve an effective beneficial feature interaction generation, we exploit three properties of beneficial feature interactions based on information theory:

- *Sufficiency*: all beneficial feature interactions together contain as much relevant information about the true output as possible.
- *Low redundancy*: each beneficial feature interaction contains as less irrelevant information about the true output as possible.
- *Disentanglement*: any two beneficial feature interactions contain as less duplicated relevant information about the true output as possible.

Based on the three properties, we leverage Deep Infomax [154] and propose supervised mutual information maximization (s-Infomax) and mutual information minimization (Infomin) methods, together with an L_0 activation regularization, to guide the interaction generation.

To effectively learn from the generated beneficial feature interactions for recommendation, we leverage the arbitrary-order correlation reasoning ability of hypergraph neural networks [156]. We propose an interaction-based hypergraph neural network (IHGNN) to perform recommendation prediction based on the generated interactions via hypergraph classification.

We summarize our contributions as follows:

- We propose a hypergraph neural network-based model, HIRS, which directly generates a set of beneficial feature interactions of arbitrary orders, and then performs recommendation prediction via hypergraph classification.
- We define three properties of beneficial feature interactions and propose s-Infomax and Infomin methods for beneficial feature interaction generation. We further propose an interaction-based hypergraph neural network (IHGNN), which effectively learns from the generated interactions for recommendation.
- Experimental results show that HIRS outperforms state-of-the-art feature interaction-based models for recommendation. Further evaluations are conducted to show our model's beneficial feature interaction generation ability.

5.2 Preliminaries

In this section, we give the preliminaries about Hypergraph Neural Networks and Deep Infomax.

5.2.1 Hypergraph Neural Networks

Different from a simple graph, each edge in a hypergraph (i.e., hyperedge) can link to an arbitrary number of nodes. A hypergraph $G_H = \langle \mathcal{V}, \mathcal{E} \rangle$ contains a node set $\mathcal{V} = \{i\}_{i=1}^{m}$, where m is the number of nodes. Each node i is represented as a vector $\boldsymbol{v}_i \in \mathbb{R}^d$ of d dimensions (e.g., node embeddings). The hyperedge set $\mathcal{E} = \{\boldsymbol{e}_j\}_{j=1}^{k}$ contains k edges. $e_{ji} \in \{0, 1\}$, where $e_{ji} = 1$ if the j^{th} edge links to node i, and 0 otherwise. We represent all the node vectors in a graph as an node representation matrix $\boldsymbol{V} \in \mathbb{R}^{m \times d}$ and the edge set as an incidence matrix $\boldsymbol{E} \in \{0, 1\}^{m \times k}$.

In general, Hypergraph Neural Network (HGNN, Feng et al. [156]) first generates an edge representation $\boldsymbol{h}_j$ for each edge j by aggregating the linked node vectors. Then, each node vector is updated by aggregating the edge representations linked to the node (called node patch representation). The procedure is:

$$\boldsymbol{h}_j = \phi(\{\boldsymbol{V}_i | \boldsymbol{E}_{ij} = 1\}_{i \in \mathcal{V}}), \boldsymbol{v}_i^{'} = \psi(\{\boldsymbol{h}_j | \boldsymbol{E}_{ij} = 1\}_{\boldsymbol{e}_j \in \mathcal{E}}), \tag{5.1}$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are aggregation functions (e.g., element-wise mean), and $\boldsymbol{v}_i^{'}$ is the patch representation of the node i.

A graph representation $\boldsymbol{c} \in \mathbb{R}^d$ can be generated by further aggregating the node patch representations:

$$\boldsymbol{c} = \eta(\{\boldsymbol{v}_i^{'}\}_{i \in \mathcal{V}}), \tag{5.2}$$

where η is an aggregation function similar to ϕ and ψ.

5.2.2 Deep Infomax

To perform mutual information maximization (Infomax), we need to first estimate the mutual inforamtion. Given two random variables A and B, MINE [215] performs the mutual information estimation via training a classifier to distinguish their joint distribution $\mathbb{J}$ from their marginal distribution $\mathbb{M}$. Specifically, MINE uses Donsker-Varadhan representation [251] of KL-divergence as the lower bound to the mutual information:

$$\begin{aligned} \mathcal{I}(A; B) = D_{KL}(\mathbb{J} || \mathbb{M}) &\geqslant \hat{\mathcal{I}}_{\omega}^{DV}(A; B) \\ &= \mathbb{E}_{\mathbb{J}}[T_{\omega}(a, b)] - \log \mathbb{E}_M[e^{T_{\omega}(a,b)}], \end{aligned} \tag{5.3}$$

where a are the samples of A, b are the samples of B, and T_{ω} is a classifier with parameters ω.

In Deep Infomax [154] that aims to maximize a mutual information, the exact mutual information value is not important. Therefore, $\hat{\mathcal{I}}_{\omega}^{DV}(A; B)$ can be conveniently maximized by optimizing a GAN-like objective function [218, 220]:

$$\mathcal{L} = \mathbb{E}_{\mathbb{J}}[\log T_{\omega}(a, b)] + \mathbb{E}_M[\log(1 - T_{\omega}(a, b))]. \tag{5.4}$$

This objective function can be optimized by training the discriminator T_{ω} through a Binary Cross Entropy (BCE) loss [218].

5.3 Problem Statement

The data format of this work is the same as last work (Section 4.3). The difference is that in this work, we consider arbitrary order feature interactions. Given $(\boldsymbol{x}_n, y_n)$, a recommendation model that explicitly considers arbitrary order feature interactions first selects (either predefined or detected) k_n feature interactions $\boldsymbol{I}_n = \{q_i\}_{i=1}^{k_n}$, where $q_i \subseteq \mathcal{J}_n$ indicates an interaction between the features in q_i. Then, a predictive model $F(\boldsymbol{x}_n, \boldsymbol{I}_n) = y_n^{'}$ is designed which has a middle state $\mathcal{H} = \{\boldsymbol{h}_i\}_{i=1}^{k_n}$, where $\boldsymbol{h}_i \in \mathbb{R}^d$ is the high-level representation of q_i, and $y_n^{'}$ is the prediction of y_n.

5.4 Our Proposed Method

In this section, we first give an overview of our proposed method. Then, we demonstrate HIRS in detail. Finally, we show the empirical risk minimization function of HIRS.

5.4.1 Model Overview

We first introduce data representation with hypergraphs. Then we show the overall structure of HIRS.

5.4.1.1 Data Representation with Hypergraphs

We focus on one input-output pair considering k arbitrary order feature interactions. Existing GNN-based models can only represent pairwise feature interactions [103]. We propose a hypergraph form data representation that is more general in representing arbitrary order feature interactions. A hypergraph is an extension of a standard graph that each edge, i.e., hyperedge, can link to an arbitrary number of nodes. We treat each data sample as a hypergraph, with each node being a feature that occurred in the data sample. Each hyperedge represents an interaction between all the linked nodes. Formally, for the feature set of each data sample input $\boldsymbol{x} = \{(a_i, w_i)\}_{i=1}^{m}$, it can be represented as a hypergraph $G_H\langle\mathcal{V}, \mathcal{E}\rangle$. The node set $\mathcal{V} = \boldsymbol{x}$. The hyperedge set $\mathcal{E}$ consists of k beneficial feature interactions and can be represented as a hyperedge incidence matrix $\boldsymbol{E} \in \{0, 1\}^{m \times k}$.

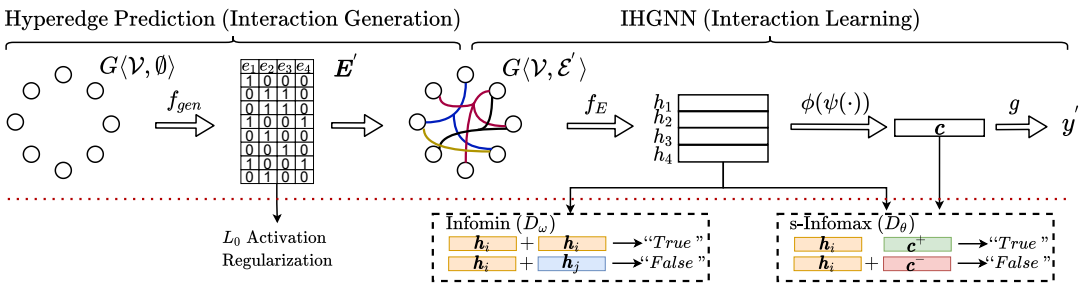


FIGURE 5.2: An Overview of HIRS. The part *above* the red dotted line is the recommendation prediction component. The part *below* the red dotted line is the s-Infomax and Infomin component and the L_0 activation regularization, which is only conducted during training.

5.4.1.2 The Overall Structure of HIRS

HIRS contains two components: the first component performs recommendations predictions, and the second component performs s-Infomax and Infomin methods based on the three properties of beneficial feature interactions. The recommendation prediction component of HIRS contains two modules: one is a hyperedge prediction module that performs interaction generation, the other is the IHGNN module that performs interaction learning. The recommendation prediction component takes a hypergraph without hyperedge as input, i.e., $G_H\langle\mathcal{V},\emptyset\rangle$. A hyperedge prediction function $f_{gen}(\mathcal{V})$ leverages the node information to generate k beneficial feature interactions as the edge set $\mathcal{E}'$. Then, IHGNN models the hypergraph with the generated edge set $G_H\langle\mathcal{V},\mathcal{E}'\rangle$ and outputs the recommendation prediction y'. The function combining the two modules is $f_r(\mathcal{V};\boldsymbol{\rho}) = f_{IHGNN}(G_H\langle\mathcal{V}, f_{gen}(\mathcal{V})\rangle)$, where $\boldsymbol{\rho}$ are all parameters in f_{gen} and f_{IHGNN}. While training, the other component containing s-Infomax and Infomin is conducted on the hyperedge representations ($\boldsymbol{h}$) and the graph representation ($\boldsymbol{c}$) in f_r, together with an L_0 activation regularization on the predicted hyperedges, to ensure the effectiveness of the interaction generation. Figure 5.2 demonstrates an overview of HIRS. In the following subsections, we describe the two components in detail.

5.4.2 The Recommendation Prediction Component

The recommendation prediction component of HIRS performs beneficial feature interaction generation via hyperedge prediction and then leverages the generated interactions for recommendation via IHGNN.

5.4.2.1 Hyperedge Prediction

The hyperedge prediction module $f_{gen}(\mathcal{V})$ generates hyperedges $\mathcal{E}^{'}$ as beneficial feature interactions in an incidence matrix format $\boldsymbol{E}^{'}$. First, each node is mapped into a node vector of d dimensions for hyperedge prediction: $\boldsymbol{V}_i^e = emb^e(a_i) \cdot w_i$, where $emb^e(\cdot)$ maps a feature into a d-dimensional embedding[1]. Then, for each node, we concatenate the vector containing the information of the node itself, and the vector summarizing the information of all other nodes in the hypergraph. The concatenated vector will be sent to an MLP to obtain a k-dimensional vector.

Finally, we need to map the values in the k-dimensional vector in to binary values as the hyperedge values of the node in the k edges. However, directly representing the hyperedge matrix by binary values results in a non-differentiable problem while training with gradient descent methods [209]. Therefore, we use the hard concrete distribution [243] as the hyperedge value to approximate the binary value. Specifically, for node i, its middle state value $\boldsymbol{\alpha}_i$ outputted by the MLP in g_{gen} will be transferred into the hyperedge values of hard concrete distribution that are the same as that in L_0-SIGN (Section 4.5.2):

$$\begin{aligned} &u \sim \mathcal{U}(0,1), \\ &s_{ij} = \text{Sigmoid}((\log u - \log(1-u) + \log(\alpha_{ij}))/\beta), \\ &\bar{s}_{ij} = s_{ij}(\delta - \gamma) + \gamma, \\ &E^{'}_{ij} = min(1, max(0, \bar{s}_{ij})), \end{aligned} \tag{5.5}$$

where u is a uniform distribution, β is a temperature value and (γ, δ) is an interval with $\gamma < 0$, $\delta > 0$.

The hard concrete distribution approximates binary values with continuous values so that the generated hyperedge values are differentiable.

In summary, the $f_{gen}(\mathcal{V})$ is:

$$\boldsymbol{E}^{'}_i = HardCon(MLP([\boldsymbol{V}^e_i, \sum\nolimits_{j \neq i} \boldsymbol{V}^e_j])), \tag{5.6}$$

where $\boldsymbol{E}^{'}_i \in \mathbb{R}^k$ is the hyperedge value vector of node i, $[\cdot]$ is the concatenation operator and $HardCon(\cdot)$ maps the output of the MLP into a hard concrete distribution.

The predicted incidence matrix $\boldsymbol{E}^{'} \in \mathbb{R}^{m \times k}$ is the concatenation of $\boldsymbol{E}^{'}_i$ of all the m nodes.

[1] emb^e ensures a feature has the same embedding in different hypergraphs, and the embedding will be updated while training.

5.4.2.2 IHGNN

With the generated edge set $\mathcal{E}^{'}$ ($\boldsymbol{E}^{'}$ in incidence matrix form), we propose an interaction-based hypergraph neural network (IHGNN) to learn $G_H\langle\mathcal{V},\mathcal{E}^{'}\rangle$.

First, we map each node in $\mathcal{V}$ into a node vector for IHGNN: $\boldsymbol{V}_i^g = emb^g(a_i)\cdot w_i$. Then, different from Hypergraph Neural Network [156] that models each hyperedge by linearly aggregating the node vectors, we use an MLP (a non-linear method) to model each hyperedge:

$$\boldsymbol{h}_j = f_E(sum(\{\boldsymbol{V}_i^g|\boldsymbol{E}_{ij}^{'}=1\}_{i\in\mathcal{V}})) \tag{5.7}$$

where f_E is an MLP, $sum(\cdot)$ is an element-wise sum and $\boldsymbol{h}_j \in \mathbb{R}^d$ is the high-level representation of edge j.

We use an MLP because feature interactions are inherently complex. The non-linearly modeling ensures IHGNN correctly and effectively models the generated interactions founded by statistical interactions defined in Section 4.4.2. This is discussed in detail in Section 5.5.1.

Then, each node representation is updated via aggregating the representations of the corresponding (linked) hyperedges by a function ψ (e.g., element-wise mean). The hypergraph representation $\boldsymbol{c}$ is obtained via further aggregating all the updated node representations by a function ϕ that is similar to ψ. Finally, we use a readout function g to map $\boldsymbol{c}$ to a scalar value as the hypergraph classification result. The IHGNN function f_{IHGNN} is:

$$f_{IHGNN}(G_H\langle\mathcal{V},\mathcal{E}^{'}\rangle) = g(\phi(\{\psi(\{\boldsymbol{h}_j|\boldsymbol{E}_{ij}^{'}=1\}_{j\in\mathcal{E}})\}_{i\in\mathcal{V}})). \tag{5.8}$$

We leverage an L_0 activation regularization like in L_0-SIGN to sparsify the generated hyperedge matrix $\boldsymbol{E}^{'}$. However, the L_0 activation regularization solely may not lead to a successful interaction detection of arbitrary orders. For example, it may lead to the generated incidence matrix containing only one edge that links to all nodes and the other $k-1$ edges being empty. As a result, all interaction information assembles in one hyperedge, which is not a beneficial feature interaction. Therefore, we exploit three properties of beneficial feature interactions in the next section to guide the beneficial feature interaction generation.

5.4.3 The s-Infomax and Infomin Component

We first formally define three properties of beneficial feature interactions. Then we describe how we achieve the three properties by s-Infomax and Infomin methods.

5.4.3.1 Three Properties of Beneficial Feature Interactions

Sufficiency: The representations of all beneficial feature interactions together contain as much information of the input about the true output as possible. In information theory, ideally, we aim to reach:

$$\mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y) = \mathcal{I}(\boldsymbol{x}; y), \tag{5.9}$$

where $\mathcal{I}(\cdot)$ is the mutual information.

Low Redundancy: Each beneficial feature interaction representation should contain as less irrelevant information about the true output as possible. Ideally, we aim to reach:

$$\sum\nolimits_{i=1}^{k} H(\boldsymbol{h}_i|y) = 0, \tag{5.10}$$

where $H(\boldsymbol{h}_i|y)$ is the entropy of $\boldsymbol{h}_i$ conditioned on y.

Disentanglement: The representations of any two beneficial feature interactions contain as less duplicated information about the true output as possible. Ideally, we aim to reach:

$$\sum\nolimits_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y) = 0. \tag{5.11}$$

Figure 5.3 shows the relations between feature interactions and the true output according to the three properties.

In HIRS, all the hyperedge representations $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\}$ are generated by a function with features $\boldsymbol{x}$ as the input, and the output y is generated through the hyperedge representations, which forms a Markov chain $\boldsymbol{x} \rightarrow \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\} \rightarrow y$. Therefore, $\mathcal{I}(\boldsymbol{x}; y) \geqslant \mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y)$. To achieve the sufficiency property, we just need to maximize $\mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y)$ to approach the optimal situation.

To achieve the low redundancy and the disentanglement properties, due to the non-negativity of entropy and mutual information, $\sum_{i=1}^{k} H(\boldsymbol{h}_i|y) \geqslant 0$ and $\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y) \geqslant 0$. Therefore, we minimize $\sum_{i=1}^{k} H(\boldsymbol{h}_i|y)$ and $\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y)$.

In summary, to achieve the three properties, we just combine all the above goals together and get the objective function in Equation (5.12):

$$\max \mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y) - \iota_1 \sum_{i=1}^{k} H(\boldsymbol{h}_i|y) - \iota_2 \sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y), \tag{5.12}$$

where ι_1 and ι_2 are scalar weights balancing the three properties.

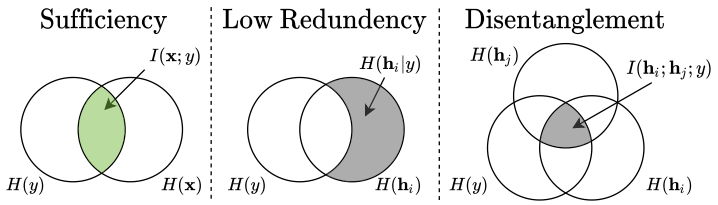


FIGURE 5.3: The Venn diagram of the three properties. The *green* part is the information that our generated interactions to retain, and the *gray* part is to discard.

5.4.3.2 s-Infomax and Infomin

Due to the difficulty of directly optimizing Equation (5.12) [154], we propose supervised mutual information maximization (s-Infomax) and mutual information minimization (Infomin) methods based on Deep Infomax (DIM, Hjelm et al. [154]) as the approximation of Equation (5.12). The s-Infomax method maximizes the mutual information between each edge representation and the label representation, i.e., $\max \mathcal{I}(\boldsymbol{h}_i; y)$. The Infomin method minimizes the mutual information between every pair of edge representations in a hypergraph, i.e., $\min \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j)$. The approximation of Equation (5.12) is:

$$\max \sum_{i=1}^{k} \mathcal{I}(\boldsymbol{h}_i; y) - \sigma \sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j), \tag{5.13}$$

where σ is a scalar weight.

We prove that Equation (5.12) reaches the optimal value when Equation (5.13) reaches the optimal value in Proposition 5.1. We detail the proof in Appendix C.

Proposition 5.1. *Consider an input-output pair $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is a set of features, a function maps $\boldsymbol{x}$ to y in a Markov form $\boldsymbol{x} \rightarrow \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\} \rightarrow y$, where $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\}$ is a set of middle states mapped from $\boldsymbol{x}$, if the $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\}$ makes Equation (5.13) reach the optimal value, it also makes Equation (5.12) reach the optimal value..*

Proposition 5.1 ensures that by optimizing Equation (5.13), we approximately optimize Equation (5.12) so that generate feature interactions that fit the three properties.

5.4.3.3 The Implementation

One difficulty of applying s-Infomax is that the label space only has two states, i.e., 0 and 1, which are discrete and are too small to provide enough output information [225]. Hence, we use the hypergraph representations to represent the label space, extending the two states into a continuous high-dimensional space. Following DIM, we sample the *joint distribution* of $\mathcal{I}(\boldsymbol{h}_i; y)$ as the pair of $\boldsymbol{h}_i$ and a random hypergraph representation $\boldsymbol{c}^+$ from training samples having the same y label. Then, we sample the *marginal*

distribution as $\boldsymbol{h}_i$ and a random hypergraph representation $\boldsymbol{c}^-$ from training samples having the opposite y label. We use a GAN-based objective function for s-Infomax:

$$\begin{aligned}\max \sum\nolimits_i \mathcal{I}(\boldsymbol{h}_i; y) = \max \frac{1}{k} \sum\nolimits_i (\log D_\theta(\boldsymbol{h}_i, \boldsymbol{c}^+) \\ + (1 - \log D_\theta(\boldsymbol{h}_i, \boldsymbol{c}^-))),\end{aligned} \tag{5.14}$$

where $D_\theta \in \mathbb{R}^{2\times d} \to \mathbb{R}$ is a discriminator that distinguishes the joint distribution and the marginal distribution.

In Infomin, we minimize the MI between different hyperedge representations in a hypergraph. We set the sampled joint distribution as $(\boldsymbol{h}_i, \boldsymbol{h}_j)$ that $i \neq j$, and the marginal distribution as $(\boldsymbol{h}_i, \boldsymbol{h}_i)$. To avoid the overfitting issue that the discriminator in Infomin may trivially compare whether two input vectors are the same, inspired by Gao et al. [252], we use a dropout function f_a to prevent each pair of input vectors from being the same. To ensure efficiency, we get one sample from the joint distribution and one sample from the marginal distribution for each node. The objective function for Infomin is also a GAN-based objective function but swaps the joint and marginal distributions:

$$\begin{aligned}\min \sum_{i\neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j) = \max \frac{1}{k} \sum_i (\log D_\omega(f_a(\boldsymbol{h}_i) f_a(\boldsymbol{h}_i)) \\ + (1 - \log D_\omega(f_a(\boldsymbol{h}_i), f_a(\boldsymbol{h}_j)))).\end{aligned} \tag{5.15}$$

Equation (5.14) and Equation (5.15) can be efficiently optimized by training the discriminators D_θ and D_ω through Binary Cross-entropy (BCE) loss [218].

Note that the main difference between s-Infomax and Infomin in implementation is the choice of the joint distribution and marginal distribution that a discriminator to distinguish. The distributions chosen in Infomin are intuitive: we want any two hyperedge representations in a hypergraph to be as different as possible (have less MI).

5.4.4 The Empirical Risk Minimization Function

The empirical risk minimization function of HIRS contains the loss function of recommendation predictions, an L_0 activation regularization, the loss functions of the s-Infomax discriminator and the Infomin discriminator:

$$\begin{aligned}\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\rho}) = \frac{1}{N} \sum_{n=1}^{N} (\mathcal{L}(f_r((G_H)_n; \boldsymbol{\rho}), y_n) + \lambda_1 \left\| \boldsymbol{E}_n' \right\|_0 \\ + \lambda_2 \mathcal{L}_D(D_\theta) + \lambda_3 \mathcal{L}_D(D_\omega)), \\ \boldsymbol{\rho}^*, \boldsymbol{\theta}^*, \boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\rho}} \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\rho}),\end{aligned} \tag{5.16}$$

where $(G_H)_n = (G_H\langle\mathcal{V}, \emptyset\rangle)_n$, λ_1, λ_2 and λ_3 are weight factors, $\mathcal{L}(\cdot)$ and $\mathcal{L}_D(\cdot)$ are both BCE loss, and $\boldsymbol{\rho}^*$, $\boldsymbol{\theta}^*$, $\boldsymbol{\omega}^*$ are final parameters.

5.5 Discussions

In this section, we first show that IHGNN satisfies the statistical interaction principle so that can perform effective feature interaction modeling. Then we show that IHGNN is a general framework for modeling and aggregating arbitrary order feature interactions. Finally, we give a time complexity analysis to show that HIRS is inherently more efficient than existing studies for beneficial feature interaction detection.

5.5.1 IHGNN Satisfying the Statistical Interaction Principle

We first give the definition of arbitrary order statistical interaction following our pairwise statistical interaction definition (Definition 4.2).

Definition 5.2. (**Arbitrary Order Statistical Interaction**) Consider a function $f(\cdot)$ with input variables $\boldsymbol{x} = \{x_1, x_2, ..., x_m\}$. Then, a set of variables $\hat{q} \subseteq \boldsymbol{x}$ is a statistical interaction of function $f(\cdot)$ if and only if there does **not** exist a set of functions $f_{\setminus i}(\cdot)$, where $x_i \in \hat{q}$ and $f_{\setminus i}(\cdot)$ is not a function of x_i, such that

$$f(\boldsymbol{x}) = \sum_{x_i \in \hat{q}} f_{\setminus i}(x_1, x_{i-1}, x_{i+1}, ..., x_m) \tag{5.17}$$

We prove that IGHNN models a set of features as a statistical interaction if they form a hyperedge in the input hypergraph in Proposition 5.3. The proof of Proposition 5.3 is in Appendix D.

Proposition 5.3. *Consider an input-output pair $(\boldsymbol{x}, y)$. Let $G\langle\mathcal{V}, \emptyset\rangle$ be the input of HIRS, where $\mathcal{V} = \boldsymbol{x}$, the IHGNN function (Equation (5.8)) flags a statistical interaction among all the features in $q \subseteq \mathcal{V}$ if $q = \{a_i | \boldsymbol{E}'_{ik} = 1\}_{i \in \mathcal{V}}$, where $\boldsymbol{E}'_k$ is a hyperedge predicted by f_{gen} (Equation (5.6)).*

Note that according to the definition of statistical interaction, if f_{IHGNN} flags a statistical interaction on all features in q, f_{IHGNN} flags a statistical interaction on any $q_s \subseteq q$. This is the reason that we use Infomin to force each hyperedge to contain different feature interactions and prevent multiple feature interactions from merging into one hyperedges.

5.5.2 IHGNN Generalizes Existing Methods

We show that some existing methods can be represented as an IHGNN in specific settings.

5.5.2.1 L_0-SIGN as IHGNN

A hypergraph is an extension of a standard graph. Therefore, GNN-based recommender systems such as L_0-SIGN and Fi-GNN [103] can be easily applied by IHGNN by changing the edge matrix into the incidence matrix form and replacing with the corresponding feature interaction modeling functions, e.g., dot product by an MLP for L_0-SIGN.

5.5.2.2 FM as IHGNN

FM models every pairwise feature interaction by dot product and sums up all the modeling results. We can regard each pairwise feature interaction in FM as a hyperedge linking to two nodes, which forms the $\mathcal{E}^{FM}$ that contains all pairwise node combinations, including all self-loops as the point-wise terms in FM.

Formally, we can write FM as:

$$f_{FM}(G_H\langle\mathcal{V},\mathcal{E}^{FM}\rangle) = \frac{1}{2}(\phi(\{\psi(\{\boldsymbol{V}_i \odot \boldsymbol{V}_j | \boldsymbol{E}^{FM}_{ik} = \boldsymbol{E}^{FM}_{jk} = 1\}_{k\in\mathcal{E}})\})), \tag{5.18}$$

where $\odot$ is dot product and ϕ, ψ are sum operations.

In addition, FM-based extensions can also be achieved based on the above derivation. For example, NFM [108] can be achieved by replacing the dot product and the ϕ by an element-wise production and an MLP, respectively.

5.5.2.3 DeepFM as IHGNN

DeepFM [5] merges the prediction of FM and the prediction of an MLP modeling all the features. We can regard DeepFM as IHGNN by adding a hyperedge linking to all nodes in addition to $\mathcal{E}^{FM}$. Since DeepFM uses different interaction modeling functions in FM and the MLP, the hyperedges are modeled differently according to the edge degree.

5.5.3 Interaction Generation Cost Analysis

HIRS generates k beneficial feature interactions through the hyperedge prediction module f_{gen}, which has the time complexity of $O(mkd)$ for a data sample of m features, with each feature being a d-dimensional vector. The s-Infomax and the Infomin methods help the interaction generation and both have the complexity of $O(kd)$. In addition, the IHGNN module models the generated feature interactions for recommendation with a $O(mkd + md)$ complexity. Therefore, the time complexity of HIRS is $O(mkd)$. Existing studies model or examine all the possible interactions by enumerating them $O(2^m d)$ [17] or stacking pairwise interaction modeling layers $O(m^3 d)$ [11, 253] when considering arbitrary orders. Since k can be set to be a much smaller number (e.g., 40) than m^2 (e.g., 196 when $m = 14$), HIRS achieves a more efficient interaction modeling. We do not include the complexity of the neural network modules in HIRS and the baselines as they are all linear to d. The running time of HIRS and the baselines are detailed in Section 5.7.

5.6 Experiments

In this section, we evaluate the following aspects of our proposed model: (i) the recommendation prediction accuracy of HIRS; (ii) the beneficial feature interactions generation ability of HIRS; (iii) the impact of different components and hyperparameters of HIRS.

5.6.1 Experimental Protocol

5.6.1.1 Datasets

We evaluate HIRS on three real-world datasets in recommender systems: **MovieLens 1M** [254], **Book-crossing** [255], and **MovieLens 25M** [254]:

MovieLens 1M [254] contains users' ratings on movies. Each data sample contains a user and a movie with their corresponding attributes as features. We further collect movies' other features, such as directors and casts from IMDB to enrich the datasets. **Book-crossing** [255] contains users' implicit and explicit ratings of books. Each data sample contains a user and a book with their corresponding features. The reprocessed words in the book titles are also regarded as features of the book.

MovieLens 25M [254] contains movie attribute information and the tags (e.g., gunfight, dragon) that users gave to the movies. We regard the attributes genres, years, and the 10 most relevant tags for movie as features.

Dataset	#Data	#Users	#Items	#Features
MovieLens	1,149,238	5,950	3,514	6,974
Book-Crossing	1,050,834	4,873	53,168	43,244
MovieLens 25M	31,147,118	162,541	62,423	226,122

TABLE 5.1: Dataset statistics.

We transfer the explicit ratings to implicit feedback. We regard the ratings greater than 3 as positive ratings for MovieLens 1M and MovieLens 25M, and regard all rated explicit ratings as positive ratings for Book-crossing due to its sparsity. Then, we randomly select the same number of negative samples equal to the number of positive samples for each user.

We summarize the statistics of the datasets in Table 5.1. Note that although MovieLens 1M and MovieLens 25M are both movie recommendation datasets, they are actually deployed with different features, e.g., MovieLens 25M contains the tag features, while MovieLens 1M does not. In this way, we can evaluate the effectiveness and the robustness of our model when different kinds of features are available in similar scenarios.

5.6.1.2 Baselines

We compare HIRS with state-of-the-art models considering feature interactions for recommendation: The baselines are based on: 1) whether they consider pairwise or high-order feature interactions; 2) whether they perform beneficial feature interaction detection. Table 5.2 summaries this information.

We compare HIRS to various baseline models, including **FM** [74], **AFM** [4], **NFM** [108], **AutoInt** [11], **DeepFM** [5], **xDeepFM** [16], our L_0-SIGN model (Chapter 4), and the following models:

Fi-GNN [103] represents each data sample as a feature graph and models all pairwise interactions in a GNN by the multi-head self-attention method.

DCNv2 [253] leverages deep and cross network to implicitly and explicitly model feature interactions.

AutoFIS [17]: detects limited high-order beneficial feature interactions by setting a gate for each feature interaction. We set the highest order of 3 following the original paper.

AFN [109] leverages logarithmic neural network to learn high-order feature interactions adaptively from data. We set the highest order of 3 following the original paper.

Mthods	High-Order	Beneficial Detection
FM		
AFM		
NFM		
Fi-GNN		
L_0-SIGN		√
AutoInt	√	
DeepFM	√	
xDeepFM	√	
DCNv2	√	
AutoFIS	√	√
AFN	√	√
HIRS	√	√

TABLE 5.2: Summaries of whether the model can consider high-order feature interactions and perform beneficial feature interaction detection.

For all baselines, we use the same MLP settings in all baselines (if use) as our interaction modeling function in HIRS for a fair comparison.

5.6.1.3 Experimental Setup

In the experiments, we use element-wise mean as the linear aggregation function for both $\psi(\cdot)$ and $\phi(\cdot)$. $g(\cdot)$ is a linear regression function. The MLPs for $f_{gen}(\cdot)$ and f_E consist of a fully connected layer of size 64 with a ReLU activation. Both D_θ and D_ω are Bilinear models. The dropout probability of f_a is 0.1. The node embedding size for interaction modeling and edge prediction is 64. The hyperedge number k is 40. We set $\lambda_1 = 0.02$, $\lambda_2 = 1$, $\lambda_3 = 0.1$, and Adam [256] as the optimization algorithm. We randomly split each dataset into training, validation, and test sets with a proportion of 70%, 15%, and 15%, respectively. We evaluate the performance of HIRS and the baseline models using the popular ranking metrics Recall@10, Recall@20, NDCG@10, and NDCG@20.

5.6.2 Overall Performance

Table 5.3, Table 5.4, and Table 5.5 show the results of our model and the baseline models on the MovieLens 1M dataset, the Book-Crossing dataset and the MovieLens 25M dataset, respectively. The results are the average of 10 times run. The best results for each dataset are in bold, and the best baseline results are underlined. The rows *Improv* (standing for Improvements) and *p-value* show the improvement and statistical significance test results (through Wilcoxon signed-rank test) of HIRS to the best baseline results, respectively.

We observe that:

- Our model outperforms all the baselines on the three datasets (up to 5.49%), and the significance test results are all lower than the threshold of 5%, which show that the improvements are significant.
- For the models that only consider pairwise feature interactions, GNN-based models (e.g., Fi-GNN and L_0-SIGN) perform better than other baselines (e.g., NFM). This shows the ability of GNNs in interaction modeling.
- Models considering higher-order feature interactions (e.g., xDeepFM and AFN) perform better than those only consider pairwise interactions (e.g., NFM and Fi-GNN). Thus, considering high-order feature interactions is beneficial for providing better prediction accuracy.
- Detecting beneficial feature interactions is vital in removing noise interactions and improving prediction accuracy (e.g., L_0-SIGN v.s. Fi-GNN).
- Our model combines GNN-based interaction modeling, higher-order interaction modeling, and beneficial feature interaction detection, and achieves the best performance.

	MovieLens 1M			
	R@10	R@20	N@10	N@20
FM	0.8527	0.8996	0.8521	0.8845
AFM	0.8738	0.9176	0.8734	0.8994
NFM	0.8979	0.9318	0.8925	0.9084
Fi-GNN	0.9060	0.9341	0.9091	0.9226
L_0-SIGN	<u>0.9092</u>	<u>0.9386</u>	<u>0.9176</u>	<u>0.9268</u>
AutoInt	0.8991	0.9318	0.9077	0.9118
DeepFM	0.8975	0.9308	0.9074	0.9128
xDeepFM	0.9049	0.9342	0.9134	0.9114
AutoFIS	0.9028	0.9317	0.9096	0.9208
DCNv2	0.9014	0.9311	0.9079	0.9104
AFN	0.9032	0.9333	0.9112	0.9232
HIRS	**0.9545**	**0.9663**	**0.9464**	**0.9500**
Improv.	4.98%	2.95%	3.13%	2.50%
p-value	0.5%	0.5%	0.5%	0.5%

TABLE 5.3: Comparing the prediction performance of HIRS with the baselines in the MovieLens 1M dataset.

	Book-Crossing			
	R@10	R@20	N@10	N@20
FM	0.7802	0.8834	0.8123	0.8539
AFM	0.8084	0.8892	0.8473	0.8779
NFM	0.8625	0.9174	0.8492	0.8843
Fi-GNN	0.8818	0.9262	0.8689	0.8861
L_0-SIGN	0.8817	0.9266	<u>0.8723</u>	0.8914
AutoInt	0.8739	0.9263	0.8636	0.8894
DeepFM	0.8765	0.9245	0.8648	0.8800
xDeepFM	0.8791	0.9249	0.8692	0.8826
AutoFIS	0.8804	0.9227	0.8706	0.8887
DCNv2	0.8773	0.9234	0.8671	0.8858
AFN	<u>0.8821</u>	<u>0.9285</u>	0.8710	<u>0.8917</u>
HIRS	**0.9279**	**0.9584**	**0.9081**	**0.9211**
Improv.	5.19%	3.22%	4.10%	3.30%
p-value	0.5%	0.5%	0.5%	0.5%

TABLE 5.4: Comparing the prediction performance of HIRS with the baselines in the Book-Crossing dataset.

5.6.3 Effectiveness of the Beneficial Properties

To ensure our model can detect beneficial feature interactions that fit the beneficial properties (sufficiency, low-redundancy, and disentanglement), we propose the s-Infomax and Infomin module and the L_0 activation regularization. This section, we evaluate the influence of them on the beneficial feature interaction detection and the prediction accuracy.

We run two variants of HIRS: (i) without s-Infomax and Infomin (*woMI*), and (ii) without the L_0 activation regularization (woL_0). Figure 5.4 shows the distribution of the generated feature interactions on three settings (the full model and the two variants). We observe that when using the variant *woMI*, most of the generated hyperedges are empty (order 0), and a few hyperedges connect to most features (order 12-14). It proves the importance of the three properties on arbitrary order beneficial interaction generation. When using the variant woL_0, most of the generated interactions have relative higher orders than using our full model. This proves that L_0 regularization helps ease the issue of mixed interaction in each hyperedge. Next, we show the prediction accuracies of the two variants and the full model in Figure 5.5. We show the results of Recall@10 and NDCG@10. Similar conclusions can be drawn from Recall@20 and NDCG@20 (the same as the remaining experiments). We can see that both *woMI* and woL_0 reduce the prediction accuracy. This proves that our full model successfully

	MovieLens 25M			
	R@10	R@20	N@10	N@20
FM	0.8331	0.8972	0.9535	0.8791
AFM	0.8484	0.9082	0.8618	0.8845
NFM	0.8604	0.9170	0.8620	0.8903
Fi-GNN	0.8692	0.9124	0.8667	0.8905
L_0-SIGN	0.8729	0.9261	0.8704	0.8915
AutoInt	0.8707	0.9233	0.8692	0.8899
DeepFM	0.8667	0.9213	0.8650	0.8883
xDeepFM	0.8673	0.9226	0.8696	0.8902
AutoFIS	0.8702	0.9228	0.8712	0.8922
DCNv2	0.8678	0.9214	0.8667	0.8890
AFN	0.8743	0.9271	0.8737	0.8938
HIRS	**0.9223**	**0.9545**	**0.8846**	**0.9011**
Improv.	5.49%	2.96%	1.26%	0.81%
p-value	0.5%	0.5%	1.4%	3.7%

TABLE 5.5: Comparing the prediction performance of HIRS with the baselines in the MovieLens 25M dataset.

generates more beneficial feature interactions than the two variants and hence gains the best performance.

5.6.4 Ablation Study on RPC Modules

We evaluate the impact of two key functions, the hyperedge detection function f_{gen} and the non-linear interaction modeling function f_E, in the two modules (the hyperedge prediction module and the IHGNN module) of the recommendation prediction component (RPC), respectively. To evaluate the impact of f_{gen}, we remove f_{gen} and use a fully connected matrix as the hyperedge matrix, i.e., $\boldsymbol{E}' = \mathbf{1}$. We name this variant as *woHP*. To evaluate f_E in HIRS, we replace f_E by a linear function, which simply remove the ReLU activation in f_E. We name this variant as *woNM*.

We summarize the results in Figure 5.6. The variant *woHP* achieves an inferior performance. In this situation, the hyperedge does not indicate a feature interaction, but all the features. Therefore, the interaction modeling function does not model beneficial feature interactions but the mix of all the feature information, which inevitably contains the noise information that decreases the prediction accuracy. The variant *woNM* also achieves an inferior performance. This is because the non-linear interaction modeling helps correctly model the generated feature interactions and hence improves the accuracy. This is consistent with our claim in Chapter 4. We also show the results of the variant with both modifications (*woBoth*). The worst performance it achieves is not

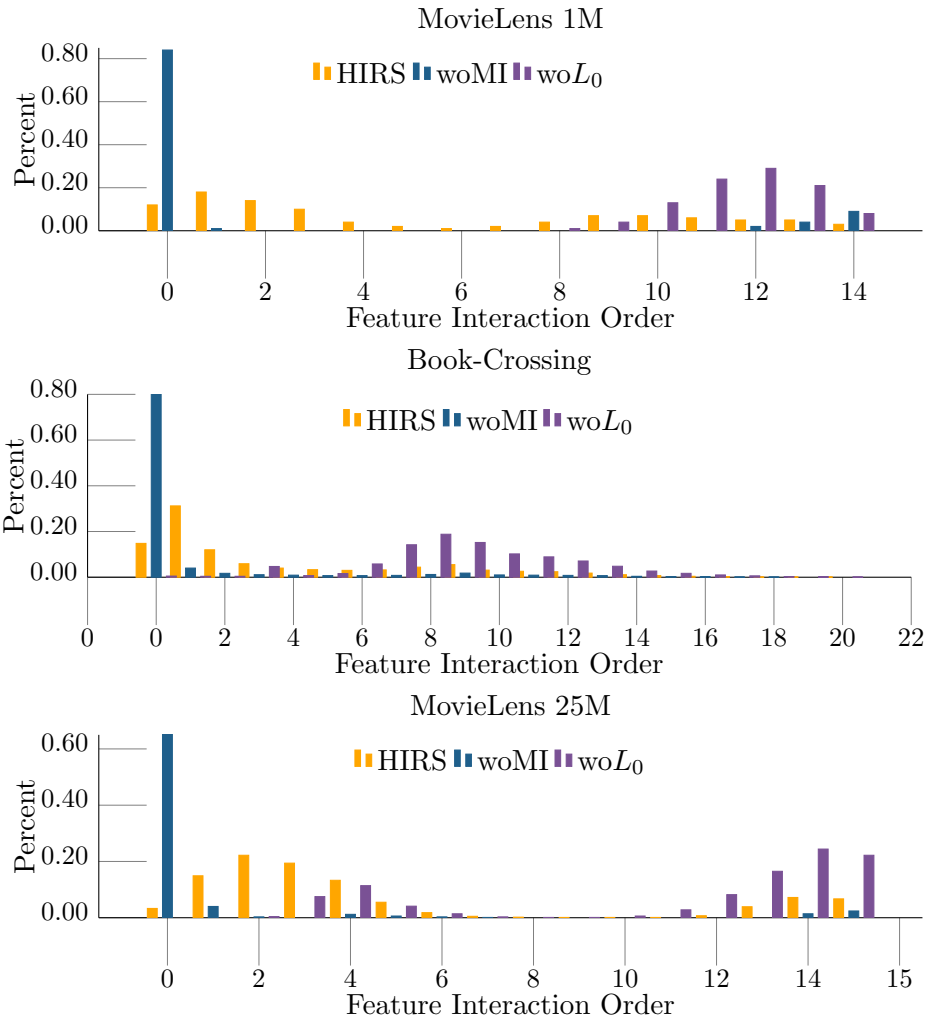


FIGURE 5.4: The distribution of the generated feature interaction orders from HIRS, HIRS without MI, and HIRS without L_0 regularization.

surprising since the whole model under this setting can be regarded as a linear mapping from feature embeddings to the final prediction.

5.7 Running Time Analysis

We have theoretically analyzed the efficiency advantage of HIRS over other baselines that explicitly consider high-order feature interactions. In this section, we experimentally evaluate the running time of HIRS and these baselines. Specifically, we run HIRS and baselines on the MovieLens 1M dataset and record the time consuming they used in each epoch while training. For the baselines, we record the running time of setting different order limitations. We then compare with HIRS that consider arbitrary order feature interactions. Table 5.6 demonstrates the time consuming results. Note that since the running environment of our model and baseline models are not exact the same, the time

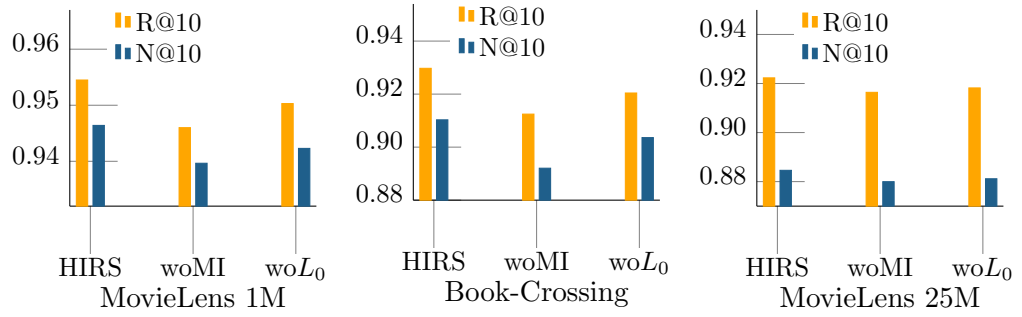


FIGURE 5.5: Comparing the performance of HIRS, HIRS without s-Infomax and Infomin, and HIRS without L_0 regularization.

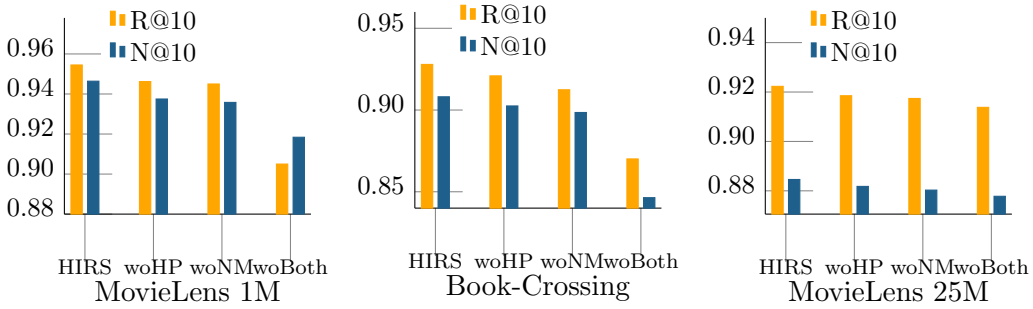


FIGURE 5.6: The recommendation performance of HIRS, HIRS without hyperedge prediction, HIRS without non-linear interaction modeling, and HIRS without both.

comparison results reported are not completely accurate. For example, some baseline models directly call packages for compiling, which is more time efficient. However, we can still gain useful information from the results.

From the table, we can see that the running time of our model approximately equals to baseline models modeling low order feature interactions, which is consistent with our theoretical analysis in Section 5.5.3. Meanwhile, the running time for baseline models increase with the highest order increasing, while HIRS has constant running time. The results proves the efficiency of our model in modeling arbitrary order feature interactions, while baseline models have much higher running time when considering up to arbitrary orders (order 14). In addition, when we detect fewer feature interactions for each data sample, e.g., $k = 20$, the running time is further decreased comparing to that of $k = 40$.

5.7.1 Parameter Study

We evaluate the impact of the number of feature interactions (k), the weight values for L_0 activation regularization (λ_1), s-Infomax (λ_2), and Infomin (λ_3) in Equation (5.16) on the prediction accuracy. Figure 5.7 shows the performance of HIRS using different numbers of edges (k). The prediction accuracy increases when the k increases from 5 to 40, but flattens over 40. This is because that when the edge number is small, some

Method	**Order**						
	2	3	4	5	...	13	14
AutoInt	5.48	6.26	10.96	12.59	...	23.34	24.27
DCNv2	9.40	14.09	18.79	22.71	...	46.19	50.11
xDeepFM	18.79	39.93	59.51	78.30	...	229.42	248.21
HIRS (k=40)				15.16			
HIRS (k=20)				11.38			

TABLE 5.6: Comparing the time (in seconds) of different models for one epoch on the MovieLens 1M dataset. All the models are run on the same machine equipped with a CPU: Intel(R) Core(TM) i7-9700K @ 3.60GHz, and a GPU: GeForce RTX 2080Ti.

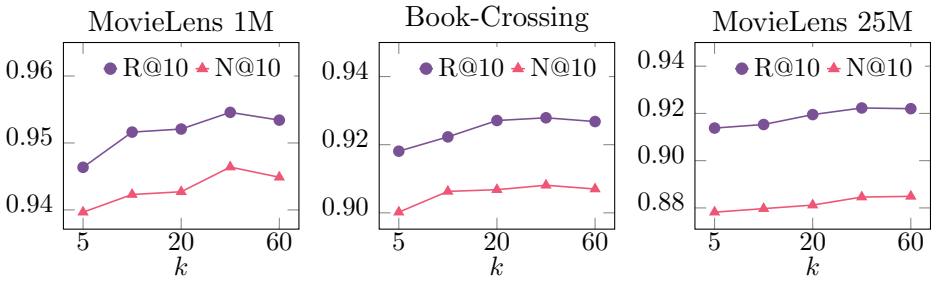


FIGURE 5.7: The recommendation performance of HIRS with different hyperedge numbers.

of the beneficial feature interactions are discarded or mixed in one hyperedge. However, the number of edge slots is enough to contain beneficial feature interactions when $k > 40$. Next, we evaluate different $\lambda_1, \lambda_2, \lambda_3$ values in Equation (5.16) on the MovieLens 1m dataset. As shown in Figure 5.8, every weight has a sweat point that leads to the best performance. For the L_0 regularization weight (λ_1), it balances the sparsity and the useful information retraining of the beneficial feature interactions. s-Infomax and Infomin balance the three properties. The sufficiency and the low redundancy are achieved by s-Infomax, and the disentanglement is achieved by Infomin. For example, a high s-Infomax weight ensures sufficiency and low redundancy, but may harm the disentanglement. Only when the three properties achieve simultaneously we can successfully generate beneficial feature interactions, and leverage them for accurate recommendations.

5.7.2 Case Study on the Detected Interactions

We do a case study to visualize the detected feature interactions. Figure 5.9 shows the detected feature interactions (in an incidence matrix) of a data sample in the MovieLens 1M dataset, where empty ones were removed. Each column is a feature interaction with corresponding features colored. We sort the feature interactions by order and set them with different colors based on their order, e.g., dark blue for second order feature

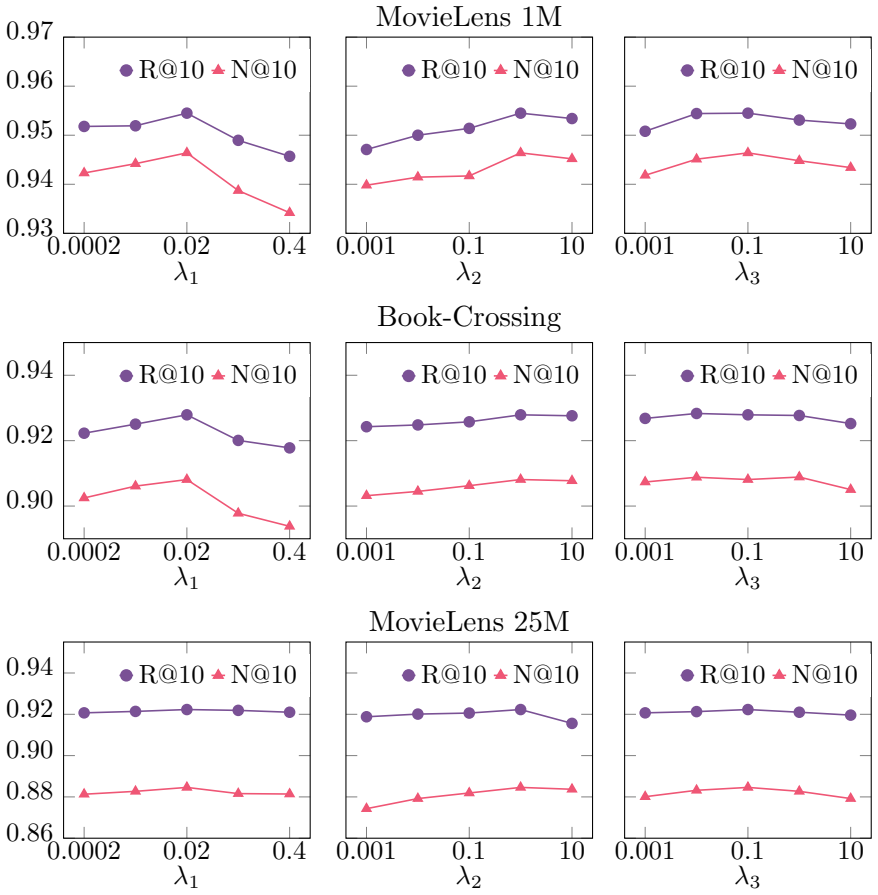


FIGURE 5.8: The performance of different weight values of L_0 activation regularization, s-Infomax, and Infomin.

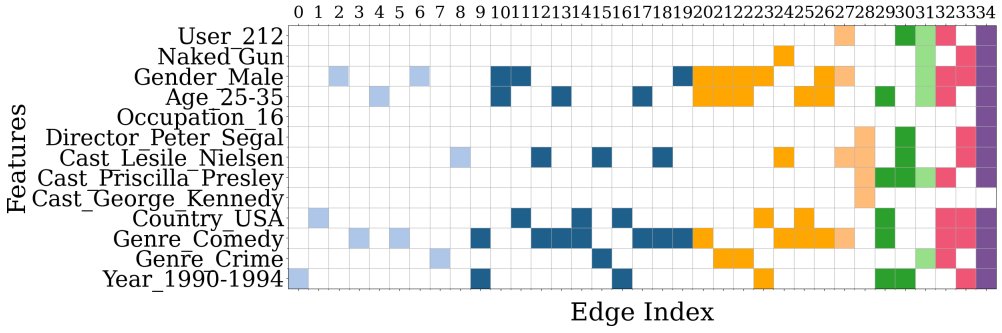


FIGURE 5.9: The detected feature interactions (in an incidence matrix) of a data sample in the MovieLens 1M dataset. Feature interactions are sorted by their orders and columns with the same color represent the interactions of the same order.

interactions and purple for 13 order feature interactions (the number of colored cells in each column indicates the order of the corresponding interaction). From the figure, we observe that: (i) Feature interactions provide potential explanations for the recommendation prediction results. For example, interaction 13 shows the interaction between age and comedy genre, which indicates that users aged 25-35 may prefer comedy movies. (ii) Features such as gender and age participate in many feature interactions, which

shows the importance of these features in the prediction. (iii) User id (*e.g., User_212*) and movie id (e.g., *Naked Gun*) usually appears in high order feature interactions (e.g., interaction 30-34). This indicates that the id features can be used to infer complex preferences or properties. (iv) There are a few duplicated feature interactions, e.g., interactions 13 and 17. It means that these interactions are important. Having them duplicated adds the weights of these interactions in recommendation predictions.

5.8 Summary

In this chapter, we investigate how to detect arbitrary order beneficial feature interactions for recommendation. We represent each data sample as a hypergraph and use a hypergraph neural network-based model named HIRS for recommendation. HIRS achieves superior results by directly generating beneficial feature interactions via hyperedge prediction and performing recommendation predictions via hypergraph classification. We define three properties of beneficial feature interactions, and propose s-Infomax and Infomin methods for effective interaction generation.

In the next chapter, we investigate the third research question of how to categorize and model feature interactions to fit their roles for more accurate recommendation.

Chapter 6

Modeling User and Item Attribute Interactions for Profile Learning and Preference Matching

This chapter is based on the paper that has been published in the International Conference on Research and Development in Information Retrieval (SIGIR) 2021: ***Neural Graph Matching based Collaborative Filtering****. Yixin Su, Rui Zhang, Sarah Erfani, Junhao Gan.*

6.1 Introduction

In this chapter, we explore to categorize and model the interactions between user/item attributes so that fit their roles for recommendation. The motivation is that existing feature interaction-based models simply treat all the attribute interactions equally, and hence, model and aggregate them in the same way. For example, factorization machine [2] models each attribute interaction as a dot product of two embedded vectors and aggregates all the modeling results linearly. GNN-based interaction modeling such as our L_0-SIGN (Chapter 4) models feature interactions equally in a GNN to generate the final predictions.

However, different attribute interactions should have different impacts on the final prediction. Specifically, when the attribute interactions are exploited to perform joint decisions, they could play different roles for recommendation. For example, without loss

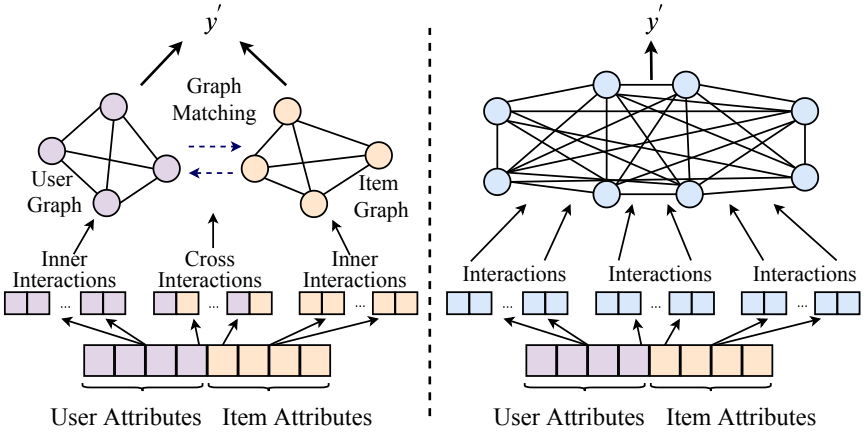


FIGURE 6.1: An illustration of the differences between GMCF (left) and existing work (right). GMCF treats attribute interactions differently in a structure of graph matching, while existing work treats all attribute interactions equally.

of generality, the goal of movie recommendation is to predict the preference of a user (represented by the user's attributes) on a movie (represented by the item's attributes). Therefore, the attribute interactions are naturally divided into two categories based on their roles for recommendation. First, the interactions between user attributes *only* (or between item attributes *only*), e.g., <*male*, *18-24*>, <*Nolan*, *sci-fi*>, serve as user (item) profile learning. These interactions are called *inner interactions*. Second, the interactions between user attributes and item attributes, e.g., <*18-24*, *sci-fi*>, serve as preference matching, such as "whether users in 18-24 years old like sci-fi movies". These interactions are called *cross interactions*. Existing work do not distinguish these two types of interactions. Such unawareness of attribute interaction types inevitably limits the capability of the existing work of leveraging attribute interactions for producing satisfactory joint decisions.

In this work, we explicitly model and aggregate inner interactions and cross interactions in different ways in a graph matching structure. Specifically, we use a graph composed of a user's (an item's) attributes to represent the user (the item). Each attribute is a node, and each pairwise attribute interaction is an edge. Then, we propose a neural Graph Matching based Collaborative Filtering model, called GMCF. Our GMCF uses a GNN to model a user attribute graph and an item attribute graph based on their inner interactions, respectively. Meanwhile, it matches the two attribute graphs at node level based on cross interactions, and makes the final prediction result. Figure 6.1 illustrates the differences between GMCF and existing work (GNN-based work as an example) in modeling and aggregating attribute interactions. As a result, GMCF consider attribute interactions in a way that fits what a recommender system aims to do: learn user and item characteristics (through graph learning), and match users' preference to items based on their characteristics (through graph matching).

We summarize the main contributions of our work as follows:

- We highlight the importance of distinguishing the impacts of attribute interactions on recommendation predictions, and categorize them into inner interactions and cross interactions. These interactions should be treated differently when they play different roles in the final predictions.
- We propose a novel neural graph matching-based model, GMCF, that effectively leverages inner interactions for user/item profile learning (through graph learning), and exploits cross interactions for preference matching (through graph matching). To our best knowledge, we are the first to represent each user and each item as an attribute graph, and model and aggregate attribute interactions in a graph matching structure.
- We conduct extensive experiments. Experimental results show that (i) our model outperforms the state-of-the-art baselines in terms of accuracy; (ii) GMCF can effectively model the two types of attribute interactions and achieves accurate predictions.

6.2 Problem Statement

In this section, we introduce the problem definition and the representations of attribute interactions.

6.2.1 Problem Statement

Since this work aims to categorize user attributes and item attributes, the problem statement has some differences to previous chapters. Here, we show the new problem statement for this work. Denote by $\mathcal{J}^U$ the *universe* of the user attributes and by $\mathcal{J}^I$ the *universe* of the item attributes. An *attribute-value pair* is a name-value pair, denoted by (a, w), where a is the name of the attribute and w is the value on this attribute. For example, $(Male, 1)$ and $(Female, 1)$ mean the user's gender is Male and Female, respectively, where $Male \in \mathcal{J}^U$ and $Female \in \mathcal{J}^U$ are considered as two user attributes. Let D be a set of N training data pairs, denoted by $D = \{(\boldsymbol{x}_n, y_n)\}_{1 \leq n \leq N}$. In each training data $(\boldsymbol{x}_n, y_n) \in D$,

- $\boldsymbol{x}_n$ is a data sample, which consists of:
  - a user attribute set $\{c^U = (a, w)\}_{a \in J_n^U}$ of attribute-value pairs with respect to a set of user attributes $J_n^U \subseteq \mathcal{J}^U$, and

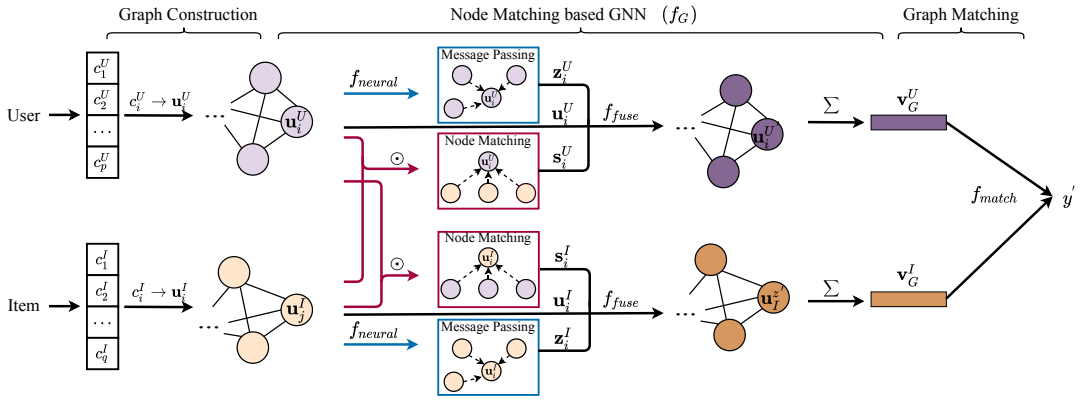


FIGURE 6.2: An overview of the GMCF model.

– an item attribute set $\{c^I = (a, w)\}_{a \in J_n^I}$ of attribute-value pairs with respect to a set of item attributes $J_n^I \subseteq \mathcal{J}^I$.

- $y_n \in \mathbb{R}$ is the implicit feedback of the user on the item.

It should be noted that the number of attribute-value pairs in each data sample may be different, as the information of some attributes may be missing, or may contain *multiple* attributes of the same type, e.g., a movie may belong to multiple genres.

Our Solution. In this work, we propose a *Neural Graph Matching based Collaborative Filtering* (GMCF) model $F_{GMCF}(\boldsymbol{x}_n)$. Our GMCF models the user and the item characteristics as two graphs, respectively, and leverages these user and item graph representations to predict y_n by graph matching techniques.

6.3 Our Proposed Method

In this section, we first describe our model in detail. First, we give an overview of our GMCF model. Next, we unveil the details of each part of the model. Finally, we discuss the relations of our model to existing work and applicable situations.

6.3.1 GMCF Overview

Our proposed model GMCF mainly consists of three modules: (i) User and Item Graph Construction Module, (ii) Node Matching based GNN Module, and (iii) Graph Representation Matching Module. Figure 6.2 shows an overview of the GMCF model. In particular, for an input data sample $\boldsymbol{x} = \{c_i^U = (a, w)\}_{1 \le i \le p} \cup \{c_j^I = (a, w)\}_{1 \le j \le q}$, each module works as follows.

The User and Item Graph Construction Module. GMCF constructs a *user attribute graph* and an *item attribute graph* respectively based on the user and item characteristic specified in the data sample $\boldsymbol{x}$. More specifically, the user attribute graph (resp., item attribute graph) is a *complete graph*, where each node corresponds to an attribute-value pair (a, w) in the user (resp., item) characteristic and is represented by the representation $\boldsymbol{u} = w\boldsymbol{v}$ of the pair, where $\boldsymbol{v} = \text{emb(a)}$ is the d-dimensional vector representation of attribute a.

The Node Matching based GNN Module. For each node i with representation $\boldsymbol{u}_i$ in the user (resp., item) attribute graph, this module first computes the *message passing information* $\boldsymbol{z}_i$ and the *node matching information* $\boldsymbol{s}_i$ with the item (resp., user) attribute graph. It then fuses $\boldsymbol{u}_i$, $\boldsymbol{z}_i$ and $\boldsymbol{s}_i$ to obtain a fused node representation $\boldsymbol{u}'_i$ for the node i. Finally, a graph representation $\boldsymbol{v}_G$ is obtained by aggregating the fused node representation $\boldsymbol{u}'_i$ for all nodes.

The Graph Representation Matching Module. Our GMCF performs a graph matching between the user and item attribute graph representations. The final prediction is obtained from matching result.

Next, we introduce the details of the three modules.

6.3.2 User & Item Graph Construction

We represent each user and each item as an *attribute graph*, with their attributes as nodes and their inner interactions as edges. Specifically, for each data sample $\boldsymbol{x}$, the user attributes are the nodes of the user attribute graph. Each node i that represents an attribute takes the attribute representation $\boldsymbol{u}_i^U$ as the node representation. Therefore, we represent the node set of the user attribute graph as $\mathcal{V}^U = \{\boldsymbol{u}_i^U\}_{i \in \mathcal{J}^U}$. Then, pairwise inner interactions are represented as edges in the graph. In summary, each user attribute graph is represented as $G^U = \langle \mathcal{V}^U, \mathcal{E}^U \rangle$, where $\mathcal{E}^U$ is the edge set that contains all edges in the graph. Note that since we consider all pairwise attribute interactions, the user attribute graph is a complete graph. We perform the same transformation for item attributes and get the item graph $G^I = \langle \mathcal{V}^I, \mathcal{E}^I \rangle$. In practice, we perform the graph construction by simply dividing attribute interactions as edges and matching pairs for different modules, which does not take additional effort than other explicit pairwise interaction modeling methods (e.g., AutoInt, Fi-GNN).

6.3.3 Node Matching based GNN

We propose a node matching based GNN, f_G that considers both inner interactions through message passing and cross interactions through node matching. In this section, we describe how f_G outputs the graph representation through modeling the two types of interactions. Note that since the modeling on user attribute graphs and item attribute graphs are symmetric, the notations for f_G in the following subsections is generic (i.e., we omit the superscript U and I) that can apply to both user and item attribute graphs.

6.3.3.1 Neural Interaction based Message Passing

In our model, the message passing method models the inner interactions for profile learning. We use our proposed SIGN model (Section 4.5.1.3) to perform the graph learning. Specifically, an MLP function $f_{neural} \in \mathbb{R}^{2\times d} \rightarrow \mathbb{R}^d$, takes the embeddings of the two nodes as input and output the interaction modeling results:

$$\boldsymbol{z}_{ij} = f_{neural}(\boldsymbol{u}_i, \boldsymbol{u}_j), \tag{6.1}$$

where $\boldsymbol{z}_{ij}$ is the interaction modeling results of the node pair (i, j).

Then, all the interaction modeling results corresponding to each node will be aggregated as the message passing information. In GMCF, we use the element-wise sum to aggregate the interaction modeling results $\boldsymbol{z}_i = \sum_{j\in N_i} \boldsymbol{z}_{ij}$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ is the message passing results of node i and N_i is a set of node i's neighbors.

f_{neural} explicitly models the interaction information between two attributes, which effectively unifies the interaction modeling for recommendation and the message passing in graph learning. In addition, inner interactions are used for capturing user (item) characteristics and are inherently complex. A high inner interaction result does not mean that the two attributes should be similar when determining the characteristics of a user (an item), which are different from cross interaction model results that reveal similarity (will be discussed in the next section). Therefore, a neural method (e.g., MLP) that non-linearly models the two attributes are desired.

6.3.3.2 Bi-Interaction based Node Matching

GMCF conducts node matching between two graphs through modeling cross interactions. Intuitively, we expect that an attribute c_i^U will have a high matching score with an attribute c_j^I if c_i^U shows a high preference on c_j^I. For example, if male users prefer sci-fi movies, the node matching score of the node pairs <*male*, *sci-fi*> should be

high. In collaborative filtering, if a user attribute has a high preference for an item attribute, their embeddings should be similar after training. To achieve this, we use the Bi-interaction [108] for node matching, which keeps the monotonically increasing correlation between interaction modeling results and the attribute similarities. As a result, if a user attribute has a high matching score on an item attribute, they have similar attribute representations. Specifically, the Bi-interaction algorithm models the attribute interactions as:

$$\boldsymbol{s}_{ij} = \boldsymbol{u}_i \odot \hat{\boldsymbol{u}}_j, \tag{6.2}$$

where $\boldsymbol{u}_i$ is the embedding of node i in one graph, $\hat{\boldsymbol{u}}_j$ is the embedding of node j in the other graph, $\boldsymbol{s}_{ij}$ is the node matching result of two nodes from different graphs, and $\odot$ is the element-wise product.

Similar to the message passing, we use an element-wise sum to aggregate the node matching result between a node in one graph and all nodes in the other graph. The aggregated node matching result is $\boldsymbol{s}_i = \sum_{j \in \hat{\mathcal{V}}} \boldsymbol{s}_{ij}$, where $\hat{\mathcal{V}}$ is the node set of the other attribute graph and $\boldsymbol{s}_i$ is the aggregated node matching result of node i.

6.3.3.3 Information Fusing

Besides message passing results, GMCF further considers the node matching results to capture the node-level matching information while generating the fused node representations. Specifically, the node fusing function $f_{fuse} \in \mathbb{R}^{3\times d} \rightarrow \mathbb{R}^d$ takes the initial node representation $\boldsymbol{u}_i$, the message passing results $\boldsymbol{z}_i$ and the node matching results $\boldsymbol{s}_i$ as input, and get the fused node representation. Formally, we have $\boldsymbol{u}'_i = f_{fuse}(\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i)$, where $\boldsymbol{u}'_i$ is the fused node representation of node i.

We can use any method for f_{fuse}, e.g., element-wise addition or recurrent neural network. Through testing, we find that the recurrent neural networks perform the best. We use GRU, an effective recurrent neural network model, as the function f_{fuse}. Therefore, f_{fuse} considers $[\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i]$ as the input sequence and the final output hidden layer of GRU is the fused node representation.

6.3.3.4 Node Representation Aggregation

The fused node representations of each graph are aggregated as the graph representation. We use the element-wise sum to aggregate the node representations. In summary, the f_G in GMCF is:

$$f_G(G, \hat{\mathcal{V}}) = \sum_{i \in \mathcal{V}} \boldsymbol{u}'_i, \tag{6.3}$$

6.3.4 Graph Matching

While performing graph matching, we get the vector representations of the user attribute graph and the item attribute graph through f_G, respectively:

$$\boldsymbol{v}_G^U = f_G(G^U, \mathcal{V}^I), \quad \boldsymbol{v}_G^I = f_G(G^I, \mathcal{V}^U). \tag{6.4}$$

We match the two graphs by using dot product as f_{match} on the two graph representations to get the predicted output $y' = {\boldsymbol{v}_G^U}^\top \boldsymbol{v}_G^I$.

6.3.5 Model Training

While training, we use the L_2 norm to regularize all the parameters of GMCF. Therefore, the empirical risk minimization function of GMCF minimizes a loss function and an L_2 norm:

$$\begin{aligned} \mathcal{R}(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(F_{GMCF}(\boldsymbol{x}_n;\boldsymbol{\theta}), y_n) + \lambda(\|\boldsymbol{\theta}\|_2), \\ \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}), \end{aligned} \tag{6.5}$$

where F_{GMCF} is the prediction function of GMCF that outputs y', $\mathcal{L}(\cdot)$ corresponds to a loss function (e.g., binary cross-entropy loss), $\boldsymbol{\theta}$ are all parameters in GMCF, and $\boldsymbol{\theta}^*$ are the final parameters.

6.4 Discussions

6.4.1 Relation to Existing Work and Discussion

Our model has close relations to attribute graph-based recommender systems and FM. In this section, we discuss the relations to the two types of recommender systems. Then, we show that our model can apply to situations when the user or item attributes are not available, which happens in practice.

6.4.1.1 Relation to L_0-SIGN

Our L_0-SIGN model are two attribute graph-based models that treat all attribute interactions equally in one graph. GMCF can be considered an extension of this framework. Specifically, if GMCF considers all interactions equally, e.g., all are modeled as

cross interactions, and puts the match function f_{match} linear, e.g., $f_{match}(\boldsymbol{v}_G^U, \boldsymbol{v}_G^I) = sum(\boldsymbol{v}_G^U) + sum(\boldsymbol{v}_G^I)$, where $sum(\cdot)$ sums up all the elements of the input vector. As a result, GMCF has a similar framework to L_0-SIGN.

6.4.1.2 Relation to FM

Different from graph-based models, FM aggregates all interaction modeling results linearly (i.e., sum up). GMCF can also be regarded as an extension of FM. Besides the modifications that reduce to the framework of Fi-GNN and L_0-SIGN (every interaction is modeled by element-wise product), if the fusing function f_{fuse} is linear, e.g., $f_{fuse}(\boldsymbol{u}_i, \boldsymbol{z}_i, \boldsymbol{s}_i) = \boldsymbol{u}_i + \frac{1}{2}\sum_{j \in V^a/\{i\}} \boldsymbol{s}_{ij}$, where $\mathcal{V}^a = \mathcal{V}^U \cup \mathcal{V}^I$, $\boldsymbol{s}_{ij} = \boldsymbol{u}_i \odot \boldsymbol{u}_j$, the prediction function becomes:

$$\begin{aligned} y' &= sum(\sum_{i \in \mathcal{V}^U}(\boldsymbol{u}_i + \frac{1}{2}\sum_{j \in \mathcal{V}^a/\{i\}} \boldsymbol{s}_{ij})) + sum(\sum_{i \in \mathcal{V}^I}(\boldsymbol{u}_i + \frac{1}{2}\sum_{j \in \mathcal{V}^a/\{i\}} \boldsymbol{s}_{ij})) \\ &= \sum_{i \in \mathcal{V}^a} sum(\boldsymbol{v}_i) \cdot w_i + \sum_{i \in \mathcal{V}^a}\sum_{j \in \mathcal{V}^a, j > i} \boldsymbol{v}_i^\top \boldsymbol{v}_j \cdot w_i \cdot w_j. \end{aligned} \tag{6.6}$$

Equation (6.6) is similar to the prediction function of FM in its original paper (Equation 1 in [2]) with the weight parameter ω_i of each attribute i being $sum(\boldsymbol{v}_i)$ and the bias term ω_0 being omitted (Note that $sum(\boldsymbol{u}_i \odot \boldsymbol{u}_j) = \boldsymbol{v}_i^\top \boldsymbol{v}_j \cdot w_i \cdot w_j$).

6.4.1.3 When the User or Item Attributes are Unavailable

In some situations, the user or item attributes are not available. GMCF is also applicable in these situations. Specifically, if one type of attribute is unavailable, e.g., user attributes, the user attribute graph becomes a single node (user ID). Then, no inner interaction modeling is in the user attribute graph, i.e., $\boldsymbol{v}_G^U = \boldsymbol{u}'_u = f_{fuse}(\boldsymbol{u}_u, \boldsymbol{s}_u)$, where $\boldsymbol{u}_u$ is the node representation of user ID. If both user and item attributes are unavailable, both the two attribute graphs are represented by a single node. Then, there is no inner interaction modeling but only the cross interaction modeling between user ID and item ID. We will evaluate our model in these scenarios in Section 6.5.3.3.

6.4.2 Time Complexity Analyses

Although GMCF contains graph construction operations, it does not take additional effort than other explicit pairwise interaction modeling methods (e.g., AutoInt, Fi-GNN) since we just need to regard feature pairs as edges and matching pairs. In GMCF, for

a data sample of p user attributes and q item attributes, for each node $\boldsymbol{u}_i^U$ in the user graph, the message passing complexity (neural interaction modeling and aggregation) is $O(p(M_1+d))$, where $O(M_1)$ is the complexity of f_{neural}; the node matching complexity (Bi-interaction modeling and aggregation) is $O(q(d+d))$. Therefore, the node will result in a fused node representation with the complexity of $O(p(M_1+d)+2qd+M_2)$, where $O(M_2)$ is the complexity of f_{fuse}. The complexity of a node in the item graph has the similar form $O(q(M_1+d)+2pd+M_2)$. The total complexity (with graph matching procedure) is $O((p^2+q^2)(M_1+d)+2pqd+(p+q)(M_2+d))$ (the node matching procedures in the two graphs are shared). Except for the differences between interaction modeling methods, we performs interaction modeling on every pair of attributes, which is the same as the methods like AutoInt and Fi-GNN. Therefore, our models' complexity is at the same magnitude as these models as long as $O(M_1)$ is not too high.

6.5 Experiments

In this section, we conduct experiments to evaluate GMCF. We focus on three questions: (i) what is the ability of GMCF in providing accurate recommendations comparing to other baselines that can consider user attributes and item attributes; (ii) what is the effectiveness of each component in GMCF on providing accurate predictions; (iii) whether the learned attribute embeddings reveal collaborative relations and semantic meaning in their latent space.

6.5.1 Experimental Protocol

We first describe the datasets and the baselines used in our experiments. Then, we illustrate the experimental set-up in detail.

6.5.1.1 Datasets

We run GMCF and baselines on three datasets, that contain both user and item attributes. Table 6.1 shows their statistic information. Below are the descriptions of the datasets. We compare with

MovieLens 1M [254] contains users' ratings on movies. Each data sample contains a user and a movie with their corresponding attributes. We further collect movies' other attributes, such as directors and casts from IMDB to enrich the datasets.

TABLE 6.1: Dataset statistics. The *attr.* refers to "attributes".

Dataset	**#Data**	**#User**	**#Item**	**#User** *attr.*	**#Item** *attr.*
MovieLens 1M	1,149,238	5,950	3,514	30	6,944
Book-Crossing	1,050,834	4,873	53,168	87	43,157
Taobao	2,599,463	4,532	371,760	36	434,254

Book-crossing [255] contains users' implicit and explicit ratings of books. Each data sample contains a user and a book with their corresponding attributes. The reprocessed words in the book titles are also regarded as attributes of the book.

Taobao [10] is a dataset that collects the log of click on display advertisement displayed on the website of Taobao. Each log contains a user with corresponding attributes such as gender and age level, a displayed advertisement with attributes such as category and brand of the advertised item.

MovieLens 1M and Book-crossing have the same data prepossessing procedure as that in Section 5.6.1.1: First, we transfer the explicit ratings to implicit feedback. We regard the ratings greater than 3 as positive ratings for MovieLens 1M and regard all rated explicit ratings as positive ratings for Book-crossing due to its sparsity. Then, we randomly select the same number of negative samples equal to the number of positive samples for each user. To ensure the datasets' quality, we select the users with more than 10 positive ratings for MovieLens 1M and have more than 20 positive ratings for Book-crossing and Taobao.

6.5.1.2 Baselines

We compare our model with competitive baselines that can take user attributes and item attributes into account, including **FM** [2], **AFM** [4], **NFM** [108], **W&D** [9], **DeepFM** [5], **AutoInt** [11] and L_0**-SIGN**.

For all baselines, we set the MLP structure (if used) for interaction modeling to be the same as the MLP for inner interaction modeling in our model for a fair comparison.

6.5.1.3 Experimental Set-Up

We randomly split each dataset into training, validation, and test set for each user with a ratio of 6:2:2. The validation set is only used to decide the best parameter setting, and the test set is only used to evaluate the models. Unless otherwise specified, we

use the following hyper-parameter settings: the node representation dimension is 64 (i.e., $d = 64$); the number of hidden layers for MLP is 1 and the number of units for the hidden layer is $4d$; the learning rate is 1×10^{-3}; the λ for the regularization is 1×10^{-5}. We use the binary cross entropy as the loss function and use Adam [256] as the optimization algorithm. We use the area under the curve (AUC), Logloss, and NDCG@k as evaluation metrics to evaluate the performance of our model and baseline models. AUC and Logloss are frequently used in the tasks with implicit feedback and NDCG@k is a frequently used metric to evaluate the top-k recommendation. We set k to 5 and 10. All the experiments are repeated 5 times and the average results are recorded.

6.5.2 Overall Performance

We compare the performance of GMCF with the baselines. Table 6.2, Table 6.3, and Table 6.4 show the prediction performance of each model on the MovieLens 1M dataset, the Book-Crossing dataset and the Taobao dataset, respectively. The best results for each dataset are in bold, and the best baseline results are in underline. The rows *Improv* and *p-value* show the improvement and statistical significance test results (through Wilcoxon signed-rank test) of GMCF and the best baseline results, respectively. From these tables, we observe that:

- GMCF outperforms all the baselines significantly, with the p-value of all metrics rejecting the null hypothesis with a level of significance of $\alpha = 5\%$. These results prove the ability of GMCF in effectively analyzing the structural information of user and item attributes for accurate predictions.
- The models that explicitly model attribute interactions (GMCF, AutoInt, Fi-GNN, L_0-SIGN) gain better prediction accuracy than other models. This indicates that the explicit interaction modeling is promising to extract useful information from attributes and attribute interactions for accurate predictions.
- FM and AFM perform worse than the other models. This is because FM and AFM do not use neural methods, but solely rely on the dot product to extract information from attribute interactions. Hence, to model more complicated interactions, sophisticated methods (e.g., MLP) are required. Accordingly, our model leverages MLP to model the inner interactions inside the user attribute graph and the item attribute graph, which provides powerful interaction modeling ability for accurate predictions.
- The GNN-based models (GMCF, Fi-GNN, L_0-SIGN) gain better prediction accuracies than other models. It shows the ability of GNNs in modeling attribute interactions for

recommendation. GMCF further models attribute interactions in a structure of graph matching, which is more suitable for recommendation and gains better performance than Fi-GNN and L_0-SIGN.

	MovieLens 1M			
	AUC	Logloss	NDCG@5	NDCG@10
FM	0.8761	0.4409	0.8143	0.8431
AFM	0.8837	0.4323	0.8270	0.8676
NFM	0.8985	0.3996	0.8486	0.8832
W&D	0.9043	0.3878	0.8538	0.8869
DeepFM	0.9049	0.3856	0.8510	0.8848
AutoInt	0.9034	0.3883	0.8619	0.8931
Fi-GNN	0.9063	0.3871	0.8705	0.9029
L_0-SIGN	0.9072	0.3846	0.8849	0.9094
GMCF	**0.9127**	**0.3789**	**0.9374**	**0.9436**
Improv	0.61%	1.48%	5.91%	4.28%
p-value	1.09%	0.25%	0.25%	0.25%

TABLE 6.2: Summary of the performance in comparison with baselines on the MovieLens 1M dataset.

	Book-Crossing			
	AUC	Logloss	NDCG@5	NDCG@10
FM	0.7417	0.5771	0.7616	0.8029
AFM	0.7541	0.5686	0.7820	0.8258
NFM	0.7988	0.5432	0.7989	0.8326
W&D	0.8105	0.5366	0.8048	0.8381
DeepFM	0.8127	0.5379	0.8088	0.8400
AutoInt	0.8130	0.5355	0.8127	0.8472
Fi-GNN	0.8136	0.5338	0.8094	0.8522
L_0-SIGN	0.8163	0.5274	0.8148	0.8629
GMCF	**0.8228**	**0.5233**	**0.8671**	**0.8951**
Improv	0.80%	0.78%	6.42%	3.73%
p-value	0.46%	0.25%	0.25%	0.25%

TABLE 6.3: Summary of the performance in comparison with baselines on the Book-Crossing dataset.

6.5.3 Study of Neural Graph Matching

In this section, We evaluate the neural graph matching of GMCF. We focus on (i) the effectiveness of modeling inner and cross interactions using different methods; (ii) the

	Taobao			
	AUC	Logloss	NDCG@5	NDCG@10
FM	0.6171	0.2375	0.0812	0.1120
AFM	0.6282	0.2205	0.0872	0.1240
NFM	<u>0.6550</u>	0.2122	0.0997	0.1251
W&D	0.6531	0.2124	0.0959	0.1242
DeepFM	<u>0.6550</u>	<u>0.2115</u>	0.0974	0.1243
AutoInt	0.6434	0.2146	0.0924	0.1206
Fi-GNN	0.6462	0.2131	0.0986	0.1241
L_0-SIGN	0.6547	0.2124	<u>0.1006</u>	<u>0.1259</u>
GMCF	**0.6679**	**0.1960**	**0.1112**	**0.1467**
Improv	1.96%	7.32%	10.53%	16.52%
p-value	0.25%	0.25%	0.25%	0.25%

TABLE 6.4: Summary of the performance in comparison with baselines on the Taobao dataset.

comparison of using different algorithms as the fusing function; (iii) the performance of GMCF when the user and item attributes are not available. We omit the results of Logloss and NDCG@5, which show similar trends with AUC and NDCG@10, respectively.

6.5.3.1 Evaluation of Inner and Cross Interaction modeling

We evaluate the inner and cross interaction modeling methods. Specifically, we focus on three questions: 1) what is the effectiveness of node-level graph matching (i.e., modeling cross interactions)? 2) whether we should use different methods to model the two types of interactions? 3) whether sophisticated nonlinear algorithms are always suitable for modeling the two types of interactions?

We run GMCF with different interaction modeling methods. For clear demonstration, we use the pair $\langle$*inner interaction model, cross interaction model*$\rangle$ to indicate the method combinations in each variation. For example, we represent the original GMCF as $\langle$*MLP, Bi*$\rangle$ to indicate an MLP-based message passing and a Bi-interaction-based node matching.

We run GMCF using the combination $\langle$*MLP, None*$\rangle$, where *None* indicates that we do not perform node-level matching (i.e., no cross interaction modeling). Then, we run two variations that use the same method to model all interactions: $\langle$*Bi, Bi*$\rangle$, $\langle$*MLP, MLP*$\rangle$. Note that the second variation uses the same MLP (i.e., the same neural architecture

with shared parameters). Finally, we further run two variations that use different methods: $\langle MLP1,\ MLP2 \rangle$ (the same architecture with different parameters) and $\langle Bi,\ MLP \rangle$. Note that MLP1 and MLP2 have the same architecture but are trained on inner and cross interactions, respectively. Figure 6.3 shows the results of using different variations. Book-crossing shows similar trend with MovieLens 1M (the same as the remaining figures).

From the figures, we can answer the above three questions:

- For question 1), $\langle MLP,\ None \rangle$ gains the worst performance. The reason is that the attribute graphs are inherently complex. Fusing this information only in the last step makes it difficult to match users' preferences on items through attributes. The node matching method provides the explicit attribute communication through the cross interactions between the two graphs, which results in a fine-grained analysis for a more accurate preference matching.
- For question 2), the combinations ($\langle Bi,\ Bi \rangle$ and $\langle MLP,\ MLP \rangle$) gain relatively worse results than the original setting ($\langle MLP,\ Bi \rangle$). This is because the inner interactions and cross interactions are inherently different that should be modeled differently that fit their roles. The inner interactions are used for profile learning and do not indicate the similarity information (i.e., two attributes having strong interaction does not mean that they are similar). Meanwhile, in collaborative filtering, two attributes are expected to show similarity information if their cross interaction is strong (high preference). The Bi-interaction algorithm fits for the requirement of cross interactions.
- For question 3), even using different methods, $\langle MLP1,\ MLP2 \rangle$ and $\langle Bi,\ MLP \rangle$ still gain worse results than the original GMCF. Although an MLP seems more powerful than the Bi-interaction algorithm, MLP cannot to effectively learn the measurement for the cross interactions. The results show that sophisticated algorithms such as MLP are expected to model the inner interactions, and similarity measurements such as the Bi-interaction algorithm are expected to model the cross interactions. Therefore, it is not surprising that the reversed version of GMCF ($\langle Bi,\ MLP \rangle$) gains the worst results among all variations.

6.5.3.2 Evaluation of the Fusing Method

The fusing method in GMCF aggregates the message passing results and the node matching results to get the fused node representation. We evaluate the effectiveness of using different algorithms as the fusing algorithm. Except the GRU, we further use

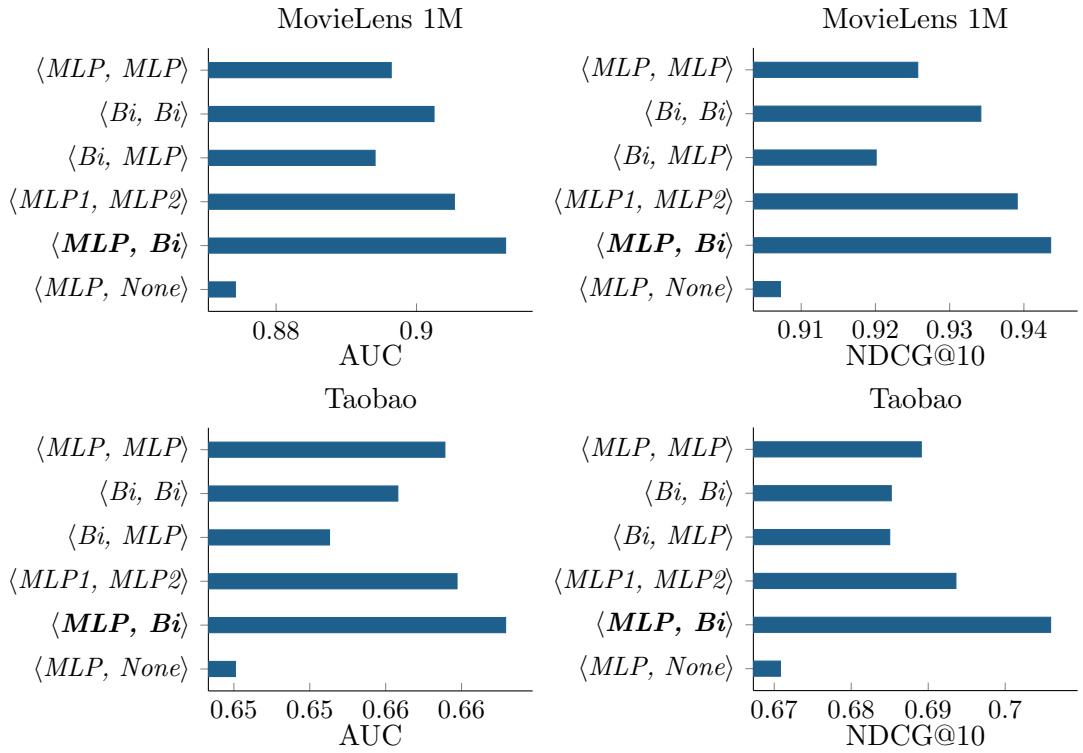


FIGURE 6.3: The comparison of using different inner interaction and cross interaction modeling combinations.

TABLE 6.5: The performance of using different fusing algorithms.

	MovieLens 1M		**Book-Crossing**		**Taobao**	
	AUC	NDCG@10	AUC	NDCG@10	AUC	NDCG@10
SUM	0.9042	0.9404	0.8190	0.8904	0.6583	0.1383
MLP	0.9022	0.9389	0.8156	0.8920	0.6636	0.1421
GRU	**0.9127**	**0.9436**	**0.8228**	**0.8951**	**0.6679**	**0.1467**

the element-wise sum on the three vectors as the fused node representation (SUM) and the MLP that concatenates the vectors as input and outputs the fused node representation (MLP). We use the MLP that has one hidden layer, with the number of neurons being $4d$.

Table 6.5 shows the experimental results of using the three algorithms as the fusing method. From the table, we can see that using GRU results in the best performance on all datasets. It shows the ability of GRU to effectively aggregate the message passing information and the node matching information into the fused node representation for accurate predictions. Summing up the results (*SUM*) gains the worst results in most situations, which indicates that the message passing and node matching information are complex. Powerful algorithms are required to fuse them.

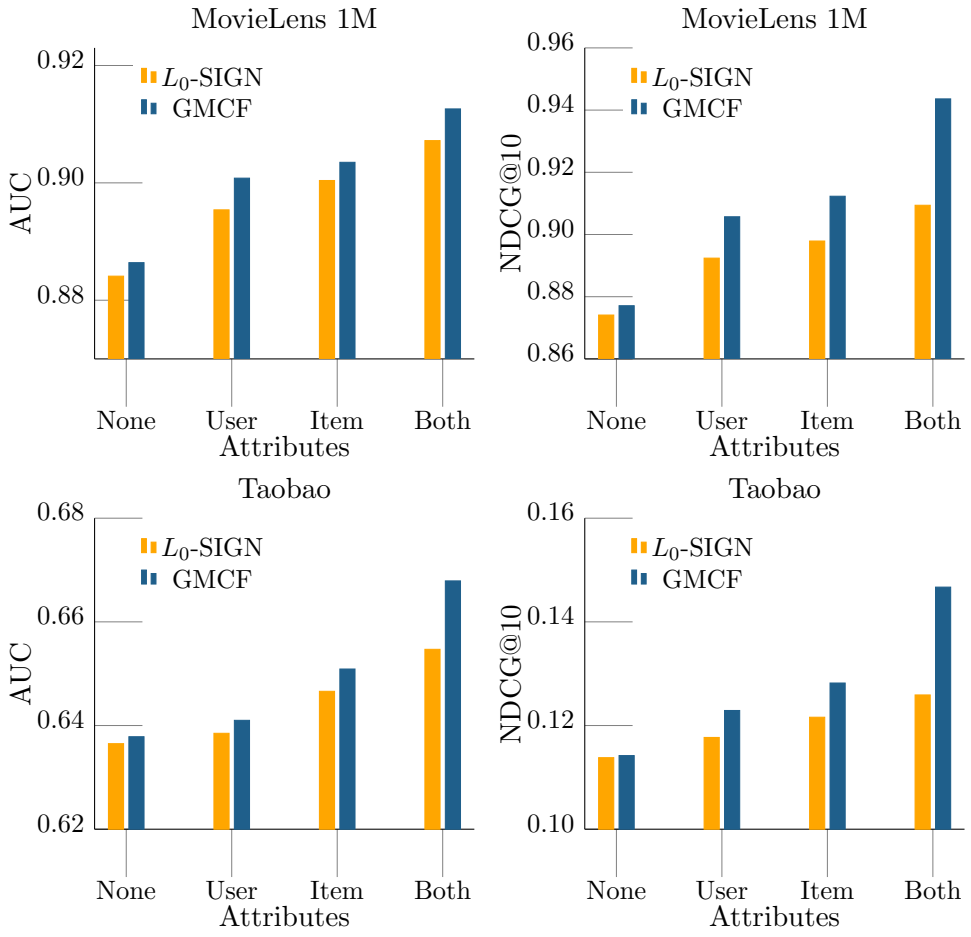


FIGURE 6.4: The comparison when using different attributes.

6.5.3.3 Evaluation of the User and Item Attributes

In some situations, user or item attributes are not available. We evaluate how GMCF performs in these situations. GMCF is a flexible framework that is applicable when the user or item attributes are not available. In these situations, the user (or item) graphs are reduced into a single node indicating user (or the item) ID. We run GMCF and the best baseline L_0-SIGN on the situations that user or item attributes are not available. Figure 6.4 shows the results. Specifically, in the x-axis, "*None*" indicates neither user or item attributes are available, "*User*" indicates only user attributes are available, "*Item*" indicates only item attributes are available, and "*Both*" indicates both user and item attributes are available.

From the figure, we observe that:

- Both models perform better when user and item attributes become available (from left to right). It shows that both user and item attributes are useful for performance gain. Although the performance gain from user attributes (*User*) seems not that significant

compared to item attributes (*Item*), the user attributes provide useful information for potential explanations of the prediction results, which will be discussed in Section 6.5.5.

- The two models have similar performance when no attributes are available (*None*). However, GMCF performs much better when user and item attributes are available. The performance gain compared to L_0-SIGN when the user and item attributes are available resulted from the generated graph matching structure, which captures more useful structural information for accurate predictions.

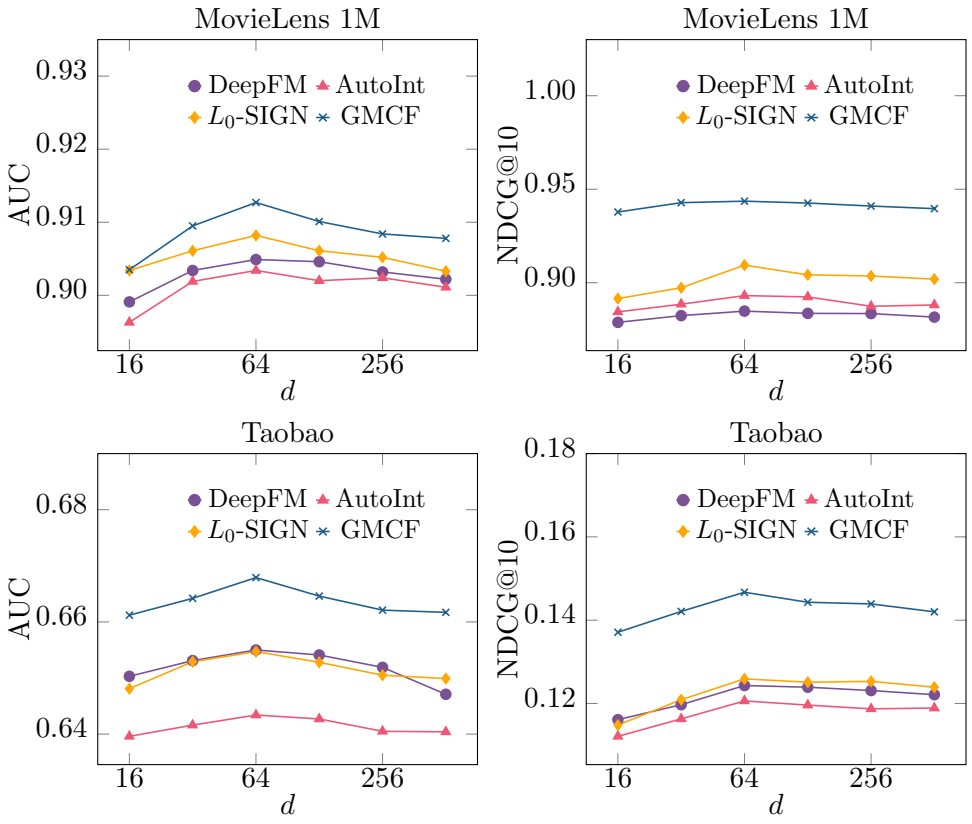


FIGURE 6.5: The performance of GMCF and selected baselines in different node (attribute) embedding dimensions.

6.5.4 Parameter study

In this section, we evaluate GMCF with different hyper-parameter settings. Specifically, we evaluate the node representations' dimension and the depth of the MLP used in our model.

TABLE 6.6: The performance of GMCF on using different MLP depths.

	MovieLens 1M		**Book-Crossing**		**Taobao**	
	AUC	NDCG@10	AUC	NDCG@10	AUC	NDCG@10
GMCF-0	0.9025	0.9354	0.8051	0.8869	0.6608	0.1341
GMCF-1	**0.9127**	**0.9436**	**0.8228**	**0.8951**	**0.6679**	**0.1467**
GMCF-2	0.9109	0.9429	0.8182	0.8924	0.6662	0.1429
GMCF-3	0.9080	0.9411	0.8178	0.8905	0.6645	0.1395
GMCF-4	0.9071	0.9377	0.8153	0.8907	0.6650	0.1393

Figure 6.5 shows the results of our model and best-performed baselines on different node representation dimensions (d). From the figure, we observe that GMCF constantly outperforms baselines on different node representation dimensions, which shows the robustness of our model in delivering superior prediction accuracy. Then, when the dimension is 64, our model and most of the baselines gain the best performance. This indicates that a highger dimension does not necessarily result in better prediction accuracy. This is because that a larger dimension means more parameters to fit, and thus is prone to cause the overfitting problem.

Then, we evaluate how the different number of hidden layers in the MLP affects our model's performance. In our model, we use an MLP with 1 hidden layer to analyze the inner interactions while message passing. Now we evaluate our model with different number of layers. Specifically, we run our model with $0, 1, 2, 3, 4$ hidden layers respectively (GMCF-0,...,GMCF-4). Note that GMCF-0 means that the MLP only performs a linear transformation from the node representations to the interaction modeling results. We use the same number of units ($4d$) for each hidden layer.

Table 6.6 shows the results of using different MLP layers in GMCF. We can see that using the MLP with 0 hidden layer (GMCF-0) gains much worse results than other settings. It shows that a powerful non-linear algorithm helps extract useful information from inner interactions than a linear transformation. This result is consistent with the results in section 6.5.3.1. When the number of the hidden layer is 1, GMCF gains the best performance. This illustrates that deeper MLP does not necessarily increase the performance due to the overfitting [5, 108], and one hidden layer is enough in our models to analyze the inner interactions.

6.5.5 Case Study

In this section, we conduct case studies to evaluate whether our model learns collaborative information between attributes and whether the attributes show semantic meaning that provide potential explanations of the predictions. Specifically, we use the learned attribute embeddings from the MovieLens 1M dataset. We first calculate the embeddings' cosine similarities between user age groups (e.g., 1-18, 19-24) and between movie genres. Then, we calculate the node matching results between the the two types of attributes. Figure 6.6 shows the similarity results and Figure 6.7 shows the node matching results. Note that the darker the color, the higher the similarity or the node matching value. The age labels indicate the age groups (e.g., *age_18* means age 18-24).

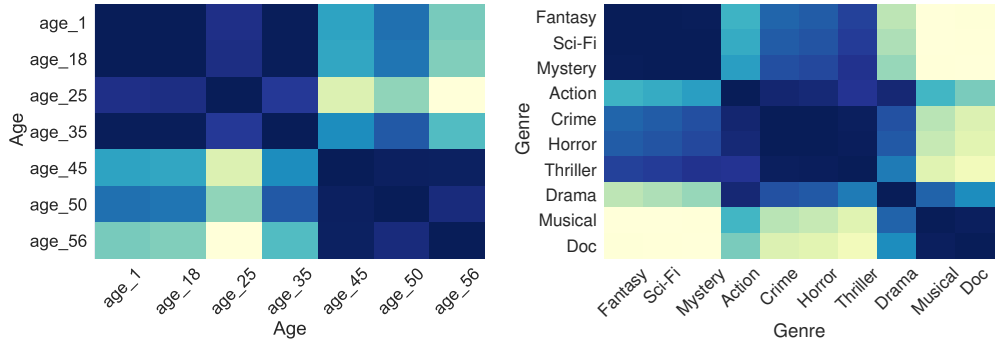


FIGURE 6.6: The case studies on the MovieLens 1M dataset: *Left*: the embedding similarities between user age groups. *Right*: the embedding similarities between movie genres.

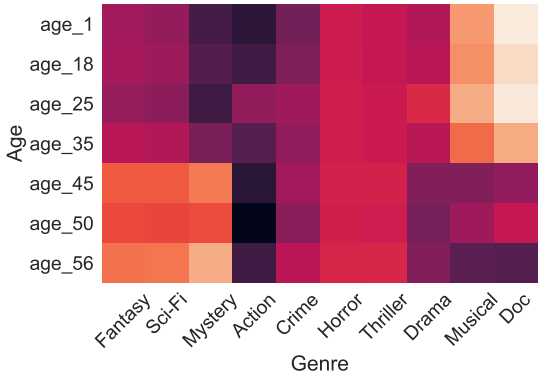


FIGURE 6.7: The case studies on the MovieLens 1M dataset: the node matching (preference) results between the ages groups and the genres.

From Figure 6.6, we can see that similar user attributes and similar item attributes have similar embeddings after training. For example, in the left figure, the age group attributes are clearly divided into two groups at the age of 45, which means that the similar age attributes are grouped. It indicates that the same age group users may have a similar preference for movies in terms of age. In the right figure, Fantasy, Sci-fi,

and Mystery have similar embeddings in the latent space. Semantically, these genres are similar. Similar results are also shown for Crime, Horror, and Thriller. The above observations show that similar attributes are successfully learned to have similar embeddings in GMCF.

Figure 6.7 shows the node matching between age groups and genres. We observe that different age groups have different preferences for movie genres. For example, younger users (under 45) have higher node matching values (preference) on fantasy, sci-fi, and mystery movies more than older users (over 45). In comparison, the older users seem to like musical and Documentary (*Doc*) movies more. These observations show our model's ability to provide potential explanations about the prediction results at attribute level, e.g., we recommend *Interstellar* (a sci-fi movie) to a 16-year-old user because younger people have a high chance to prefer a sci-fi movie.

6.6 Summary

User and item attribute interactions provide useful information in recommender systems for accurate predictions. Existing studies treat all the attribute interactions equally. In this chapter, we identify two types of attribute interactions: inner and cross interactions. We propose a neural Graph Matching based Collaborative Filtering (GMCF) model. The GMCF model exploits the two types of interactions for different purposes in a graph matching structure. Specifically, GMCF explicitly performs user and item profile learning with the inner interactions, and performs preference matching for recommendation based on the cross interactions. Experimental results show that our GMCF model is effective and outperforms all state-of-the-art baselines in terms of accuracy on three widely used datasets.

However, another type of feature is widely explored in recommender systems: context feature. We believe that context features are informative in inferring intrinsic and extrinsic factors, which are two important factors that jointly influence users' decisions to interact with items. In the next chapter, we explore how we to context features to effectively learn intrinsic and extrinsic factors for more accurate recommendation predictions.

Chapter 7

Modeling Context Feature Interactions for Intrinsic and Extrinsic Factor Learning

This chapter is based on the paper that has been submitted to the Conference on Neural Information Processing Systems (NeurIPS) 2022: ***Intrinsic-Extrinsic Disentangled Recommendation with Context-aware Contrastive Learning****. Yixin Su, Wei Jiang, Fangquan Lin, Cheng Yang, Sarah Monazam Erfani, Junhao Gan.*

7.1 Introduction

In this chapter, we explore how to better leverage context feature interactions for more accurate recommendations. Specifically, we aim to learn the factors that jointly affect users' decisions. For example, in the scenario of app recommendation, a user's decision to take Uber (a taxi calling app) to work can be motivated by the heavy rain on that day. On the other hand, Uber is chosen over other taxi calling apps as the user might find this app's user interface more convenient.

Psychological research has revealed two basic types of intents (i.e., factors influencing the user's perspective) for an individual's decision making: intrinsic intent and extrinsic intent [257–259]. Intrinsic intent is an internal factor for inherent satisfaction, which is often stable for an individual (e.g., choosing Uber over other apps). On the contrary, extrinsic intent is a contextual factor motivated by the environment, which is often varied in different contexts [257] (e.g., taking a taxi to work due to heavy rain). From the user's perspective, learning intrinsic and extrinsic intents (i.e., factors) can be beneficial to

recommendation tasks. Meanwhile, users and items are usually learned symmetrically in recommender systems [260, 261]. Therefore, we also consider intrinsic and extrinsic factors from the item's perspective. The four factors jointly affect users' decision making.

However, the full potential of these factors has been rarely explored by the existing work in recommender systems. Some studies neglect the various factors of users and items. The final prediction mainly relies on learning unified representations of the users and the items [101, 262, 263]. With the intrinsic and extrinsic factors entangled behind each decision, the real factors that drive the decision may be incorrectly inferred, resulting in a suboptimal recommendation. Other studies learn several disentangled representations and treat each representation as a factor [186, 264]. However, the factors are learned implicitly in a purely data-driven way with no theoretical guarantee. Therefore, it is still a challenge to effectively learn and incorporate intrinsic and extrinsic factors for recommender systems.

In this work, in order to tackle this challenge, we first formally define intrinsic and extrinsic factors for recommendation tasks. Then, following these definitions, we propose an Intrinsic-Extrinsic Disentangled Recommendation (IEDR) model containing two modules, a recommendation prediction (RP) module, and a contrastive intrinsic-extrinsic disentangling (CIED) module. IEDR takes the information of a user and an item as input, and combines them with context information, to generate intrinsic and extrinsic factor representations for both the user and the item. CIED module ensures the success of the factor learning. It contains two components: a contrastive learning component that helps learn a context-invariant intrinsic representation, and a mutual information minimization-based component that disentangles the intrinsic and extrinsic representations. The two components jointly help learn intrinsic and extrinsic factors that correspond to the factor definitions.

In this work, we made the following contributions:

- To the best of our knowledge, this is the first work to define intrinsic and extrinsic factors for recommender systems with respect to context. The two types of factors are also generated from the perspectives of both users and items for accurate recommendation.
- We propose IEDR to explicitly learn disentangled intrinsic and extrinsic factor representations. A context-invariant contrastive learning component and a mutual information minimization-based disentangling component are proposed to ensure the success of factor inferring.

- Experimental results show that IEDR significantly outperforms state-of-the-art methods in terms of recommendation accuracy through learning intrinsic and extrinsic factor representations.
- We conduct theoretical analyses to demonstrate the connection of context-invariant contrastive learning with information theory.

7.2 Preliminaries

7.2.1 Variational Contrastive Log-ratio Upper Bound (vCLUB) of Mutual Information

Suppose we aim to learn an intrinsic representation $\boldsymbol{o}_{in}^u$ and an extrinsic representation $\boldsymbol{o}_{ex}^u$, where their mutual information is minimized. In the vCLUB method [265], a variational distribution $q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u;\boldsymbol{\theta}_1^u)$ of parameter $\boldsymbol{\theta}_1^u$ (e.g., an MLP) is proposed to predict the extrinsic factor given the intrinsic factor. Then, the vCLUB-based mutual information upper bound can be derived as:

$$\mathcal{I}_{\text{vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u) := \mathbb{E}_{p(\boldsymbol{o}_{in}^u,\boldsymbol{o}_{ex}^u)}[\log q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u)] - \mathbb{E}_{p(\boldsymbol{o}_{in}^u)p(\boldsymbol{o}_{ex}^u)}[\log q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u)], \tag{7.1}$$

where $p(\boldsymbol{o}_{in}^u,\boldsymbol{o}_{ex}^u)$ is the joint distribution and $p(\boldsymbol{o}_{in}^u)p(\boldsymbol{o}_{ex}^u)$ are is the marginal distribution.

vCLUB performs mutual information estimation and minimization in two steps iteratively. In the first step, to ensure Equation (7.1) holds as the upper bound, $\boldsymbol{\theta}_1^u$ is trained to make the log-likelihood function $\mathcal{L}_{appr}(u,c) := \frac{1}{N}\sum_{i=1}^N \log q_1^u((\boldsymbol{o}_{ex}^u)_i|(\boldsymbol{o}_{in}^u)_i)$ maximized (Theorem 3.2 of [265]). In the second step, $\boldsymbol{\theta}_1^u$ is frozen, and other parameters (e.g., the parameters to generate $\boldsymbol{o}_{in}^u$ and $\boldsymbol{o}_{ex}^u$) are trained to minimize $\mathcal{I}_{\text{vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u)$ so that the mutual information is minimized.

7.2.2 Related Work

In this section, we briefly discuss existing works related to two techniques we used in this work, contrastive learning and factor disentanglement for recommender systems, and discuss the relations between our work and these works.

7.2.2.1 Contrastive Learning

Contrastive learning has achieved great success in computer vision [222, 224, 225, 266, 267] and neural language processing [252, 268–270]. Recently, contrastive learning has

attracted attention in recommender systems. Yao et al. [271] conduct contrastive learning on users and items, respectively, on a two-tower framework to learn robust user and item representations. In addition, Wu et al. [188] propose a contrastive learning framework on a user-item bipartite graph to capture robust high-degree relationships between users and items. Lin et al. [272] and Jiang et al. [273] leverage contrastive learning to eliminate popularity bias. We propose the first method that learns intrinsic factor representations that are invariant to context through a contrastive learning approach.

7.2.2.2 Factor Disentanglement for Recommender Systems

Recent studies prove that, in a prediction task, disentangling factors that affect the final prediction will result in superior performance [274–277]. For recommender systems, Ma et al. [264] learn latent factors that affect users' decisions by learning a set of prototypes; and each prototype representation is regarded as one latent factor. Wang et al. [186] learn disentangled user-item interaction representation with a user-item bipartite graph. Each representation is a combination of latent factor representations. However, these models only ensure that the learned factor representations are disentangled, but cannot guarantee whether they really refer to important factors. Our IEDR model learns disentangled intrinsic and extrinsic factors that are inspired by their definitions in psychology, and empirical experiments demonstrate that learning these factors can produce more accurate recommendations.

7.3 Problem Statement and Definitions

In this work, we can consider user, item and context features. Let $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{C}$ denote the user set, item set and context set, respectively. Similar to that in Section 6.2.1, each user $u \in \mathcal{U}$ consists of p user features-value pair $u = \{(a_i, w_i)\}_{1 \leq i \leq p}$ (e.g., user ID, gender). Each item $v \in \mathcal{V}$ is represented by q item features $v = \{(a_i, w_i)\}_{1 \leq i \leq q}$ (e.g., branch, color). Each context $c \in \mathcal{C}$ consists of m context features $c = \{(a_i, w_i)\}_{1 \leq i \leq m}$ to describe the context state when a user select an item (e.g., weather, daytime).

Let a dataset $\mathcal{D}$ contains N instances of (u, v, c) with a corresponding label $y \in \{1, 0\}$, indicating whether the user u select the item v in the context c. The recommendation task can be formulated as predicting the probability $y' = p(u, v, c)$ of the selection. In our proposed IEDR model, the intrinsic factor $\boldsymbol{o}_{in}$ and the extrinsic factor $\boldsymbol{o}_{ex}$ are explicitly inferred for both users and items, and jointly leveraged to perform the prediction.

Next, we formally define intrinsic and extrinsic factors from the users' perspective. Similar definition can be drawn from the items' perspective.

Definition 7.1. (**Intrinsic Factor and Extrinsic Factor**) Consider a user u and a set $\mathcal{C}$ of contexts; an **intrinsic factor** of user u is a factor that is invariant to the contexts in $\mathcal{C}$, i.e., $f_{in}(u, c) = f_{in}(u, c')$, where f_{in} is a function to learn intrinsic factor representations, and c and c' are two arbitrary different contexts in $\mathcal{C}$. On the other hand, an **extrinsic factor** of user u is a factor that changes w.r.t. $\mathcal{C}$, i.e., there exist contexts c and c' in $\mathcal{C}$ such that $f_{ex}(u, c) \neq f_{ex}(u, c')$, where f_{ex} learns extrinsic factor representations.

7.4 Our Proposed Method

The overview of our IEDR model is visualized in Figure 7.1. More specifically, our proposed IEDR model consists of the following modules:

- A recommendation prediction (RP) module that takes a user and an item as input, and combines them with contexts, to generate intrinsic and extrinsic factor representations for both the user and the item. The predicted probability y' is then generated jointly by these representations.
- A contrastive intrinsic-extrinsic disentangled (CIED) module that is applied both the user and item sides to support the intrinsic and extrinsic factor learning. The module contains two components: a context-invariant contrastive learning component, and a disentangling component. Together, the two components ensure the learned factors satisfy Definition 7.1.

7.4.1 Recommendation Prediction Module

The recommendation prediction (RP) module is a symmetric structure that generates user intrinsic and extrinsic factor representations $(\boldsymbol{o}^u_{in}, \boldsymbol{o}^u_{ex})$ from the user side, and generates item intrinsic and extrinsic factor representations $(\boldsymbol{o}^v_{in}, \boldsymbol{o}^v_{ex})$ from the item side. On the user side, we first generate a user representation and a context representation based on user features and context features, respectively. Here we construct a user graph leveraging the inner interactions of users (Chapter 6). In the same way, we construct a context graph. We use two statistical interaction graph network (SIGN) models (Section 4.5.1.3) to learn the user graph and the context graph to generate the the user (graph) representation $\boldsymbol{u} \in \mathbb{R}^d$ and the context (graph) representation $\boldsymbol{c} \in \mathbb{R}^d$, respectively. Then, a factor generation function $f^u_{ie}(\boldsymbol{u}, \boldsymbol{c}) : \mathbb{R}^{2\times d} \rightarrow \mathbb{R}^{2\times d}$ (e.g., a neural network) takes the user representation and the context representation as input, and simultaneously generates a user intrinsic representation $\boldsymbol{o}^u_{in}$ and a user extrinsic representations

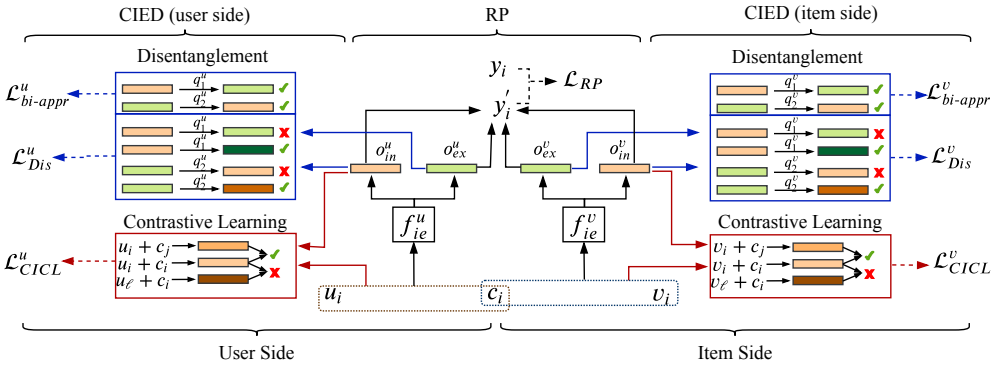


FIGURE 7.1: An Overview of IEDR. It is a symmetric structure on the user side and the item side. The middle part (the black arrows) is the recommendation prediction (RP) module (Section 7.4.1). It generates factor representations ($\boldsymbol{o}_{in}$ and $\boldsymbol{o}_{ex}$) for producing the recommendation prediction y'. The side parts are two contrastive intrinsic-extrinsic disentanglement (CIED) modules. Each CIED includes a context-invariant contrastive learning component (the red arrows, Section 7.4.2.1), and a disentangling component (the blue arrows, Section 7.4.2.2) to ensure the success of the factor learning. The losses generated through the modules ($\mathcal{L}_{RP}, \mathcal{L}_{CICL}, \mathcal{L}_{bi\text{-}appr}, \mathcal{L}_{Dis}$) will be optimized as a two-step multi-task training (Section 7.4.3).

$\boldsymbol{o}^u_{ex}$. Here, the output is a $2d$-dimensional vector, with the first d-dimensional terms as $\boldsymbol{o}^u_{in}$ and the rest as $\boldsymbol{o}^u_{ex}$. Also, the same model structure is adopted on the item side. We use a different SIGN function to generate the item (graph) representation $\boldsymbol{v} \in \mathbb{R}^d$, while using the same context representation as that on the user side. A factor generation function for an item $f^v_{ie}(\boldsymbol{v}, \boldsymbol{c})$ obtains the item's intrinsic factor representation $\boldsymbol{o}^v_{in}$ and extrinsic factor representation $\boldsymbol{o}^v_{ex}$.

Finally, we leverage the four factor representations to jointly perform the prediction: $y' = f_{pred}(\boldsymbol{o}^u_{in}, \boldsymbol{o}^u_{ex}, \boldsymbol{o}^v_{in}, \boldsymbol{o}^v_{ex})$. We linearly combine the intrinsic and extrinsic representations and use the dot product as the prediction function: $f_{pred}(\boldsymbol{o}^u_{in}, \boldsymbol{o}^u_{ex}, \boldsymbol{o}^v_{in}, \boldsymbol{o}^v_{ex}) = (\boldsymbol{o}^u_{in} + \boldsymbol{o}^u_{ex})^\top (\boldsymbol{o}^v_{in} + \boldsymbol{o}^v_{ex})$. A cross-entropy loss function is adopted to minimize the prediction error: $\mathcal{L}_{\text{RP}}(u, v, c) := -y \log(y') + (1 - y) \log(1 - y')$.

7.4.2 Contrastive Intrinsic-Extrinsic Disentangling Module

While the RP module can generate factor representations, solely using this module cannot correctly distinguish intrinsic representations from extrinsic ones. To address this, we propose a contrastive intrinsic-extrinsic disentangled (CIED) module. It is conducted on both the user and the item sides. In the following, we describe the CIED on the user side only. The module on the item side has the same structure, only with different parameters related to items. CIED consists of a *context-invariant contrastive learning component* and a *disentangling component*. The contrastive learning component learns intrinsic representations that are invariant in different contexts. The disentangling component leverages a mutual information minimization task to disentangle the intrinsic

and extrinsic representations. In the following, we describe these two components in detail.

7.4.2.1 Context-Invariant Contrastive Learning Component

We propose a context-invariant contrastive learning component to learn the intrinsic representations. More specifically, we maximize the agreement between the intrinsic representation pairs generated from the same user with different contexts (positive pairs), and at the same time minimize the agreement between the intrinsic representation pairs generated from the same context with different users (negative pairs). For ease of understanding, we represent the intrinsic representations with the subscript $(\boldsymbol{o}^u_{in})_{ij}$ if it is generated through user u_i and context c_j, i.e., $(\boldsymbol{o}^u_{in})_{ij} = f^u_{ie}(\boldsymbol{u}_i, \boldsymbol{c}_j)$. For each data sample $(u_i, v_i, c_i) \in \mathcal{D}$, we calculate an objective function based on InfoNCE [268]:

$$\mathcal{L}^u_{\text{CICL}}(u_i, c_i) := -\log \frac{\exp\left(\text{sim}((\boldsymbol{o}^u_{in})_{ii}, (\boldsymbol{o}^u_{in})_{ij})/\tau\right)}{\sum_{u_\ell \in \mathcal{U}} \exp\left(\text{sim}((\boldsymbol{o}^u_{in})_{ii}, (\boldsymbol{o}^u_{in})_{\ell i})/\tau\right)}, \tag{7.2}$$

where c_j is an arbitrary context, $\text{sim}(\cdot)$ is cosine similarity, and τ is a temperature value.

The objective function is intuitive: one user should have the same intrinsic factor in different contexts, while different users can have their individual interests (different intrinsic factors).

Implementation. To optimize Equation (7.2), we need to generate c_j ($c_i \neq c_j$). We adopt a simple method to perform random sampling from the contexts in the same batch. Alternatively, we propose a dropout-based method inspired by [252], which justifies that a vector with a large dropout rate can be considered a new vector. Therefore, we can apply dropout with a relatively large rate (e.g., 0.5) to generate a new context representation. In practice, we integrate the two context generating methods to generate c_j for each c_i. Meanwhile, we need to iterate over all users to generate $(\boldsymbol{o}^u_{in})_{\ell i}$ (the intrinsic representations generated from an arbitrary user u_ℓ and context c_i), which is prohibitive when the user number is large. Here we randomly sample L users from the same batch to generate the negative intrinsic representations $(\boldsymbol{o}^u_{in})_{\ell i}$, where $\ell = 1, 2, \cdots, L$. We use the categorical cross-entropy to optimize Equation (7.2) following [268].

7.4.2.2 Disentangling Component

We then perform an intrinsic-extrinsic factor disentangling via minimizing the mutual information between the $\boldsymbol{o}^u_{in}$ and $\boldsymbol{o}^u_{ex}$ generated from $f^u_{ie}(\boldsymbol{u}, \boldsymbol{c})$. Inspired by the vCLUB

method [265], we minimize their mutual information via estimating a vCLUB-based upper bound. According to our observations in Section 7.6.5.2, the asymmetric property of the vCLUB may lead to a less robust and inferior intrinsic-extrinsic disentangling. Further discussions on this drawback of vCLUB can be found in Section 7.5.3. Therefore, we propose a simple yet effective bidirectional extension of vCLUB for symmetry, which is more robust and achieves better disentanglement. In the bidirectional vCLUB, two variational distributions (e.g., approximated via neural networks) $q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u;\boldsymbol{\theta}_1^u)$ and $q_2^u(\boldsymbol{o}_{in}^u|\boldsymbol{o}_{ex}^u;\boldsymbol{\theta}_2^u)$ are proposed with parameters $\boldsymbol{\theta}_1^u$ and $\boldsymbol{\theta}_2^u$, to predict the two types of factors, respectively. Then a bidirectional vCLUB-based mutual information upper bound can be obtained as:[1]

$$\begin{aligned}\mathcal{I}_{\text{bi-vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u) := \frac{1}{2}\Big(&\mathbb{E}_{p(\boldsymbol{o}_{in}^u,\boldsymbol{o}_{ex}^u)}[\log q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u)] - \mathbb{E}_{p(\boldsymbol{o}_{in}^u)p(\boldsymbol{o}_{ex}^u)}[\log q_1^u(\boldsymbol{o}_{ex}^u|\boldsymbol{o}_{in}^u)] \\ &+ \mathbb{E}_{p(\boldsymbol{o}_{in}^u,\boldsymbol{o}_{ex}^u)}[\log q_2^u(\boldsymbol{o}_{in}^u|\boldsymbol{o}_{ex}^u)] - \mathbb{E}_{p(\boldsymbol{o}_{ex}^u)p(\boldsymbol{o}_{in}^u)}[\log q_2^u(\boldsymbol{o}_{in}^u|\boldsymbol{o}_{ex}^u)]\Big).\end{aligned} \tag{7.3}$$

Therefore, by minimizing the bidirectional upper bound $\mathcal{I}_{\text{bi-vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u)$ as above, we minimize the mutual information between $\boldsymbol{o}_{in}^u$ and $\boldsymbol{o}_{ex}^u$.

Implementation. The optimization of the disentangling component is conducted in two steps iteratively. In the first step, we estimate the upper bound via training $\boldsymbol{\theta}_1^u$ and $\boldsymbol{\theta}_2^u$ to minimize the loss function $\mathcal{L}_{bi\text{-}appr}^u(u_i,c_i) := -\frac{1}{2}\Big(\log q_1^u\big((\boldsymbol{o}_{ex}^u)_{ii}|(\boldsymbol{o}_{in}^u)_{ii}\big) + \log q_2^u\big((\boldsymbol{o}_{in}^u)_{ii}|(\boldsymbol{o}_{ex}^u)_{ii}\big)\Big)$. Following [265], we use the mean squared error to evaluate q_1^u and q_2^u. In the second step, we freeze $\boldsymbol{\theta}_1^u$ and $\boldsymbol{\theta}_2^u$, and minimize the mutual information by training other parameters to minimize the upper bound $\mathcal{L}_{Dis}^u(u_i,c_i) = \mathcal{I}_{\text{bi-vCLUB}}\big((\boldsymbol{o}_{in}^u)_{ii};(\boldsymbol{o}_{ex}^u)_{ii}\big)$.

7.4.3 Multi-task Training

We perform a two-step multi-task training to minimize the empirical risk of IEDR. The two steps run alternatively until convergence. In the first step, we freeze all parameters except for $\boldsymbol{\theta}_1^u,\boldsymbol{\theta}_2^u,\boldsymbol{\theta}_1^v$, and $\boldsymbol{\theta}_2^v$, where $\boldsymbol{\theta}_1^v,\boldsymbol{\theta}_2^v$ are the parameters of $q_1^v(\boldsymbol{o}_{ex}^v|\boldsymbol{o}_{in}^v;\boldsymbol{\theta}_1^v)$ and $q_2^v(\boldsymbol{o}_{in}^v|\boldsymbol{o}_{ex}^v;\boldsymbol{\theta}_2^v)$ of the disentangling component on the item side. Then, we minimize $\mathcal{R}(\boldsymbol{\theta}_1^u,\boldsymbol{\theta}_2^u,\boldsymbol{\theta}_1^v,\boldsymbol{\theta}_2^v) = \frac{1}{N}\sum_{i=1}^N\big(\mathcal{L}_{bi\text{-}appr}^u(u_i,c_i) + \mathcal{L}_{bi\text{-}appr}^v(v_i,c_i)\big)$. In the second step, we freeze $\boldsymbol{\theta}_1^u,\boldsymbol{\theta}_2^u,\boldsymbol{\theta}_1^v$, and $\boldsymbol{\theta}_2^v$, and minimize the following function:

$$\begin{aligned}\arg\min\mathcal{R}(\boldsymbol{\omega}) = \frac{1}{N}\sum_{i=1}^N\Big(&\mathcal{L}_{\text{RP}}(u_i,v_i,c_i) + \lambda_1\big(\mathcal{L}_{\text{CICL}}^u(c_i,u_i) + \mathcal{L}_{\text{CICL}}^v(c_i,v_i)\big) \\ &+ \lambda_2\big(\mathcal{L}_{Dis}^u(u_i,c_i) + \mathcal{L}_{Dis}^v(v_i,c_i)\big)\Big),\end{aligned} \tag{7.4}$$

[1] $\mathcal{I}_{\text{bi-vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u)$ is the average of two vCLUB-based upper bounds of different directions. Therefore, it is obvious that $\mathcal{I}_{\text{bi-vCLUB}}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u)$ is still an upper bound of $\mathcal{I}(\boldsymbol{o}_{in}^u;\boldsymbol{o}_{ex}^u)$.

where $\mathcal{L}^v_{bi\text{-}appr}$, $\mathcal{L}^v_{\text{CICL}}$, and $\mathcal{L}^v_{Dis}$ are the losses on the item side, λ_1 and λ_2 are weight factors, and $\boldsymbol{\omega}$ are all the trainable parameters except for $\boldsymbol{\theta}^u_1, \boldsymbol{\theta}^u_2, \boldsymbol{\theta}^v_1$, and $\boldsymbol{\theta}^v_2$.

7.5 Discussions

This section provides theoretical analyses of our model, including the connection of the context-invariant contrastive learning component with information theory, and the guarantee to exclude a trivial solution in intrinsic-extrinsic disentangling.

7.5.1 Context-invariant Contrastive Learning in Information Theory

In this section, we analyze the context-invariant contrastive learning from the perspective of information theory. As formally defined in Theorem 7.2, optimizing Equation (7.2) is equivalent to perform mutual information maximization between the intrinsic and user representations and simultaneously perform the mutual information minimization between the intrinsic and context representations. The theorem for the item side can be derived in the same fashion. The proof can be found in Appendix E.

Theorem 7.2 (Equivalence of contrastive loss $\mathcal{L}^u_{CICL}$)**.** *Optimizing the contrastive loss is equivalent to:*

$$\arg\min \sum_{i=1}^{N} \mathcal{L}^u_{CICL}(u_i, c_i) = \arg\max\Big(\mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) - \mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{c})\Big). \tag{7.5}$$

7.5.2 Preventing a Trivial Solution in CIED

The two components in the CIED module jointly ensure the success of the intrinsic and extrinsic factor representation learning. However, CIED may fall into a trivial solution: $f^u_{ie}(\boldsymbol{u}, \boldsymbol{c})$ maps u to $\boldsymbol{o}^u_{in}$ without considering c, and maps c to $\boldsymbol{o}^u_{ex}$ without considering u. Although this trivial solution minimizes $\mathcal{L}_{\text{CICL}}(u, c)$ and $\mathcal{L}_{Dis}(u, c)$, $\boldsymbol{o}^u_{in}$ (resp. $\boldsymbol{o}^u_{ex}$) does not really the intrinsic (resp. extrinsic) factor, but it is just a mapping of the user information (resp. context information).

We prove that this trivial solution can be avoided by setting $f^u_{ie}(\boldsymbol{u}, \boldsymbol{c})$ as a *non-linear* function, leading $\boldsymbol{u}$ and $\boldsymbol{c}$ statistically interacted (the definition of statistical interaction can be found in Definition 5.2). Specifically, based on Definition 5.2, we can express the trivial solution as that $f^u_{ie}(\boldsymbol{u}, \boldsymbol{c})$ learns no statistical interaction between $\boldsymbol{u}$ and $\boldsymbol{c}$:

$$f^u_{ie}(\boldsymbol{u}, \boldsymbol{c}) = \iota_1 f_1(\boldsymbol{u}) + \iota_2 f_2(\boldsymbol{c}), \tag{7.6}$$

where f_1 outputs $\boldsymbol{o}_{in}^u$, f_2 outputs $\boldsymbol{o}_{ex}^u$, and ι are weight scalars.

However, if $\boldsymbol{u}$ and $\boldsymbol{c}$ are modeled as a statistical interaction in $f_{ie}^u(\boldsymbol{u}, \boldsymbol{c})$, the trivial solution can be prevented. Since $f_{ie}^u(\boldsymbol{u}, \boldsymbol{c})$ only takes $\boldsymbol{u}$ and $\boldsymbol{c}$ as inputs, we just need f_{ie}^u to be a non-additive model. That is, $f_{ie}^u(\boldsymbol{u}, \boldsymbol{c})$ should contain a third term $f_3(\boldsymbol{u}, \boldsymbol{c})$:

$$f_{ie}^u(\boldsymbol{u}, \boldsymbol{c}) = \iota_1 f_1(\boldsymbol{u}) + \iota_2 f_2(\boldsymbol{c}) + \iota_3 f_3(\boldsymbol{u}, \boldsymbol{c}), \tag{7.7}$$

where f_3 is a non-additive model and $\iota_3 \neq 0$.

Therefore, in the optimized situation, $\boldsymbol{o}_{in}^u = \iota_1 f_1(\boldsymbol{u})$ learns part of the information from users that do not interact with context information. $\boldsymbol{o}_{ex}^u = \iota_2 f_2(\boldsymbol{c}) + \iota_3 f_3(\boldsymbol{u}, \boldsymbol{c})$ learns the context information ($f_2(\boldsymbol{c})$) and the information that changes given different contexts ($f_3(\boldsymbol{u}, \boldsymbol{c})$).

7.5.3 Potential Problems of the Asymmetric vCLUB Method

The vCLUB-based mutual information minimization method proposed in [265] is an asymmetric method. Section 7.2.1 gives an introduction about how vCLUB method performs mutual information minimization. In this section we explain the possible reason that *vCLUB* is less robust and perform worse than our proposed bidirectional vCLUB method (*BiDis*).

If we directly apply vCLUB to our disentangling component, the parameter $\boldsymbol{\theta}_1^u$ of a variational distribution $q_1^u(\boldsymbol{o}_{ex}^u | \boldsymbol{o}_{in}^u; \boldsymbol{\theta}_1^u)$ will be trained to approach the vCLUB-based upper bound in Equation (7.1) (Step 1). Then, $\boldsymbol{\theta}_1^u$ is frozen, and $\boldsymbol{o}_{ex}^u, \boldsymbol{o}_{in}^u$ are trained to minimize $\mathcal{I}(\boldsymbol{o}_{in}^u; \boldsymbol{o}_{ex}^u)$ via minimizing the upper bound $\mathcal{I}_{\text{vCLUB}}(\boldsymbol{o}_{in}^u; \boldsymbol{o}_{ex}^u)$ (Step 2). However, this way of minimizing mutual information may result in an unexpected outcome: the mutual information may be minimized via making $\boldsymbol{o}_{in}^u$ contain as less information as possible. To better illustrate the possible outcome, we design q_1^u as a linear function and is well trained in Step 1 to ensure Equation (7.1) is an upper bound of $\mathcal{I}(\boldsymbol{o}_{in}^u; \boldsymbol{o}_{ex}^u)$. Figure 7.2 shows how the unexpected result may occur. In Step 2, $\boldsymbol{o}_{ex}^u, \boldsymbol{o}_{in}^u$ will be trained to minimize Equation (7.1). To achieve this goal, it ensures q_1^u cannot predict $\boldsymbol{o}_{ex}^u$ given the corresponding $\boldsymbol{o}_{in}^u$ from the joint distribution (the first term of Equation (7.1)), and at the same time ensures the output of q_1^u is similar to the other $\boldsymbol{o}_{ex}^u$'s from the marginal distribution (the second term of Equation (7.1)).

From $\boldsymbol{o}_{in}^u$ perspective (blue circles), the goal can be achieved by pushing the $\boldsymbol{o}_{in}^u$ to move from its original position (optimizing the first term of Equation (7.1)), and move towards the mean of the other $\boldsymbol{o}_{in}^u$'s (optimizing the second term of Equation (7.1)). From $\boldsymbol{o}_{ex}^u$ perspective (red circles), the goal can be achieved by pushing the $\boldsymbol{o}_{ex}^u$ away from its

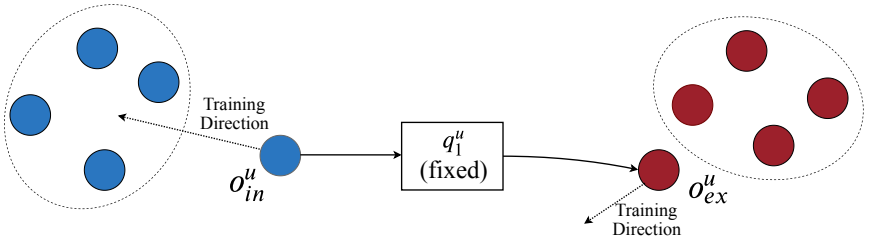


FIGURE 7.2: An illustrative example demonstrating the potential problem of the asymmetric learning in vCLUB. The blue circles are intrinsic representations and the red circles are extrinsic representations. The dotted arrows are the directions that *vCLUB* will push $\boldsymbol{o}^u_{in}$ and $\boldsymbol{o}^u_{ex}$ to move toward in their space.

original position (optimizing the first term of Equation (7.1)) and the the mean of the other $\boldsymbol{o}^u_{ex}$'s (optimizing the second term of Equation (7.1)).

This clusters all the $\boldsymbol{o}^u_{in}$'s together, making $\boldsymbol{o}^u_{in}$'s contain less information, while all the $\boldsymbol{o}^u_{ex}$'s try to split away from each other, making $\boldsymbol{o}^u_{ex}$'s contain more information. The mutual information minimization procedure is like "transfering" the information from $\boldsymbol{o}^u_{in}$'s to $\boldsymbol{o}^u_{ex}$'s, which is not what we expect. *BiDis*, however, is a symmetric disentangling method on $\boldsymbol{o}^u_{in}$'s and $\boldsymbol{o}^u_{ex}$'s so that will not result in this issue. This may be the reason that *vCLUB* performs worse and less robust than our proposed symmetrical disentangling component.

7.6 Experiments

We conduct empirical analysis to demonstrate the effectiveness of our model. In this section, we focus on 1) the recommendation performance of IEDR compared to the state-of-the-art methods; 2) the effectiveness of each component in IEDR; and 3) the ability of disentangling intrinsic and extrinsic factors of IEDR.

7.6.1 Experimental Protocol

Datasets. We evaluate our models in two scenarios: a mobile app recommendation and a restaurant recommendation. In the mobile app recommendation, we use the Frappe [247] dataset that records mobile app usage logs. Each data sample logs users' app usage in a certain context (e.g., weather, time, location). In addition, some relevant properties of the apps are also captured (e.g., category, developer). In the restaurant recommendation, we use the Yelp dataset [278]. It records users' reviews on local restaurants. Due to the fact that a user usually goes to restaurants in the same city, geographic isolation appears in the dataset. Therefore, we select the records in New York City. We regard

each record as a data sample that the user has been to the restaurant. We leverage the user/item features and context features (e.g., day of the week) to predict whether a user will go to a given restaurant in a specific context.

For each dataset, only users that have more than 5 records (data samples) are chosen. We choose the last record of each user for testing, and the second last record of each user for validation. The rest of the records are used for training. Each of these data sample is considered as a positive sample ($y = 1$). In addition, for each positive data sample in the training set, we randomly choose 2 items (but keep the user and contexts) as negative samples ($y = 0$), meaning the user did not select the 2 items in that context. For each test/validation data sample, we randomly choose 99 items as negative samples to ensure a more robust evaluation. The statistics of the datasets are shown in Table 7.1.

Datasets	**Data Samples**			**User**		**Item**		**Context**
	Train	Valid	Test	Count	Features	Count	Features	Features
Frappe	282,426	69,500	69,500	695	0	4,082	2,892	318
Yelp	518,208	633,600	633,600	6,336	24	12,902	66	13,034

TABLE 7.1: Dataset statistics. "Count" refers to the number of users/items, "Features" represent the number of different features (for "User" and "Item", the number of features excludes the user/item ids).

Baseline methods. IEDR models the feature interactions of users, items, and contexts. Therefore, we compare our model with competitive feature interaction-based recommendation methods. The methods include AFM [4], NFM [108], AutoInt [11], DeepFM [5], WDL [9], DCN [113], IFM [117], and our SIGN model proposed in Chapter 4. The above methods model all the factors in a unified representation without considering the factors that affect users' decisions. Meanwhile, we compare IEDR with the methods that learn implicit factors. They are disentangled variational auto-encoder for recommendation (DisRec) [264] and disentangled graph collaborative filtering (DGCF) [186]. We implement these methods based on their released code. Note that since DisRec and DGCF models do not consider any feature, their task is to simply predict whether a user will select an item. IERD and other baseline models, however, consider the user-item interactions in specific contexts (a user's decision to select an item may be different in different contexts). For DisRec and DGCF, to prevent the test data samples from appearing in the training set, we remove the data samples from the training set that appear in the test set (with different contexts in other models). For a fair comparison, we set the factor number to 4 for DisRec and DGCF. We use the same MLP structure for feature interaction modeling and the same embedding size for features as our IEDR model.

Implementation details. In IEDR, all the MLPs have the same hidden structure: one hidden layer of 128 dimensions and a ReLU activation after that. The input and output sizes of MLPs varies based on their needs. We set the embedding dimension to 32 for all the features. f_{ie} is an MLP that outputs a 64-dimension vector, with the first 32 dimensions are the intrinsic factor representation and the last 32 dimensions are the extrinsic factor representation. For the second (dropout-based) negative context generating method in the context-invariant contrastive learning component, the dropout rate is set to 0.5. The number of negative pairs for contrastive learning is 40 for each data sample (note that the actual negative pairs will be doubled since both $(\boldsymbol{o}_{in}^u)_{ii}$ and $(\boldsymbol{o}_{in}^u)_{ij}$ will generate 40 negative pairs). The temperature τ is set to 0.5. In the disentangling component, q_1 and q_2 are MLPs that output vectors that have the same dimension of intrinsic/extrinsic factor representations. The number of negative samples of the bidirectional vCLUB-based method is 5 for each direction. We set λ_1 to 0.1 for the Frappe dataset and 0.01 for the Yelp dataset, and set λ_2 to 0.1 for both datasets. We run all the experiments on a machine equipped with a CPU: Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz, and a GPU: Nvidia Tesla v100 GPU.

7.6.2 Overall Performance

We compare the recommendation performance of our model with various baselines. We use three common evaluation metrics for recommender systems: AUC, NDCG@k and HR@k with k being 5 and 10. The performance results are reported in Table 7.2. Each result is the average of 10 times run. For each metric, the best results are in bold, and the best baseline results are underlined. The rows *Improv* (standing for Improvements) and *p-value* show the improvement and statistical significance (through the Wilcoxon signed-rank test [279]) of IEDR over the best baseline results, respectively. From Table 7.2, we can see that our model significantly outperforms all baselines on all evaluation metrics, and the *p-value*s are all less than the 5% threshold, indicating the significance of the improvements. The improvements can come from IEDR's ability to infer the real reasons for users' decisions via learning intrinsic and extrinsic factors. The baselines considering factor disentanglement (DisRec and DGCF) perform inferior to our model. One reason is that the implicit factor disentanglement is not the best way to infer these factors. In Section 7.6.3, we empirically verify that replacing the disentanglement module in IEDR with an implicit approach (as in DisRec and DGCF) leads to a decrease in recommendation accuracy. Another reason is that DisRec and DGCF disentangling factors are purely based on user-item interactions, and do not consider feature information. This may lead to critical information loss for recommendation. Meanwhile, our model outperforms all the feature interaction-based baselines, which neglect the factor

	Frappe				
	AUC	N@5	N@10	H@5	H@10
AFM	93.18	63.52	67.44	77.84	84.71
NFM	95.86	68.30	70.73	83.00	90.40
AutoInt	95.83	69.45	71.41	84.04	90.10
DeepFM	96.09	69.20	71.28	82.70	89.50
WDL	95.96	68.02	70.33	81.70	88.90
DCN	95.48	68.78	71.20	82.59	90.07
IFM	95.32	66.91	69.13	80.90	87.60
SIGN	95.92	69.38	71.49	83.91	90.37
DisRec	85.51	56.81	60.07	67.42	76.29
DGCF	86.13	58.40	61.44	69.05	77.53
IEDR	**96.34**	**72.40**	**74.11**	**85.94**	**91.25**
Improv	0.26%	4.24%	3.66%	2.26%	0.94%
p-value	3.72%	0.25%	0.25%	0.25%	0.83%
	Yelp				
AFM	91.96	42.79	47.17	58.69	72.21
NFM	93.32	45.99	50.33	61.90	75.27
AutoInt	93.82	46.61	50.80	63.72	76.55
DeepFM	93.26	44.20	48.50	60.26	73.55
WDL	93.41	45.47	49.71	61.90	74.89
DCN	93.40	43.91	48.31	60.20	73.77
IFM	93.83	46.74	50.86	63.04	75.69
SIGN	93.67	46.80	50.94	63.68	76.41
DisRec	84.01	34.82	37.90	48.29	63.17
DGCF	85.29	36.35	39.06	50.05	64.62
IEDR	**94.22**	**48.68**	**53.05**	**65.23**	**78.29**
Improv	0.42%	4.01%	4.14%	2.38%	2.28%
p-value	2.34%	0.25%	0.25%	0.25%	0.25%

TABLE 7.2: Comparing the prediction performance (in percentage) with the baselines. The *Improv* and *p-value* rows show the relative improvements and the statistical significance of IEDR over the best performed baselines, respectively. N@k refers to NDCG@k and H@k refers to HR@k.

learning. Our model explores how context features can be leveraged to learn disentangled intrinsic and extrinsic factors for recommendation, and hence it achieves better prediction accuracy.

7.6.3 Effectiveness of Model Components

Our model has several components. This section evaluates these components in detail, and we demonstrate only the results in NDCG@10 due to the space limit. Other metrics show similar trends.

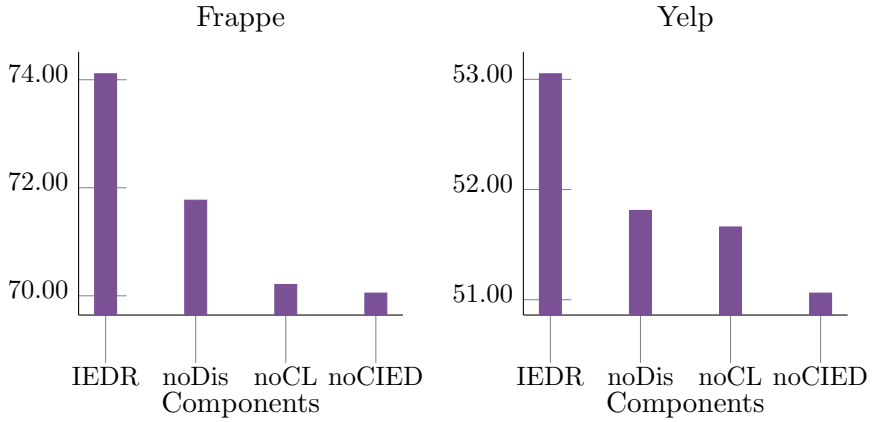


FIGURE 7.3: Ablation studies with different component(s) removed.

7.6.3.1 Ablation Study of Contrastive Intrinsic-Extrinsic Disentangling Module

The contrastive intrinsic-extrinsic disentangling (CIED) module contains a context-invariant contrastive learning component, and a disentangling component. In this section, we conduct an ablation study to show the impact of these components. We run our model in three variations: 1) without the contrastive learning component (*noCL*); 2) without the disentangling component (*noDis*); 3) without both components (*noCIED*), where the CIED module is not applied. Figure 7.3 compares our IEDR model with these variations. Comparing our model with the CIED module (*IEDR*) to *noCIED*, the significant performance gain shows the importance of learning intrinsic and extrinsic factors for accurate recommendation prediction. (*noCL*) can be regarded as performing implicit factor learning. We can see that (*noCL*) performs inferior than *IEDR*. This indicates that explicit intrinsic and extrinsic factor learning is superior to implicit factor learning methods (as in the factor disentangling baselines) in inferring the real reason for users' decisions. Employing both components achieves better performance than learning with only one. This is because either component cannot individually learn intrinsic and extrinsic factors successfully, highlighting the importance of learning intrinsic and extrinsic factors for an accurate recommendation.

7.6.4 Different Negative Context Generation Methods

We propose two negative context generating methods in the contrastive learning component: 1) sample other contexts; 2) use a large dropout rate on the original context. We evaluate the two methods in this section. Table 7.3 shows the results of our model when using only *NegGens1*, only *NegGens2*, and *NegGen1&2*. We can see that *NegGen1* results in a better performance than using *NegGen2*. This is because *NegGen1* uses the

	Frappe	Yelp
NegGen1	73.01	52.49
NegGen2	71.50	51.82
NegGen1&2	**74.11**	**53.05**

TABLE 7.3: Comparing the performance of IEDR using different negative context generating methods (for the contrastive learning component).

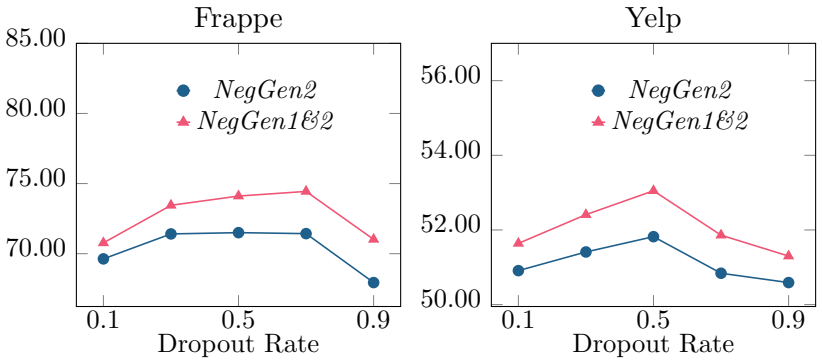


FIGURE 7.4: The performance of different dropout rates for the method 2 (*NegGen2*).

true context representations, which are consistent with what may appear in the test samples. Meanwhile, we see that *NegGen1&2* results in the best performance. This is because *NegGen2* provides more unseen (randomly generated) context representations, which strengthens the generalization ability of our model. Next, we evaluate *NegGen2* with different dropout rates. In Figure 7.4. The best performance can be achieved when the dropout rates range from 0.5 to 0.7. This is consistent with our claim in Section 7.4.2.1. The reason is that a small dropout rate (e.g., 0.1) pushes the generated context representation too close to the original one; hence it cannot be considered as a different context. However, a relatively large dropout rate (e.g., 0.9) loses too much information; hence, it is no longer a valid context representation. In addition, for *NegGen1&2* of all the dropout rates, the results consistently outperform those that only use *NegGen2*.

7.6.5 Falling Into Trivial Solutions

As discussed in Section 7.5.2, our model may fall into a trivial solution if $f_{ie}^{u}(\boldsymbol{u}, \boldsymbol{c})$ is a linear mapping method. To evaluate how the trivial solution influences our model in learning the factors, we run our model with f_{ie} being linear. Specifically, we concatenate $\boldsymbol{u}$ and $\boldsymbol{c}$ and feed them into an MLP without a hidden layer or activation (a linear mapping), making it easy to fall into the trivial solution. We call this variation *Linear*. Then, we avoid this by simply adding a nonlinear activation function (ReLU) activation after the linear mapping. We call this variation *Nonlinear*. Figure 7.5 shows the weight values of f_{ie} of the two variations. The color shows the weights

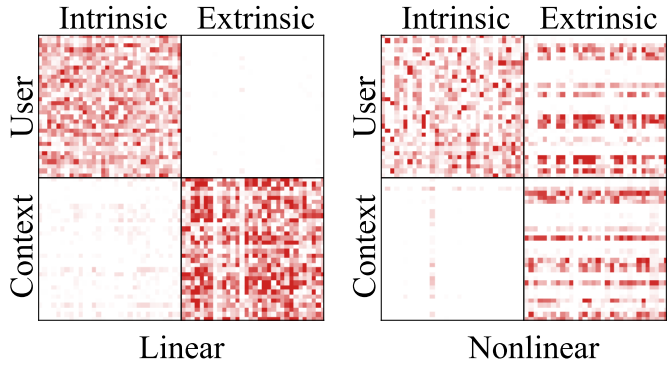


FIGURE 7.5: Visualizations of f_{ie} weights for the *Linear* and *Nonlinear* models.

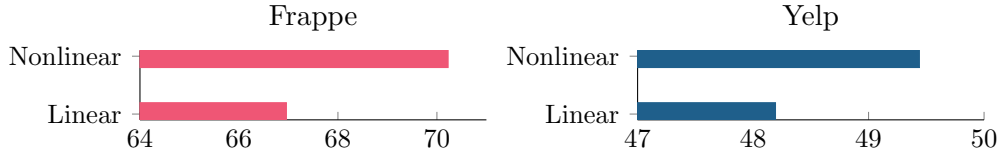


FIGURE 7.6: Comparing the performance of the *Linear* and *Nonlinear* models on different datasets.

mapping from user/context representations to intrinsic/extrinsic representations. The darker the color, the larger the weight (more information of user/context is mapped into intrinsic/extrinsic representations). The figure shows that in the *Linear* variation, user information is largely mapped into intrinsic representation (user-intrinsic block) but not extrinsic representation (user-extrinsic block). Context information is largely mapped into extrinsic representation (context-extrinsic block) but not intrinsic representation (context-intrinsic block). This means that the *Linear* variation falls into the trivial solution. On the contrary, in the *Nonlinear* variation, user information is mapped into extrinsic representation (user-extrinsic block), showing that the extrinsic representation contains both user and context information. Figure 7.6 shows the performance of the two variations. We can see that *Linear* model performs worse than *Nonlinear* model. It proves that learning intrinsic and extrinsic factors results in a better performance than simply mapping user and context information into two representations, respectively (the trivial solution).

7.6.5.1 Other Feature Modeling Methods

In the RP module, although we use a SIGN-based method to learn user, item, and context features, the module can use any feature modeling method. Here, we use other methods to evaluate whether our model still performs well. Specifically, we run our model with the other three variations using different feature modeling methods: 1) averaging feature embeddings (*MEAN*); 2) adding an MLP on top of the averaged feature

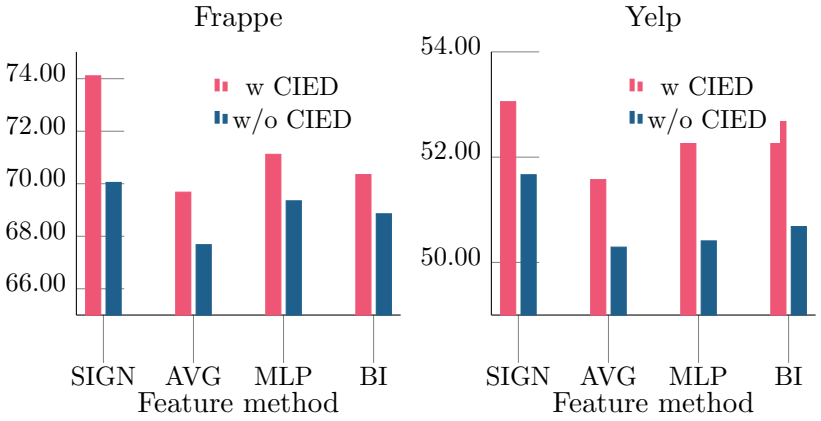


FIGURE 7.7: The model performance when equipped with different feature modeling methods.

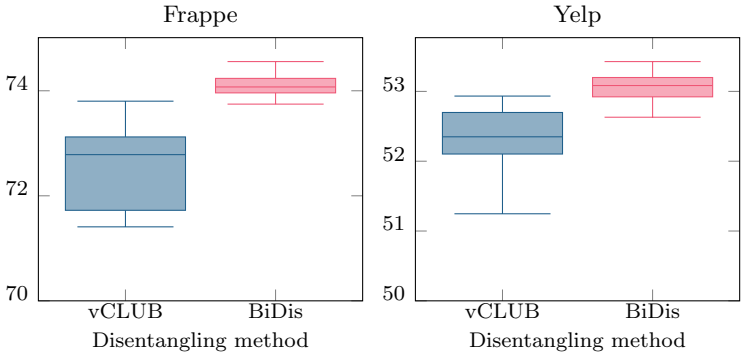


FIGURE 7.8: The performance and variance statistics of vCLUB and BiDis.

embedding (*MLP*); and 3) modeling and aggregating feature interactions through a Bi-interaction layer proposed in [108] (*BI*). The results are shown in Figure 7.7. We report the results of each variation with and without the CIED module. From the figure, we can see that when equipped with the CIED module, all feature modeling methods perform better than those without the module. It shows that our proposed CIED module can learn intrinsic and extrinsic factors for more accurate recommendation when different feature modeling methods are applied. Meanwhile, we can see that the feature modeling methods can impact the performance. *MEAN* is just a linear aggregation of features, resulting in the worst performance. Both *MLP* and *BI* have better feature modeling ability and hence have better performance than *MEAN*. The SIGN-based feature modeling (*SIGN*) is the state-of-the-art feature interaction modeling method and performs the best.

7.6.5.2 Disentangling Component Evaluation

We propose a bidirectional vCLUB-based disentangling method (*BiDis*) that disentangles the intrinsic and extrinsic factors. In this section, we compare our *BiDis* method

with the original vCLUB method (*vCLUB*). From Figure 7.8, we can see that our *BiDis* demonstrates a better performance than *vCLUB*, and its variance is much smaller than *vCLUB*. This indicates that *BiDis* generates more robust predictions, which is consistent with our analysis in Section 7.4.2.2. In addition, we visualize the intrinsic and extrinsic representations learned by the two disentanglement methods with t-SNE in Section 7.6.6.1, and observe that using *BiDis* results in a better intrinsic-extrinsic disentanglement. This is the reason why *BiDis* can deliver a better performance than *vCLUB*.

7.6.6 Disentanglement Verification

This section verifies the intrinsic and extrinsic factor disentangling ability of IEDR, including a visualization of the learned intrinsic and extrinsic representations and a case study to show the differences of these factors in users' decision making.

7.6.6.1 Visualization of Intrinsic and Extrinsic Representations

This section provides intrinsic and extrinsic representation visualizations in three variations: 1) the contrastive learning component is removed (*noCL*); 2) the disentangling component is removed (*noDis*); and 3) the asymmetric disentanglement method (*vCLUB*) is used. Figure 7.12 compares these results. We include our main observations below:

- The intrinsic and extrinsic factors are perfectly disentangled with our CIED module (*IEDR*).
- Without the disentangling component (*noDis*), the intrinsic and extrinsic disentangling procedure may not succeed. This is because there is no restriction on extrinsic representations. Therefore, the extrinsic representations can contain any information, including the information of the intrinsic factor.
- *noCL* has worse disentangling performance than *IEDR*, either. This is because the factors disentangled in *noCL* are implicit. The implicit factors only ensure the disentanglement between the factors of the same data sample, but not between the factors of other data samples. For example, some context information may be stored in the intrinsic representation in data sample 1 but be stored in the extrinsic representation in data sample 2.
- *noCIED* performs worst among all variations, which is reasonable since it does not distinguish the intrinsic and extrinsic representations.

- *vCLUB* performs disentanglement, but is not very stable in some situations. This is consistent with our analysis in Section 7.4.2.2.

7.6.6.2 Case Study

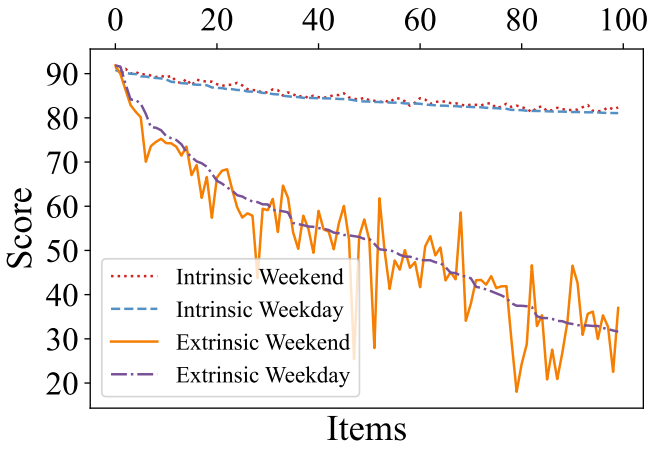


FIGURE 7.9: Comparing of a user's top 100 intrinsic and extrinsic scores in two different contexts.

Rank	User1	User2	User3
1	Phtography	Cards & Casino	Finance
2	Sports	Productivity	Health & Fitness
3	Health & Fitness	Cards & Casino	Lifestype
4	Tools	Sports Games	Communication
5	Health & Fitness	Brain & Puzzle	Communication
6	Personalization	Communication	Lifestyle
7	Personalization	Tools	Lifestyle
8	Communication	Sports	Communication
9	Personalization	Arcade & Action	Communication
10	Health & Fitness	Tools	Health & Fitness

TABLE 7.4: The items (in category) of the highest intrinsic scores for different users.

We conducted a case study to analyze the differences between the learned intrinsic and extrinsic factors. We randomly choose a user from the Frappe dataset and generate the intrinsic matching scores (the dot product of the user's intrinsic representation and the items' (apps) intrinsic representations) in two different contexts (Weekday and Weekend). The same for the extrinsic matching scores. We sort the matching scores for the intrinsic and extrinsic factors, respectively, and list the top 100 items. The results are demonstrated in Figure 7.9. Note that for the same x-axis value, intrinsic scores are for the same item, and extrinsic scores may be for another item. We can see that intrinsic scores are much more consistent in different contexts. This shows that the learned intrinsic factors are consistent in different contexts. Meanwhile, the extrinsic scores vary

	Frappe	Yelp
IEDR$_{sp}$, p=0.1	70.68	52.03
IEDR$_{sp}$, p=0.5	68.25	51.49
IEDR$_{sp}$, p=0.1, noDis	70.56	52.02
IEDR$_{sp}$, p=0.1, noCL	70.31	51.62
IEDR$_{sp}$, p=0.1, noCIED	70.16	51.10
IEDR	**74.11**	**53.05**

TABLE 7.5: Comparing the performance of IEDR$_{sp}$ with different dropout rates (for *NegGen2*).

a lot in different contexts. It means that in different contexts, the user has different extrinsic factors.

Then, we evaluate how users may have individual intrinsic factors (real interests). Table 7.4 lists the categories of the items with 10 highest intrinsic scores for three users, respectively. From the table, we can see that different users have different intrinsic interests, e.g., *User1* prefers sports and fitness apps while *User2* prefers gaming apps. Besides, it is intuitive that these apps may more likely be used in personal time instead of during working time. The intrinsic factors show users' real interests instead of just giving high scores to the often-used apps (e.g., productivity apps like Gmail and Chrome).

7.6.7 Comparing the Impact of Different Contrastive Learning Variations

To learn the intrinsic factor, we propose a context-invariant contrastive learning method. However, directly generating intrinsic factor representations through only user information seems to be a more direct way, i.e., $\boldsymbol{o}^u_{in} = f^u_{ie}(\boldsymbol{u})$. We argue that the intrinsic representations learned this way cannot guarantee the representations be intrinsic factors. This is because the information in the learned intrinsic factor representations can vary with different contexts, since these factors have never been modeled w.r.t. the contexts.

In this section, we empirically show that this approach to learn intrinsic factors is inferior to our context-invariant contrastive learning method in producing accurate recommendation. To do so, we design a variation (IEDR$_{sp}$) by splitting the intrinsic-extrinsic factor generation into two functions: $\boldsymbol{o}^u_{in} = f^u_{in}(\boldsymbol{u})$, and $\boldsymbol{o}^u_{ex} = f^u_{ex}(\boldsymbol{u}, \boldsymbol{c})$. Both f_{in} and f_{ex} have the same structure as f_{ie}, with the output dimension being a half to ensure the consistency of the factor representation dimension. The contrastive learning component does not consider context information but uses a standard InfoNCE-based contrastive

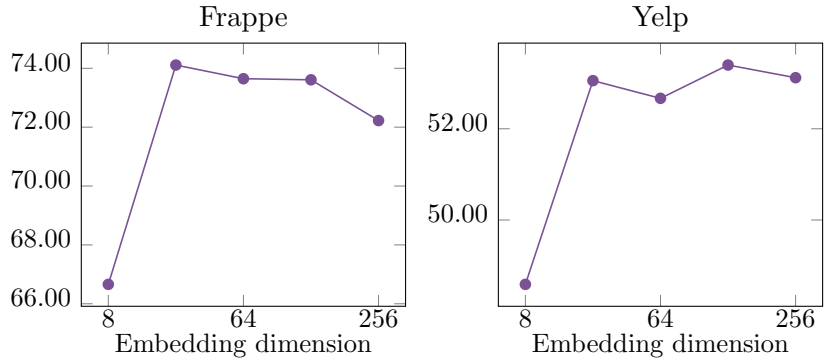


FIGURE 7.10: Hyperparameter study: different embedding dimensions d.

learning for learning robust user/item representations following [271]. Table 7.5 illustrates the results of IEDR$_{sp}$ compared to our model with IEDR$_{sp}$ using different dropout rates ($p = 0.1$ and $p = 0.5$) in the contrastive learning component, and different component combinations (*noDis*, *noCL*, *noCIED*). We can see that our model outperforms the variation in recommendation accuracy. It proves that IEDR$_{sp}$ cannot ensure a successful intrinsic factor learning and hence incur a worse recommendation accuracy. Different from IEDR, IEDR$_{sp}$ gains better performance when with the lower dropout rate. This is because, in IEDR$_{sp}$, the dropout generates views that represent the same user instead of different users. This is consistent with the conclusion in [252].

7.6.8 Effectiveness of Model Hyperparameters

We evaluate our model with different hyperparameter settings, including embedding dimensions, number of negative samples, and loss weight values. Below we summaries our observations.

7.6.8.1 Embedding Dimension

We run our model with different feature embedding dimensions. We show the results of running our model using different embeddings in Figure 7.10. Choosing the embedding dimension is a trade off between the expression ability and efficiency. From the figure, we can see that larger dimensions result in better prediction accuracy. However, the improvement is not significant when the dimension is larger than 32. A larger dimension may even reduce the performance due to the overfitting problem (e.g., dimension 256 for the Frappe dataset).

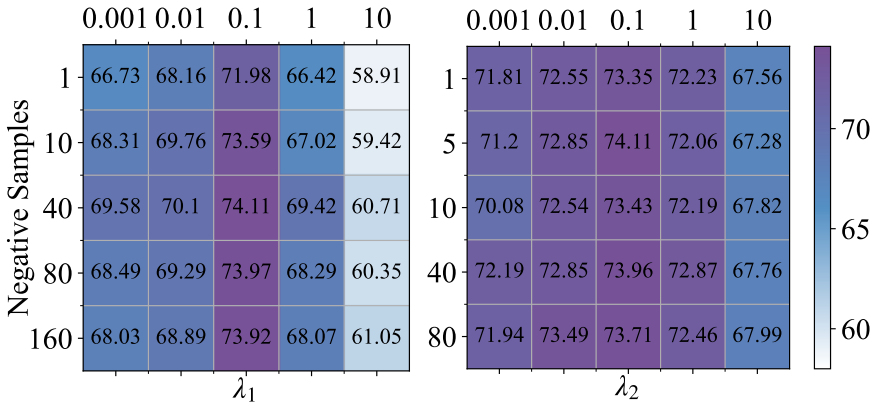


FIGURE 7.11: The performance of different numbers of negative samples and the loss weights in the risk minimization function for the contrastive learning component (left) and the disentangling component (right), respectively.

7.6.8.2 The Number of Negative sample and Loss Weight

Both the contrastive learning component and the disentangling component are contrastive-based methods that require negative sampling. In this section, we evaluate how the number of negative samples influences the performance. We also compare the influence of different loss weights of the two components. We run our model with different negative sample numbers and loss weights for the two components, respectively. From Figure 7.11, we can see that a large loss weight, or a large number of negative samples does not necessarily result in a better performance. There is the best combination of the loss weight and the number of negative samples for both components. A large or small loss weight may make the multi-task training unbalanced, harming the final performance. For the number of negative samples, a small number will make the training insufficient, while a large number may cause an overfitting problem.

7.7 Summary

In this chapter, we explore to leverage context feature interactions for more accurate recommendation predictions. Inspired by psychology, we define intrinsic and extrinsic factors that affect users' decisions in recommender systems. Following these definitions, we propose an intrinsic-extrinsic disentangled recommendation (IEDR) model to perform recommendations via learning disentangled intrinsic and extrinsic factors from the perspectives of both users and items. A context-invariant contrastive learning component and a mutual information minimization-based disentangling component are proposed to ensure the success of factor learning. Extensive experiments prove our model's ability to learn intrinsic and extrinsic factors and leverage the learned factors for accurate recommendation prediction.

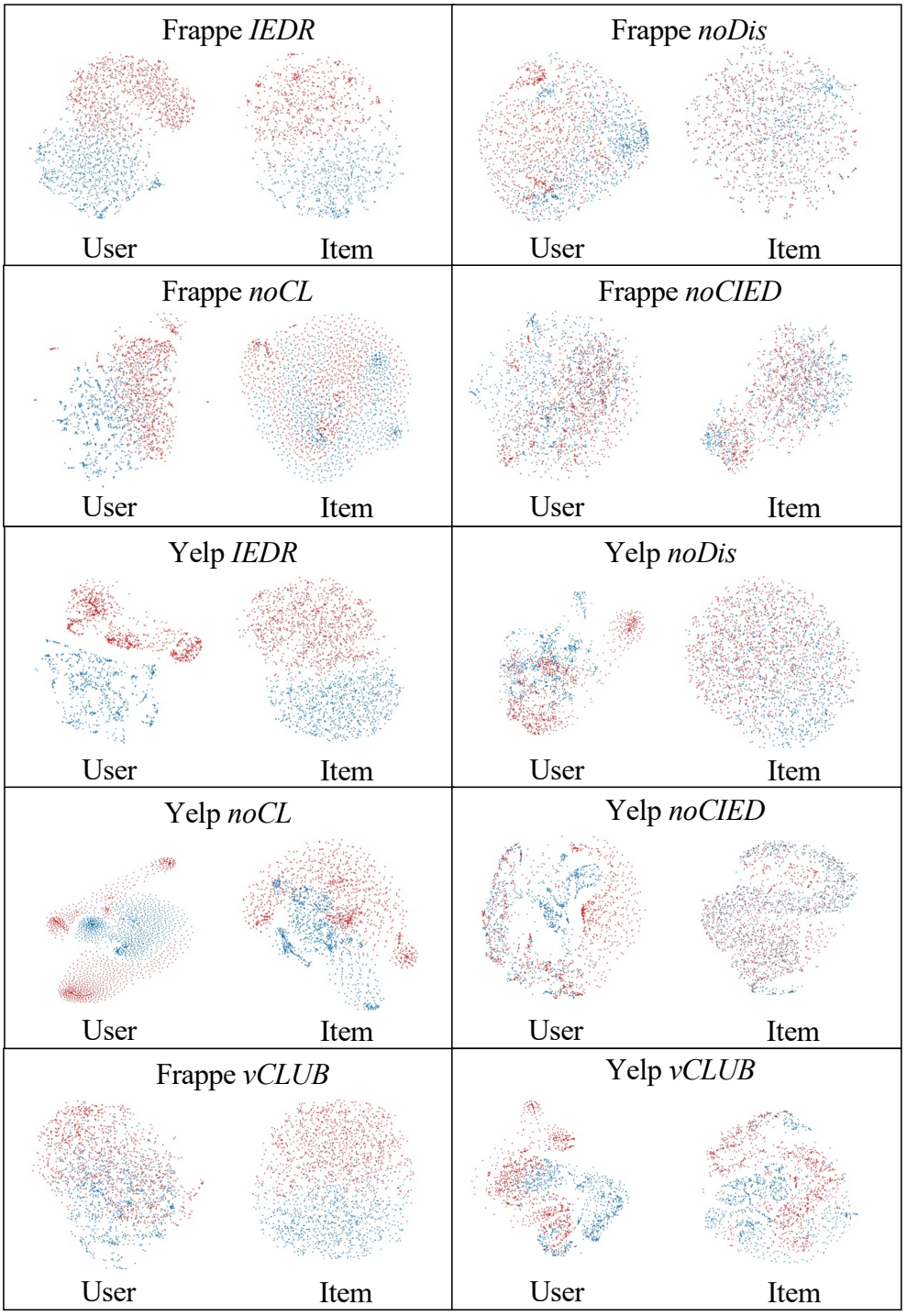


FIGURE 7.12: The complete intrinsic-extrinsic disentanglement visualizations in t-SNE. The blue dots are intrinsic representations, and the red dots are extrinsic representations.

Chapter 8

Conclusion and Future Works

Feature interaction modeling is an essential research topic in recommender systems. This thesis explored various directions to improve recommender systems' performance via explicit feature interaction modeling. Specifically, first, we extended the matrix factorization algorithm so that it can consider feature interactions for recommendation. Second, we studied to detect beneficial feature interactions and leverage the beneficial ones for more accurate and efficient recommendations. Third, we investigated novel ways to categorize and model feature interactions, so they fit their roles in the recommendation. In the following, we conclude each work in this thesis and discuss potential future research directions.

8.1 Conclusion

In Chapter 3, in order to lift the limitation that matrix factorization cannot consider feature interactions, we proposed a recommendation model called multi-matrix factorization (MMF). Our model can consider feature interactions for more accurate and interpretable recommendations under the framework of matrix factorization. Specifically, MMF learns an attribute rating matrix for each type of item attributes using the matrix factorization algorithm, which analyzes the interactions between user and the type of attributes. Then, the ratings from different attribute matrices are aggregated as the final prediction. We conducted extensive experiments to evaluate our MMF model. Experimental results show that our model significantly outperforms MF and its variations. Meanwhile, by considering feature interactions, MMF provides reasonable interpretations to the predictions and relieves the item cold-start problem.

In Chapter 4, we investigated the problem that unbeneficial feature interactions may introduce noises and decrease the prediction accuracy in recommender systems. We

proposed L_0-SIGN, a graph neural network-based approach that detects pairwise beneficial feature interactions and explicitly models these beneficial feature interactions for recommendation. Specifically, we designed a novel graph representation that considers features as nodes and pairwise beneficial feature interactions as edges. Then, the beneficial interaction detection is achieved via edge prediction, and the recommendation prediction is achieved via graph classification through our proposed statistical interaction graph neural network (SIGN). Meanwhile, we proposed an L_0 activation regularization approach to ensure the most beneficial feature interactions are detected. This is proved theoretically by showing that the empirical risk minimization procedure of L_0-SIGN is a variational approximation of the information bottleneck principle. Experimental results prove the effectiveness of our model in detecting pairwise beneficial feature interactions, and leveraging beneficial feature interactions for accurate predictions via our SIGN model.

In Chapter 5, we continued studying how to detect beneficial feature interactions and explored extending the interaction detection to arbitrary orders. In this work, we represented each arbitrary order beneficial feature interaction as a hyperedge in a hypergraph. Then, we proposed an interaction-based hypergraph neural network that models these beneficial interactions in a hypergraph for recommendation. To efficiently detect arbitrary order beneficial feature interactions, we explored generating a set of beneficial feature interactions directly instead of examining every possible one. Meanwhile, three properties of beneficial feature interactions (sufficiency, low redundancy, and disentanglement) were defined as the guidance for the generation. Mutual information maximization-based approaches (s-Infomax and Infomin) were proposed to ensure the detected feature interactions fit the three properties. Experimental results show that our model significantly outperforms baseline algorithms in recommendation accuracy. Further studies prove the effectiveness of our proposed s-Infomax and Infomin approaches in helping generate arbitrary order beneficial feature interactions.

In Chapter 6, we investigated modeling user/item attribute interactions that fit their roles in recommendations. We defined two types of attribute interactions: inner interactions for profile learning and cross interactions for preference matching. Then, we proposed a neural graph matching-based framework to model the two types of interactions for more accurate predictions. Specifically, the inner interactions were leveraged to construct user graphs and item graphs. The profile learning procedure is conducted via learning these graphs through graph neural networks. The cross interactions were leveraged to match a user graph and an item graph as the preference matching procedure. The matching result is the final prediction. Experimental results show that modeling the two types of interactions differently significantly outperforms the baselines

that model them equally. Further experiments prove that not only the modeling structure (i.e., the graph matching structure), but suitable modeling methods for different types of interactions are essential to fulfill their roles in recommendation.

In Chapter 7, we investigated leveraging another type of feature interaction, context feature interactions, to improve the performance of recommender systems. Inspired by psychology, we explored leveraging context feature interactions to infer two types of most important factors that influence users' decision-making: intrinsic factor and extrinsic factor, and model the two types of factors to jointly generate the predictions. Specifically, we proposed the intrinsic-extrinsic disentangled recommendation model, which contains a context-invariant contrastive learning component to learn intrinsic factors, and a mutual information minimization-based disentangling component to distinguish intrinsic factor and extrinsic factor. The two components together ensure the effectiveness of the factor learning. Extensive experiments were conducted to evaluate our proposed model. Experimental results show that our proposed two components are the key to the success of the factor learning. Meanwhile, our model performs predictions based on the learned factors and outperforms state-of-the-art baselines in terms of prediction accuracy.

8.2 Future Works

In the following, we outline several potential future research directions for each research question we studied in the thesis.

For the first research question of enabling matrix factorization to model feature interactions, the potential future research directions include:

- Our proposed MMF model aggregates multiple matrix factorization results (user-attribute interactions) on different types of item attributes. However, this procedure does not conduct nonlinear modeling (e.g., DNNs), limiting its potential to learn more sophisticated relations inside feature interactions. One potential future research direction is to leverage DNN-based matrix factorization methods such as neural collaborative filtering [85] to conduct powerful feature interaction modeling. Also, we can explore applying GNN-based models, such as SIGN, in MMF.
- MMF only models the interactions between users and item attributes. However, there are other feature interactions, such as those containing user attributes. Considering more abundant types of feature interactions can further improve the prediction accuracy. It is promising to extend the model structure from performing user-attribute matrix factorization to attribute-attribute matrix factorization. In this way, the interactions between user attributes and item attributes can be considered.

For the second research question of detecting beneficial feature interactions for recommendation, the potential future research directions include:

- In chapter 4, L_0-SIGN leverages an L_0 activation regularization to ensure the effectiveness of the beneficial feature interaction detection. However, to ensure the success of the detection, we need to choose an appropriate weight value for the L_0 activation regularization. A too-small value may result in an insufficient unbeneficial interaction discarding. In contrast, a too-large value may cause too many feature interactions to be discarded and lose the training signal. Therefore, investigating how to dynamically balance the weight value is an interesting research direction to further improve the robustness of L_0-SIGN.

- Our proposed statistical interaction graph neural network (SIGN) aggregates the feature interaction modeling results linearly to keep SIGN satisfying the definition of statistical interaction. However, recent researches related to GNNs proposed more sophisticated and more powerful aggregation approaches [12, 13]. It is a promising research direction to explore more sophisticated aggregation methods for SIGN while keeping the statistical interaction property.

- A feature graph is a flexible structure to represent feature interactions. However, it is still an open question how to efficiently store the graphs for model training. For L_0-SIGN (the same for our other GNN-based models), a large number of graphs (e.g., millions) need to be stored for training. Therefore, it is a challenging but promising research direction to seek a more efficient way to store graphs for training.

- In chapter 5, the HIRS model requires predefining a fixed number of feature interactions to be detected. However, different data samples may have different numbers of beneficial feature interactions. Therefore, one potential future research direction is to automatically determine an optimal number of beneficial feature interactions for each data sample.

- Currently, the interaction detection in HIRS is a procedure that cannot be intervened manually (e.g., determine which interactions can/cannot be detected). It is an interesting and promising potential research direction to enable the manual intervention for the feature interaction detection, which can prevent some ethical problems (e.g., to decrease the probability of juvenile violent crime, we can manually prevent the interaction between the user age *teenager* and the movie genre *violence* from being detected).

- Our proposed information theory-driven feature interaction detection methods (the information bottleneck guarantee in L_0-SIGN and the three properties in HIRS) are

frameworks that can be applied to more interesting research questions. For example, in multimodal learning, we can incorporate any available information without manually considering whether the information is relevant to the targets. Then, we can use our proposed methods to detect and retain the most useful information automatically.

For the third research question of categorizing and modeling feature interactions that fit their roles in recommender systems, the potential future research directions include:

- Both our proposed GMCF model in Chapter 6 and IEDR model in Chapter 7 consider pairwise feature interactions. We have proven in Chapter 5 that considering higher-order feature interactions can further improve the prediction accuracy. Therefore, one possible future research direction is to extend the two models to consider higher-order feature interactions. While considering higher-order feature interactions, corresponding changes may be required, such as a more efficient node-matching strategy in GMCF.
- In chapter 7, IEDR learns intrinsic and extrinsic factors and leverages the factors to jointly perform prediction. This is based on the assumption that the two types of factors mainly drive users' decisions. However, there may be other factor classification methods. Therefore, one potential future research direction is to incorporate other factor classification methods in IEDR to further improve the prediction accuracy.
- IEDR uses a latent vector to represent intrinsic (and extrinsic) factor. However, users may have multiple intrinsic (extrinsic) factors, and some may mainly drive a decision. For example, a user who decides to watch a comedy movie may largely be influenced by an intrinsic factor *for relaxation*. At the same time, the user may also want to watch an action movie that is influenced by another intrinsic factor *for excitement*. Therefore, future researches can perform more fine-grained factor learning to achieve more accurate predictions.
- Our proposed GMCF and IEDR model a user (or an item) as a graph, leading to a promising research direction to learn an entity holistically using the powerful GNNs. Moreover, it enlightens a way to construct complex entities through graphs in a hierarchical way (each entity may be constructed by a set of smaller entities). This is consistent with how the world is constructed (e.g., a set of atoms forms a molecule, a set of molecules forms a bigger entity, and so on).

Finally, our works for the last two research questions (L_0-SIGN, HIRS, GMCF, and IEDR) are all under the framework of GNN-based explicit feature interaction modeling. Each work improves the framework from different aspects. As a result, these works may

have the potential to be unified into a large recommender system.[1] For example, we can leverage the beneficial feature interaction generation method (with s-Infomax & Infomin) proposed in HIRS to detect arbitrary order beneficial feature interactions. Then, we leverage the detected feature interactions to construct user, item, and context graphs based on the inner interactions defined in GMCF. Next, we can leverage the IHGNN model proposed in HIRS to learn these graphs to generate user, item, and context representations. At the same time, node matching can be conducted through cross interactions defined in GMCF. The representations can be leveraged to generate intrinsic and extrinsic factors (with the contrastive learning component and the disentangling component) proposed in IEDR, and the final prediction can be generated through these factors. If these works are properly unified, the large recommender system may have the potential to combine the advantages of all these works.

[1] Note that HIRS contains all the key techniques proposed in L_0-SIGN.

Appendix A

Proof of Theorem 4.3

Theorem 4.3. *(**Statistical Interaction in SIGN**) Consider a graph $G\langle\mathcal{V},\mathcal{E}\rangle$, where $\mathcal{V}$ is the node set and $\mathcal{E} = \{e_{ij}\}_{i,j\in\mathcal{V}}$ is the edge set that $e_{ij} \in \{0,1\}, e_{ij} = e_{ji}$. Let $G\langle\mathcal{V},\mathcal{E}\rangle$ be the input of SIGN function $f_S(G)$ in Equation 4.7, then the function flags pairwise statistical interaction between node i and node j if and only if they are linked by an edge in $G\langle\mathcal{V},\mathcal{E}\rangle$, i.e., $e_{ij} = 1$.*

Proof. We prove Theorem 4.3 by proving two lemmas:

Lemma A.1. *Under the condition of Theorem 4.3, for a graph $G\langle\mathcal{V},\mathcal{E}\rangle$, if $f_S(G)$ shows pairwise statistical interaction between node i and node j, where $i,j \in \mathcal{V}$, then the two nodes are linked by an edge in $G\langle\mathcal{V},\mathcal{E}\rangle$, i.e., $e_{ij} = 1$.*

Proof. We prove this lemma by contradiction. Assume that the SIGN function f_S with $G\langle\mathcal{V},\mathcal{E}_{\setminus e_{ij}}\rangle$ as input shows pairwise statistical interaction between node i and node j, where $G\langle\mathcal{V},\mathcal{E}_{\setminus e_{ij}}\rangle$ is a graph with $\mathcal{E}_{\setminus e_{ij}}$ being a set of edges that $e_{ij} = 0$.

Recall that the SIGN function in Equation 4.7. Without losing generality, we set both the aggregation functions ϕ and ψ being element-wise average. That is:

$$f_S(G) = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}}(\frac{1}{\rho(i)}\sum_{j\in\mathcal{V}}(e_{ij}h(\boldsymbol{u}_i,\boldsymbol{u}_j))), \tag{A.1}$$

where $\rho(i)$ is the degree of node i.

From Equation A.1, we know that the SIGN function can be regarded as the linear aggregation of non-additive statistical interaction modeling procedures $h(\boldsymbol{u}_k,\boldsymbol{u}_m)$ for all nodes pairs $(k,\ m)$ that $k,m \in \mathcal{V}$ and $e_{km} = 1$. Since $\mathcal{E}_{\setminus e_{ij}}$ does not contain an edge between i and j (i.e., $e_{ij} = 0$), the SIGN function does not perform interaction modeling between the two nodes into final predictions.

According to Definition 4.2, since i and j have statistical interaction, we cannot find a replacement form of SIGN function like:

$$\begin{aligned} f_S(G) =& \hat{h}_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_{|\mathcal{V}|}) \\ &+ \hat{h}_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_{|\mathcal{V}|}), \end{aligned} \tag{A.2}$$

where $\hat{h}_{\setminus i}$ and $\hat{h}_{\setminus j}$ are functions without node i and node j as input, respectively.

However, from our assumption, since there is no edge between node i and node j, there is no interaction modeling function that performs between them in $f_S(G)$. Therefore, we can easily find many such $\hat{h}_{\setminus i}$ and $\hat{h}_{\setminus j}$ that satisfy Equation A.2. For example:

$$\begin{aligned} &\hat{h}_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_{|\mathcal{V}|}) = \\ &\frac{1}{|\mathcal{V}|} \sum_{k \in \mathcal{V} \setminus \{i\}} (\frac{1}{\rho(k)} \sum_{m \in \mathcal{V} \setminus \{i\}} (e_{km} h(\boldsymbol{u}_k, \boldsymbol{u}_m))), \end{aligned} \tag{A.3}$$

and

$$\begin{aligned} &\hat{h}_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_{|\mathcal{V}|}) = \\ &\quad \frac{1}{|\mathcal{V}|} \sum_{m \in \mathcal{V} \setminus \{j\}} (\frac{1}{\rho(i)} e_{im} h(\boldsymbol{u}_i, \boldsymbol{u}_m)) \\ &\quad + \frac{1}{|\mathcal{V}|} \sum_{k \in \mathcal{V} \setminus \{j\}} (\frac{1}{\rho(k)} e_{ki} h(\boldsymbol{u}_k, \boldsymbol{u}_i)). \end{aligned} \tag{A.4}$$

Therefore, it contradicts our assumption. Lemma A.1 is proved. □

Lemma A.2. *Under the condition of Theorem 4.3, for a graph $G\langle \mathcal{V}, \mathcal{E} \rangle$, if an edge links node i and node j in G (i.e., $i, j \in \mathcal{V}$ and $e_{ij} = 1$), then $f_S(G)$ shows pairwise statistical interaction between node i and node j.*

Proof. We prove this lemma by contradiction as well. Assume there is a graph $G\langle \mathcal{V}, \mathcal{E} \rangle$ with a pair of nodes (i, j) that $e_{ij} = 1$, but shows no pairwise statistical interaction between this node pair in $f_S(G)$.

Since $e_{ij} = 1$, we can rewrite SIGN function as:

$$\begin{aligned} f_S(G) =& \frac{1}{|\mathcal{V}|} \sum_{k \in \mathcal{V}} (\frac{1}{\rho(k)} \sum_{m \in \mathcal{V}} (e_{km} h(\boldsymbol{u}_k, \boldsymbol{u}_m))) \\ &+ \frac{\rho(i) + \rho(j)}{|\mathcal{V}| \rho(i) \rho(j)} (h(\boldsymbol{u}_i, \boldsymbol{u}_j)), \end{aligned} \tag{A.5}$$

where $(k, m) \notin \{(i, j), (j, i)\}$.

In our assumption, $f_S(G)$ shows no pairwise statistical interaction between node i and node j. That is, we can write $f_S(G)$ in the form of Equation A.2 according to Definition 4.2. For the first component in the RHS of Equation A.5, we can easily construct functions $\hat{h}_{\setminus i}$ and $\hat{h}_{\setminus j}$ in a similar way of Equation A.3 and Equation A.4 respectively. However, for the second component in the RHS of Equation A.5, the non-additive function $h(\boldsymbol{u}_i, \boldsymbol{u}_j)$ operates on node i and node j. Through the definition of non-additive function, we cannot represent a non-additive function h as a form like $h(\boldsymbol{u}_i, \boldsymbol{u}_j) = f_1(\boldsymbol{u}_i) + f_2(\boldsymbol{u}_j)$, where f_1 and f_2 are functions. That is to say, we cannot merge the second component in the RHS into either $\hat{h}_{\setminus i}$ or $\hat{h}_{\setminus j}$.

Therefore, Equation A.5 cannot be represented as the form of Equation A.2, and the node pair (i, j) shows pairwise statistical interaction in $f_S(G)$, which contradicts our assumption. Lemma A.2 is proved. □

Combing Lemma A.1 and Lemma A.2, Theorem 4.3 is proved. □

Appendix B

Proof of Corollary 4.4

Corollary 4.4. *(**Statistical Interaction in** L_0**-SIGN**) Consider a graph G that the edge set is unknown. Let G be the input of L_0-SIGN function $F_{LS}(G)$ in Equation 4.8, the function flags pairwise statistical interaction between node i and node j if and only if they are predicted to be linked by an edge in G by L_0-SIGN, i.e., $e'_{ij} = 1$.*

Proof. In Equation 4.8, we can perform the prediction procedure by first predicting edge values on all potential node pairs. Then we perform node pair modeling and aggregating the results to get the predictions (as illustrated in Figure 4.1). Specifically, we can regard the edge prediction procedure in L_0-SIGN as being prior to the following SIGN procedure. The edge values in an input graph $G\langle\mathcal{V}, \emptyset\rangle$ can be first predicted by function F_{ep}. Then, we have the graph $G\langle\mathcal{V}, \mathcal{E}'\rangle$, where E' is a predicted edge set. Therefore, the following procedure is the same as the SIGN model with $G\langle\mathcal{V}, \mathcal{E}'\rangle$ as the input graph, which satisfies Theorem 4.3. □

Appendix C

Proof of Proposition 5.1

Proposition 5.1. *Consider an input-output pair $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is a set of features, a function maps $\boldsymbol{x}$ to y in a Markov form $\boldsymbol{x} \rightarrow \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\} \rightarrow y$, where $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\}$ is a set of middle states mapped from $\boldsymbol{x}$, if the $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k\}$ makes Equation (5.13) reach the optimal value, it also makes Equation (5.12) reach the optimal value.*

Proof. Due to the non-negativity of mutual information, $\mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j) \geqslant 0$. $\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j)$ reaches the smallest value when $\mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j) = 0$ for all $i \neq j$. That is, each pair of hyperedge representations are independent. Therefore, we have

$$\mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y) = \sum_{i=1}^{k} \mathcal{I}(\boldsymbol{h}_i; y). \tag{C.1}$$

It means that in the situation that $\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j)$ reaches the smallest value, maximizing $\sum_{i=1}^{k} \mathcal{I}(\boldsymbol{h}_i; y)$ equals to maximizing $\mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y)$. Next, since

$$0 \leqslant \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y) \leqslant \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j), \tag{C.2}$$

$\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y) = 0$ when $\sum_{i \neq j} \mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j) = 0$.

Finally, since $\mathcal{I}(\boldsymbol{h}_i; y) = H(\boldsymbol{h}_i) - H(\boldsymbol{h}_i|y)$, maximizing $\mathcal{I}(\boldsymbol{h}_i; y)$ tries to reduce $H(\boldsymbol{h}_i|y)$ and in the optimal situation, $H(\boldsymbol{h}_i|y) = 0$.

In summary, when the formulation in Equation (5.13) reaches the optimal situation, we have maximized $\mathcal{I}((\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_k); y)$, and have $\mathcal{I}(\boldsymbol{h}_i; \boldsymbol{h}_j; y) = 0$ and $H(\boldsymbol{h}_i|y) = 0$, which make Equation (5.12) reach the optimal situation. □

Appendix D

Proof of Proposition 5.3

Proposition 5.3. *(**Statistical Interaction in IHGNN**) Consider an input-output pair $(\boldsymbol{x}, y)$. Let $G\langle\mathcal{V}, \emptyset\rangle$ be the input of HIRS, where $\mathcal{V} = \boldsymbol{x}$, the IHGNN function (Equation (5.8)) flags a statistical interaction among all the features in $q \subseteq \mathcal{V}$ if $q = \{a_i | \boldsymbol{E}'_{ik} = 1\}_{i \in \mathcal{V}}$, where $\boldsymbol{E}'_k$ is a hyperedge predicted by f_{gen} (Equation (5.6)).*

Proof. We prove by contradiction: assume there is a $q = \{a_i | \boldsymbol{E}'_{ik} = 1\}_{i \in \mathcal{V}}$ but the IHGNN function does not flag a statistical interaction on all the features in q.

In HIRS, the edge modeling function f_E is a neural network, which is a non-linear function. Without losing generality, we set both the aggregation function ϕ and ψ as the element-wise mean function and g is a linear regression function. Therefore, we can regard f_{IGHNN} as:

$$f_{IHGNN}(G_H\langle\mathcal{V}, \mathcal{E}'\rangle) = \sum_{j \in \mathcal{E}'} \alpha_j f_E(sum(\{\boldsymbol{V}_i^g | \boldsymbol{E}'_{ij} = 1\}_{i \in \mathcal{V}})), \tag{D.1}$$

where α_j are some scalar values.

According to Definition 5.2, if f_{IHGNN} does not flag a statistical interaction on all the features in q, we can rewrite Equation (D.1) as a linear combination of a set of functions that each function does not rely on at least one of the features in q. That is:

$$f_{IHGNN}(G_H\langle\mathcal{V}, \mathcal{E}'\rangle) = \sum_{i \in q} f_{\setminus i}(\{\boldsymbol{V}_j^g\}_{j \in \mathcal{V} \setminus \{i\}}), \tag{D.2}$$

However, since all features in q form a hyperedge in $\mathcal{E}'$, Equation (D.1) contains a component $f_E(sum(\{\boldsymbol{V}_i^g | i \in q\}_{i \in \mathcal{V}}))$, which non-linearly models all features in q. This component does not belong to any part of the right hand side of Equation (D.2). Therefore, Equation (D.2) does not hold, which is a contradiction of the assumption. □

Appendix E

Proof of Theorem 7.2

Theorem 7.2. *(**Equivalence of contrastive loss** $\mathcal{L}^u_{CICL}$) Optimizing the contrastive loss is equivalent to:*

$$\arg\min \sum_{i=1}^{N} \mathcal{L}^u_{CICL}(u_i, c_i) = \arg\max \Big(\mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) - \mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{c}) \Big) .$$

Proof. Since the mutual information is not explicitly intractable, we approximate the right side of Equation (7.5) with a lower bound (i.e., MINE [280]) and an upper bound (i.e., CLUB [265]) of mutual information, respectively. More formally,

$$\mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) \geq \mathcal{I}_{MINE}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) := \mathbb{E}_{p(\boldsymbol{o}^u_{in}, \boldsymbol{u})} \left[\log p(\boldsymbol{o}^u_{in}, \boldsymbol{u}) \right] - \log \mathbb{E}_{p(\boldsymbol{o}^u_{in}) p(\boldsymbol{u})} \left[p(\boldsymbol{o}^u_{in}, \boldsymbol{u}) \right] ; \quad \text{(E.1)}$$

$$\mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{c}) \leq \mathcal{I}_{CLUB}(\boldsymbol{o}^u_{in}, \boldsymbol{c}) := \mathbb{E}_{p(\boldsymbol{o}^u_{in}, \boldsymbol{c})} \left[\log p(\boldsymbol{o}^u_{in} | \boldsymbol{c}) \right] - \mathbb{E}_{p(\boldsymbol{o}^u_{in}) p(\boldsymbol{c})} \left[\log p(\boldsymbol{o}^u_{in} | \boldsymbol{c}) \right] . \quad \text{(E.2)}$$

With the approximated terms above, proving Equation. (7.5) turns to verify:

$$\arg\min \sum_{i=1}^{N} \mathcal{L}_{\text{CICL}}(u_i, c_i) = \arg\max \Big(\mathcal{I}_{MINE}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) - \mathcal{I}_{CLUB}(\boldsymbol{o}^u_{in}, \boldsymbol{c}) \Big) . \quad \text{(E.3)}$$

By minimizing $\mathcal{L}_{\text{CICL}}$, we aim to make $(\boldsymbol{o}^u_{in})_{ii}$ similar to $(\boldsymbol{o}^u_{in})_{ij}$. This procedure can be interpreted in probability as: increasing the probability of $f^u_{ie}(\boldsymbol{u}_i, \boldsymbol{c}_j)$ to predict $(\boldsymbol{o}^u_{in})_{ii}$. Therefore, maximizing the $\exp(\text{sim}((\boldsymbol{o}^u_{in})_{ii}, (\boldsymbol{o}^u_{in})_{ij})/\tau)$ in Equation (7.2) is equivalent to maximizing $p((\boldsymbol{o}^u_{in})_{ii} | \boldsymbol{u}_i, \boldsymbol{c}_j)$ ($\exp(\cdot)$ is monotone increasing so that does not influence the conclusion). Similarly, minimizing $\exp(\text{sim}((\boldsymbol{o}^u_{in})_{ii}, (\boldsymbol{o}^u_{in})_{\ell i})/\tau)$ is equivalent to

minimizing $p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{\ell}, \boldsymbol{c}_{i})$. Therefore, we have

$$
\begin{aligned}
&-\sum_{i=1}^{N}\mathcal{L}_{\mathrm{CICL}}(u_i, c_i) \\
=&\sum_{i=1}^{N}\log\frac{\exp(\mathrm{sim}((\boldsymbol{o}_{in}^{u})_{ii}, (\boldsymbol{o}_{in}^{u})_{ij})/\tau)}{\sum_{u_{\ell}\in\mathcal{U}}\exp(\mathrm{sim}((\boldsymbol{o}_{in}^{u})_{ii}, (\boldsymbol{o}_{in}^{u})_{\ell i})/\tau)} \\
=&\sum_{i=1}^{N}\log[\exp(\mathrm{sim}((\boldsymbol{o}_{in}^{u})_{ii}, (\boldsymbol{o}_{in}^{u})_{ij})/\tau)] - \sum_{i=1}^{N}\log[\sum_{u_{\ell}\in\mathcal{U}}\exp(\mathrm{sim}((\boldsymbol{o}_{in}^{u})_{ii}, (\boldsymbol{o}_{in}^{u})_{\ell i})/\tau)] \\
=&\sum_{i=1}^{N}\log[p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{i}, \boldsymbol{c}_{j})] - \sum_{i=1}^{N}\log[\sum_{u_{\ell}\in\mathcal{U}}p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{\ell}, \boldsymbol{c}_{i})].
\end{aligned}
$$

Equation (7.2) only samples one context c_j for each data point. However, during the training, all contexts in $\mathcal{C}$ are expected to be sampled. If we count all contexts, we have

$$
\begin{aligned}
&\sum_{i=1}^{N}\log[p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{i}, \boldsymbol{c}_{j})] - \sum_{i=1}^{N}\log[\sum_{u_{\ell}\in\mathcal{U}}p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{\ell}, \boldsymbol{c}_{i})] \\
=&\sum_{i=1}^{N}\sum_{c_j\in\mathcal{C}}\log[p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{i}, \boldsymbol{c}_{j})] - \sum_{i=1}^{N}\log[\sum_{u_{\ell}\in\mathcal{U}}p((\boldsymbol{o}_{in}^{u})_{ii}|\boldsymbol{u}_{\ell}, \boldsymbol{c}_{i})] \\
=&\mathbb{E}_{p(\boldsymbol{o}_{in}^{u}, \boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}_{in}^{u}|\boldsymbol{u}, \boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}_{in}^{u}, \boldsymbol{c})}\log\mathbb{E}_{p(\boldsymbol{u})}[p(\boldsymbol{o}_{in}^{u}|\boldsymbol{u}, \boldsymbol{c})]
\end{aligned}
\tag{E.4}
$$

Equation (E.4) is the probability form of the objective function of the context-invariant counteractive learning component (Equation (7.2)). Equation (E.4) maximizes the likelihood $p(\boldsymbol{o}_{in}^{u}|\boldsymbol{u}, \boldsymbol{c})$ given the joint distribution of users and intrinsic factors, with the marginal distribution of contexts. Meanwhile, it minimizes the likelihood $p(\boldsymbol{o}_{in}^{u}|\boldsymbol{u}, \boldsymbol{c})$ given the joint distribution of contexts and intrinsic factors, with the marginal distribution of user.[1]

From Equation (E.4), we further have:

[1]Note that only if $f_{ie}^{u}(\boldsymbol{u}, \boldsymbol{c})$ is a many-to-one (or one-to-one) mapping then Equation (E.4) and Equation (7.2) will be equivalent. Otherwise, given a sample pair $(\boldsymbol{u}, \boldsymbol{c})$, $f_{ie}^{u}(\boldsymbol{u}, \boldsymbol{c})$ may have different $\boldsymbol{o}_{in}^{u}$ outputs (i.e., one-to-many). In this situation, the first term of Equation (E.4) cannot guarantee that the same user with different context will have the same intrinsic factor (i.e., they may have various intrinsic factor representations while still meet the objective of the first term of Equation (E.4)). We use an MLP as $f_{ie}^{u}(\boldsymbol{u}, \boldsymbol{c})$, which is a many-to-one mapping function. Therefore, we can ensure the equivalence between Equation (E.4) and Equation (7.2).

$$
\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})} \log \mathbb{E}_{p(\boldsymbol{u})}[p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] \\
&\overset{(a)}{=} \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] + \big(\mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})] - \mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})]\big) \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})p(\boldsymbol{u})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})] \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})] \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})] \\
&\quad + \big(\mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})]\big) \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{u})}[\log p(\boldsymbol{u})] \\
&\quad - \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] + \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] \\
&= \mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] \\
&\quad - \big(\mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})]\big) \\
&= \mathbb{E}_{p(\boldsymbol{c})}\left(\mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{u})}[\log p(\boldsymbol{o}^u_{in},\boldsymbol{u}|\boldsymbol{c})]\right) \\
&\quad - \mathbb{E}_{p(\boldsymbol{u})}\left(\mathbb{E}_{p(\boldsymbol{o}^u_{in},\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})] - \mathbb{E}_{p(\boldsymbol{o}^u_{in})p(\boldsymbol{c})}[\log p(\boldsymbol{o}^u_{in}|\boldsymbol{u},\boldsymbol{c})]\right) .
\end{aligned}
\tag{E.5}
$$

(a): In the second term, pushing the log inside the expectation dose not change the minimizer.

Equation (E.1) and the first term of Equation (E.5) both maximize the expected log-ratio of the joint distribution over the marginal distributions [154]. Therefore, maximizing this term in Equation (E.5) will have the same effect to maximizing Equation (E.1). We can interpret the first term of Equation (E.5) as maximizing the mutual information between users and the corresponding intrinsic factor, conditioned on a given context. Similarly, maximizing the negative of the second term of Equation (E.5) will have the same effect of minimizing Equation (E.2), which can be interpreted as minimizing the mutual information between contexts and the corresponding intrinsic factors, conditioned on a given users.

Therefore, we can conclude that:

$$
\arg\min \sum_{i=1}^{N} \mathcal{L}^u_{\text{CICL}}(u_i, c_i) = \arg\max \Big(\mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{u}) - \mathcal{I}(\boldsymbol{o}^u_{in}, \boldsymbol{c}) \Big) .
$$

□

Bibliography

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[2] Steffen Rendle. Factorization machines. In *Proceedings of the Industrial Conference on Advances in Data Mining (ICDM)*, pages 995–1000, 2010.

[3] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 208–211, 2008.

[4] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3119–3125, 2017.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1725–1731, 2017.

[6] Michail Vlachos, Vassilios G Vassiliadis, Reinhard Heckel, and A Labbi. Toward interpretable predictive models in b2b recommender systems. *IBM Journal of Research and Development*, 60(5/6):11–1, 2016.

[7] Michael Tsang, Hanpeng Liu, Sanjay Purushotham, Pavankumar Murali, and Yan Liu. Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5804–5813, 2018.

[8] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[9] 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. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (RecSys)*, pages 7–10, 2016.

[10] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1059–1068, 2018.

[11] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1161–1170, 2019.

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–14, 2017.

[13] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–17, 2019.

[14] Pat Langley et al. Selection of relevant features in machine learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, volume 184, pages 245–271, 1994.

[15] Norbert Siegmund, Sergiy S Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. Predicting performance via automated feature-interaction detection. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 167–177, 2012.

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1754–1763, 2018.

[17] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 2636–2645. ACM, 2020.

[18] Robert L. Peabody and Bertram M. Gross. The managing of organizations: The administrative struggle. *The ANNALS of the American Academy of Political and Social Science*, 360(1):197–198, 1965.

[19] Alvin Toffler. Future shock. *Sydney. Pan*, 1970.

[20] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the conference on Hypertext and Hypermedia (HyperText)*, pages 73–82, 2009.

[21] Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. Personalized search on the world wide web. In *The Adaptive Web*, pages 195–230. Springer, 2007.

[22] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[23] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, pages 325–341. Springer, 2007.

[24] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of The Web Conference (WWW)*, pages 1041–1042, 2008.

[25] Mahiye Uluyagmur, Zehra Cataltepe, and Esengul Tayfur. Content-based movie recommendation using different feature sets. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS)*, volume 1, pages 17–24, 2012.

[26] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6):734–749, 2005.

[27] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern Information Retrieval*, volume 463. ACM, 1999.

[28] Gerard Salton. A new comparison between conventional indexing (medlars) and automatic text processing (smart). *Journal of the American Society for Information Science*, 23(2):75–84, 1972.

[29] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[30] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the International Conference on Digital Libraries (DL)*, pages 89–98, 1998.

[31] Kurt D Bollacker, Steve Lawrence, and C Lee Giles. Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pages 116–123, 1998.

[32] Michael D Ekstrand, Praveen Kannan, James A Stemper, John T Butler, Joseph A Konstan, and John T Riedl. Automatically building research reading lists. In *Proceedings of the Conference on Recommender Systems (RecSys)*, pages 159–166, 2010.

[33] Joeran Beel, Stefan Langer, Marcel Genzmehr, and Andreas Nürnberger. Introducing docear's research paper recommender system. In *Proceedings of the Joint Conference on Digital Libraries (JCDL)*, pages 459–460, 2013.

[34] Gi Hyun Lim, Seung Woo Hong, Inhee Lee, Il Hong Suh, and Michael Beetz. Robot recommender system using affection-based episode ontology for personalization. In *Proceedings of the International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 155–160, 2013.

[35] Zameer Gulzar, A Anny Leema, and Gerard Deepak. Pcrs: Personalized course recommender system based on hybrid approach. *Procedia Computer Science*, 125: 518–524, 2018.

[36] Felice Ferrara, Nirmala Pudota, and Carlo Tasso. A keyphrase-based paper recommender system. In *Proceedings of the Italian Research Conference on Digital Library Management Systems (IRCDL)*, pages 14–25, 2011.

[37] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[38] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[39] Yolanda Blanco-Fernandez, Jose J Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, and Martin Lopez-Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *Transactions on Consumer Electronics*, 54(2):727–735, 2008.

[40] Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating tags in a semantic content-based recommender. In *Proceedings of the Conference on Recommender Systems (RecSys)*, pages 163–170. ACM, 2008.

[41] Michal Kompan and Mária Bieliková. Content-based news recommendation. In *Proceedings of the International Conference on Electronic Commerce and Web Technologies (EC-Web)*, pages 61–72. Springer, 2010.

[42] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. Terms of a feather: Content-based news recommendation and discovery using twitter. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 448–459. Springer, 2011.

[43] Md Nuruddin Monsur Adnan, Mohammed Rashid Chowdury, Iftifar Taz, Tauqir Ahmed, and Rashedur M Rahman. Content based news recommendation system based on fuzzy logic. In *Proceedings of the International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 1–6. IEEE, 2014.

[44] Pedro Cano, Markus Koppenberger, and Nicolas Wack. Content-based music audio recommendation. In *Proceedings of the International Conference on Multimedia (MM)*, pages 211–212, 2005.

[45] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, 2013.

[46] SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S Ashok, and B Venkatesh. Content-based movie recommendation system using genre correlation. In *Smart Intelligent Computing and Applications*, pages 391–397. Springer, 2019.

[47] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1):115–153, 2001.

[48] Bing Xu, Mingmin Zhang, Zhigeng Pan, and Hongwei Yang. Content-based recommendation in e-commerce. In *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA)*, pages 946–955. Springer, 2005.

[49] Chen Jian, Yin Jian, and Huang Jin. Automatic content-based recommendation in e-commerce. In *Proceedings of the International Conference on e-Technology, e-Commerce and e-Service (EEE)*, pages 748–753. IEEE, 2005.

[50] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, 2009.

[51] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Conference*

on Uncertainty in Artificial Intelligence (UAI), pages 43–52. Morgan Kaufmann, 1998.

[52] Juan Wang, Lan Yue-xin, and Wu Chun-ying. Survey of recommendation based on collaborative filtering. *Journal of Physics: Conference Series*, 1314(1), 2019.

[53] Gourav Jain, Tripti Mahara, and Kuldeep Narayan Tripathi. A survey of similarity measures for collaborative filtering-based recommender system. *Soft Computing: Theories and Applications*, pages 343–352, 2020.

[54] Alfred Kobsa. User modeling and user-adapted interaction. In *Proceedings of the Conference Companion on Human Factors in Computing Systems (CHI)*, pages 415–416, 1994.

[55] Daniel Billsus, Michael J Pazzani, et al. Learning collaborative information filters. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 98, pages 46–54, 1998.

[56] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the Conference on Electronic Commerce (ICEC)*, pages 158–167, 2000.

[57] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of The Web Conference (WWW)*, pages 285–295, 2001.

[58] Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-based Systems*, 56:156–166, 2014.

[59] S Nithya, A Srinivasan, M Senthilkumar, et al. Calculating the user-item similarity using pearson's and cosine correlation. In *Proceedings of the International Conference on Trends in Electronics and Informatics (ICEI)*, pages 1000–1004. IEEE, 2017.

[60] Yong Wang, Jiangzhou Deng, Jerry Gao, and Pu Zhang. A hybrid user similarity model for collaborative filtering. *Information Sciences*, 418:102–118, 2017.

[61] Zhenhua Tan and Liangliang He. An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle. *IEEE Access*, 5:27211–27228, 2017.

[62] Bidyut Kr Patra, Raimo Launonen, Ville Ollikainen, and Sukumar Nandi. A new similarity measure using bhattacharyya coefficient for collaborative filtering in sparse data. *Knowledge-Based Systems*, 82:163–177, 2015.

[63] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 247–254, 2001.

[64] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[65] Yung-Hsin Chien and Edward I George. A bayesian model for collaborative filtering. In *Proceedings of the Artificial Intelligence and Statistics Conference (AISTATS)*, 1999.

[66] Koji Miyahara and Michael J Pazzani. Improvement of collaborative filtering with the simple bayesian classifier. *Information Processing Society of Japan*, 43(11), 2002.

[67] Priscila Valdiviezo-Diaz, Fernando Ortega, Eduardo Cobos, and Raúl Lara-Cabrera. A collaborative filtering approach based on naïve bayes classifier. *IEEE Access*, 7:108581–108592, 2019.

[68] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, volume 1, pages 114–129. Menlo Park, CA, 1998.

[69] Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the WorkShop on International Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[70] Songjie Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *J. Softw.*, 5(7):745–752, 2010.

[71] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2): 133–151, 2001.

[72] Thomas K Landauer and Michael L Littman. Computerized cross-language document retrieval using latent semantic indexing, April 5 1994. US Patent 5,301,109.

[73] Badrul M Sarwar, George Karypis, J Konstan, and John Riedl. Incremental svd-based algorithms for highly scaleable recommender systems. In *Proceedings of the International Conference on Computer and Information Technology (ICCIT)*, pages 345–354, 2002.

[74] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 426–434, 2008.

[75] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 549–553. SIAM, 2006.

[76] Amy N Langville, Carl D Meyer, Russell Albright, James Cox, and David Duling. Initializations for the nonnegative matrix factorization. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 23–26. Citeseer, 2006.

[77] Sanjeev Arora, Rong Ge, Ravi Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization—provably. *SIAM Journal on Computing*, 45(4):1582–1611, 2016.

[78] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Processings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 1257–1264, 2008.

[79] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 880–887, 2008.

[80] Nathan Srebro, Jason DM Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 17, pages 1329–1336. Citeseer, 2004.

[81] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 713–719, 2005.

[82] Markus Weimer, Alexandros Karatzoglou, Quoc Le, and Alex Smola. Cofirank-maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 222–230, 2007.

[83] Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 249–256, 2006.

[84] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

[85] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of The Web Conference (WWW)*, pages 173–182, 2017.

[86] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.

[87] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2227–2233, 2018.

[88] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of The Web Conference (WWW)*, pages 689–698, 2018.

[89] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Doubly robust joint learning for recommendation on data missing not at random. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 6638–6647. PMLR, 2019.

[90] Tim Miranda, Mark Claypool, Anuja Gokhale, Tim Mir, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[91] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User Modeling and User-adapted Interaction*, 10(2):147–180, 2000.

[92] Barry Smyth and Paul Cotter. A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.

[93] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

[94] Nathaniel Good, J Ben Schafer, Joseph A Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, John Riedl, et al. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 315149–315352, 1999.

[95] Ian Soboroff and Charles Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 86–91, 1999.

[96] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 714–720, 1998.

[97] Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems, 2000.

[98] Michelle Keim Condliff, David D Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, volume 99, pages 23–30. Citeseer, 1999.

[99] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 253–260, 2002.

[100] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[101] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the Conference on Recommender Systems (RecSys)*, pages 191–198, 2016.

[102] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[103] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 539–548, 2019.

[104] Christopher M. Bishop. *Pattern Recognition and Machine Learning, 5th Edition.* Information Science and Statistics. Springer, 2007.

[105] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 417–431. Springer, 2020.

[106] Ming-Yen Cheng and Hau-tieng Wu. Local linear regression on manifolds and its geometric interpretation. *Journal of the American Statistical Association*, 108 (504):1421–1434, 2013.

[107] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1135–1144, 2016.

[108] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 355–364, 2017.

[109] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 3609–3616, 2020.

[110] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[111] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 5941–5948, 2019.

[112] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *KDD workshop*, pages 1–4, 2019.

[113] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1–7, 2017.

[114] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 46–54, 2018.

[115] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[116] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. Efficient non-sampling factorization machines for optimal context-aware recommendation. In *Proceedings of The Web Conference (WWW)*, pages 2400–2410, 2020.

[117] Yantao Yu, Zhen Wang, and Bo Yuan. An input-aware factorization machine for sparse prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1466–1472, 2019.

[118] Wantong Lu, Yantao Yu, Yongzhe Chang, Zhen Wang, Chenhui Li, and Bo Yuan. A dual input-aware factorization machine for ctr prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3139–3145, 2020.

[119] Tong Chen, Hongzhi Yin, Quoc Viet Hung Nguyen, Wen-Chih Peng, Xue Li, and Xiaofang Zhou. Sequence-aware factorization machines for temporal predictive analytics. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1405–1416. IEEE, 2020.

[120] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3351–3359, 2016.

[121] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 850–858. PMLR, 2016.

[122] Junlin Zhang, Tongwen Huang, and Zhiqi Zhang. Fat-deepffm: Field attentive deep field-aware factorization machine. In Petra Perner, editor, *Proceedings of the Industrial Conference on Advances in Data Mining (ICDM)*, pages 43–57. ibai Publishing, 2019.

[123] Bruce W Wilshire. *William James and Phenomenology: A study of the Principles of Psychology*. Ams Pr Inc, 1968.

[124] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112, 2014.

[125] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[126] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1480–1489, 2016.

[127] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1243–1252. PMLR, 2017.

[128] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NeurIPS)*, 28:1693–1701, 2015.

[129] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2048–2057. PMLR, 2015.

[130] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.

[131] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Transactions on Multimedia*, 19(6):1245–1256, 2017.

[132] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4438–4446, 2017.

[133] Chenkai Guo, Yifan Xu, Xiaolei Hou, Naipeng Dong, Jing Xu, and Quanqi Ye. Deep attentive factorization machine for app recommendation service. In *Proceedings of the International Conference on Web Services (ICWS)*, pages 134–138. IEEE, 2019.

[134] Zhulin Tao, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. Hoafm: a high-order attentive factorization machine for ctr prediction. *Information Processing & Management*, 57(6):102076, 2020.

[135] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[136] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, 2019.

[137] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[138] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 7354–7363. PMLR, 2019.

[139] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3146–3154, 2019.

[140] Seongjun Yun, Raehyun Kim, Miyoung Ko, and Jaewoo Kang. Sain: Self-attentive integration network for recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1205–1208, 2019.

[141] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. Deep feedback network for recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2519–2525, 2020.

[142] Zhibo Xiao, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Hao Wang. Deep multi-interest network for click-through rate prediction. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2265–2268, 2020.

[143] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 313–321, 2020.

[144] Yunxiang Zhao, Jianzhong Qi, Qingwei Liu, and Rui Zhang. Wgcn: Graph convolutional networks with weighted structural features. *arXiv preprint arXiv:2104.14060*, 2021.

[145] Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. Graph pooling via coarsened graph infomax. *arXiv preprint arXiv:2105.01275*, 2021.

[146] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2224–2232, 2015.

[147] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017.

[148] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–14, 2016.

[149] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4502–4510, 2016.

[150] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11, 2017.

[151] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11, 2018.

[152] Zekun Li, Shu Wu, Zeyu Cui, and Xiaoyu Zhang. Graphfm: Graph factorization machines for feature interaction modeling. *CoRR*, abs/2105.11866, 2021.

[153] Yuexiang Xie, Zhen Wang, Yaliang Li, Bolin Ding, Nezihe Merve Gürel, Ce Zhang, Minlie Huang, Wei Lin, and Jingren Zhou. Fives: Feature interaction via edge search for large-scale tabular data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 3795–3805, 2021.

[154] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[155] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[156] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 3558–3565, 2019.

[157] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1025–1035, 2017.

[158] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[159] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 793–803, 2019.

[160] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5410–5419, 2017.

[161] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph t-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[162] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: an efficient subgraph-based framework for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 335–351, 2018.

[163] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2018.

[164] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12026–12035, 2019.

[165] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3): 93–93, 2008.

[166] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 1506–1515, 2017.

[167] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1957–1967, 2017.

[168] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for amr-to-text generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1616–1626, 2018.

[169] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. Graph-to-sequence learning using gated graph neural networks. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 273–283, 2018.

[170] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[171] Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *Proceedings of the International Conference on Business Process Management (BPM)*, pages 48–63. Springer, 2009.

[172] Xifeng Yan, Philip S Yu, and Jiawei Han. Substructure similarity search in graph databases. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 766–777, 2005.

[173] Peter Willett, John M Barnard, and Geoffrey M Downs. Chemical similarity searching. *Chemical Information and Computer Sciences*, pages 983–996, 1998.

[174] John W Raymond, Eleanor J Gardiner, and Peter Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[175] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 321–328, 2003.

[176] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research (JMLR)*, 11:1201–1242, 2010.

[177] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 488–495, 2009.

[178] Nino Shervashidze and Karsten Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1660–1668, 2009.

[179] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 384–392, 2019.

[180] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3835–3845, 2019.

[181] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 737–744, 1994.

[182] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4938–4947, 2020.

[183] Bo Jiang, Pengfei Sun, and Bin Luo. Glmnet: Graph learning-matching convolutional networks for feature matching. *Pattern Recognition*, 121:108167, 2022.

[184] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion, 2017.

[185] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the International conference on Research and Development in Information Retrieval (SIGIR)*, pages 165–174, 2019.

[186] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. Disentangled graph collaborative filtering. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1001–1010, 2020.

[187] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 639–648, 2020.

[188] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 726–735, 2021.

[189] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of The Web Conference (WWW)*, pages 417–426, 2019.

[190] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 235–244, 2019.

[191] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *Proceedings of The Web Conference (WWW)*, pages 2091–2102, 2019.

[192] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. Knowledge-aware coupled graph neural network for social recommendation. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2021.

[193] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of The Web Conference (WWW)*, pages 413–424, 2021.

[194] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 417–426, 2018.

[195] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label

smoothness regularization for recommender systems. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 968–977, 2019.

[196] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 950–958, 2019.

[197] Zuoxi Yang and Shoubin Dong. Hagerec: hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *Knowledge-Based Systems*, 204:106194, 2020.

[198] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. Semi-supervised user profiling with heterogeneous graph attention networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 19, pages 2116–2122, 2019.

[199] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of The Web Conference (WWW)*, pages 151–161, 2019.

[200] Peijie Sun, Le Wu, and Meng Wang. Attentive recurrent social recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 185–194, 2018.

[201] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. An efficient adaptive transfer neural network for social-aware recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 225–234, 2019.

[202] Thomas Blumensath, Mehrdad Yaghoobi, and Mike E Davies. Iterative hard thresholding and l0 regularisation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, pages III–877. IEEE, 2007.

[203] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[204] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[205] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[206] Joseph F Murray and Kenneth Kreutz-Delgado. An improved focuss-based learning algorithm for solving sparse linear inverse problems. In *Conference Record of Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, volume 1, pages 347–351. IEEE, 2001.

[207] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[208] Kyle K Herrity, Anna C Gilbert, and Joel A Tropp. Sparse approximation via iterative thresholding. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*, volume 3, pages III–III. IEEE, 2006.

[209] Chengzhi Shi, Ben Glocker, and Daniel C Castro. Pvae: Learning disentangled representations with intrinsic dimension via approximated l0 regularization. In *Proceedings of Machine Learning Research (PMLR)*, pages 1–6, 2019.

[210] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[211] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, 36(4):1567–1594, 2008.

[212] Seyoung Kim and Eric P Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[213] Valeria Fonti and Eduard Belitser. Feature selection using lasso. *VU Amsterdam Research Paper in Business Analytics*, 30:1–25, 2017.

[214] Huaqing Zhang, Jian Wang, Zhanquan Sun, Jacek M Zurada, and Nikhil R Pal. Feature selection for neural networks using group lasso regularization. *IEEE Transactions on Knowledge and Data Engineering*, 32(4):659–673, 2019.

[215] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 531–540, 2018.

[216] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[217] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blindx deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[218] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–17, 2019.

[219] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[220] Maosen Li, Siheng Chen, Ya Zhang, and Ivor W Tsang. Graph cross networks with vertex infomax pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[221] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 776–794. Springer, 2020.

[222] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[223] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4116–4126. PMLR, 2020.

[224] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4116–4126, 2020.

[225] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[226] Chun-Yi Liu, Chuan Zhou, Jia Wu, Hongtao Xie, Yue Hu, and Li Guo. Cpmf: A collective pairwise matrix factorization model for upcoming event recommendation. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1532–1539, 2017.

[227] Xiaojie Wang, Jianzhong Qi, Kotagiri Ramamohanarao, Yu Sun, Bo Li, and Rui Zhang. A joint optimization approach for personalized recommendation diversification. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 597–609. Springer, 2018.

[228] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, pages 1419–1428, 2017.

[229] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 353–362, 2016.

[230] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 448–456, 2011.

[231] Fernando Ortega, Antonio Hernando, Jesus Bobadilla, and Jeon Hyung Kang. Recommending items to group of users using matrix factorization based collaborative filtering. *Information Sciences*, 345:313–324, 2016.

[232] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. A neural autoregressive approach to collaborative filtering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 764–773, 2016.

[233] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):19, 2016.

[234] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, page 35, 2007.

[235] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[236] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 255–262, 2016.

[237] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLR)*, 9(Nov):2579–2605, 2008.

[238] Daria Sorokina, Rich Caruana, Mirek Riedewald, and Daniel Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1000–1007, 2008.

[239] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[240] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *AI*, 97(1-2):245–271, 1997.

[241] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Proceedings of the European Conference on Machine Learning and Data Mining (ECML PKDD)*, pages 437–452, 2011.

[242] Stephen Merity, Bryan McCann, and Richard Socher. Revisiting activation regularization for language rnns. In *Proceedings of the International Conference on Machine Learning Workshop*, 2017.

[243] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–12, 2017.

[244] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–16, 2017.

[245] David J.C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge university press, 2003.

[246] Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *ASA*, 83(404):1023–1032, 1988.

[247] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014*, 2015.

[248] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016.

[249] Shirui Pan, Jia Wu, and Xingquan Zhu. Cogboost: Boosting for fast cost-sensitive graph classification. In *Transactions on Knowledge and Data Engineering (TKDE)*, pages 2933–2946, 2015.

[250] Shirui Pan, Xingquan Zhu, Chengqi Zhang, and S Yu Philip. Graph stream classification using labeled and unlabeled graphs. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 398–409, 2013.

[251] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

[252] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[253] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of The Web Conference (WWW)*, pages 1785–1797, 2021.

[254] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Transactions on Interactive Intelligent Systems (TIIS)*, pages 1–19, 2015.

[255] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of The Web Conference (WWW)*, pages 22–32, 2005.

[256] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.

[257] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, pages 54–67, 2000.

[258] Roland Bénabou and Jean Tirole. Intrinsic and extrinsic motivation. *The Review of Economic Studies*, pages 489–520, 2003.

[259] Robert J Vallerand. Toward a hierarchical model of intrinsic and extrinsic motivation. *Advances in Experimental Social Psychology*, pages 271–360, 1997.

[260] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2333–2338, 2013.

[261] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the Conference on Recommender Systems (RecSys)*, pages 269–277, 2019.

[262] Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. In *Proceedings of the International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

[263] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 346–353, 2019.

[264] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5712–5723, 2019.

[265] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1779–1788. PMLR, 2020.

[266] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8765–8775, 2020.

[267] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021.

[268] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[269] Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. Reducing word omission errors in neural machine translation: A contrastive learning approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6191–6196, 2019.

[270] Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–13, 2021.

[271] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. Self-supervised

learning for large-scale item recommendations. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 4321–4330, 2021.

[272] Fangquan Lin, Wei Jiang, Jihai Zhang, and Cheng Yang. Dynamic popularity-aware contrastive learning for recommendation. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, pages 964–968, 2021.

[273] Wei Jiang, Fangquan Lin, Jihai Zhang, Cheng Yang, Hanwei Zhang, and Ziqiang Cui. Dynamic sequential recommendation: Decoupling user intent from temporal context. In *Proceedings of the Industrial Conference on Advances in Data Mining Workshop*, pages 18–26, 2021.

[274] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2172–2180, 2016.

[275] Valentin Thomas, Emmanuel Bengio, William Fedus, Jules Pondard, Philippe Beaudoin, Hugo Larochelle, Joelle Pineau, Doina Precup, and Yoshua Bengio. Disentangling the independently controllable factors of variation by interacting with the world. In *Proceedings of Workshop in the Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[276] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[277] Tan Wang, Zhongqi Yue, Jianqiang Huang, Qianru Sun, and Hanwang Zhang. Self-supervised learning disentangled group representation as feature. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[278] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. Graph convolution machine for context-aware recommender system. *Frontiers of Computer Science*, pages 1–12, 2022.

[279] Frank Wilcoxon. Individual comparisons by ranking methods. *Breakthroughs in Statistics*, pages 196–202, 1992.

[280] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 531–540, 2018.