Accurate, Efficient, and Explainable Modelling of Context-Dependent Preferences Using Matrix Factorisation

Farhad Zafari

Submitted in fulfilment of the requirements of the degree of

Doctor of Philosophy

Faculty of Science, Engineering and Technology
Swinburne University of Technology
Australia

January 11, 2019
Abstract

Nowadays, consumers are overwhelmed with a large number of products/services available on the web, and recommender systems help them choose the right products/services to buy. They also help businesses match their products/services to the right consumers and hence increase their profits. A dilemma faced in preference modelling in recommender systems is whether to choose an understandable/explainable simple system while sacrificing accuracy, or an accurate complex model while sacrificing explainability. According to our investigations, consumer preferences are generally comprised of several aspects. These aspects describe a user’s preferences over the different options, their characteristics and the dependencies between the characteristics, the influence of social relationships on preferences, the temporal properties of preferences, and the biases users have when specifying preferences on options. A general problem is how to model these aspects efficiently using an accurate model that can be easily explained to users and businesses. The explainability of recommendations is important, because explanations to users have been shown to improve the system transparency and usability. Although each of the preference aspects has been investigated in isolation, there is currently no general architecture that unifies these aspects into a single model. Developing such a multi-aspect architecture not only helps us achieve a high recommendation quality, but also potentially enables us to provide intuitive reasons for the recommendations.

This research makes several important contributions to the field. Hidden aspects in consumer feedback are used to make more accurate recommendations to the users. One aspect is how the users perceive the items in terms of the features they have. We show the importance of capturing these features in the latent factor models, and how these models can overcome the degrading effects of using large numbers of factors. Other aspects relate to the social influence and temporal properties of preferences. We model the social influence on all aspects of preferences, and capture the drifting behavior of aspects over time. We make several assumptions regarding the mathematical properties of these aspects. These assumptions enable us to efficiently capture all the hidden preference aspects, their social
and temporal properties, and interactions. The model strikes a balance between efficiency and accuracy and achieves higher accuracies than previous models. The preference aspects are captured in an integrated component-based architecture, where each aspect corresponds to a component. The component-based design permits us to consider the interplay between various preference aspects and the domain-dependent interaction effects between them. This component-based architecture facilitates making more accurate recommendations by considering interactions. It also enhances explainability, since each component captures one meaningful aspect that can be explained to the users. The intuitive and comprehensive explanations to the users and businesses regarding the preference aspects are provided for both businesses and consumers. The meaning of explanations is also enhanced by using the verbal reviews by the users to improve the understandability.
Acknowledgements

PhD is a long, exciting, rewarding, and at the same time very challenging journey. Like every other journey that we go through in life, this journey takes us through many ups and downs, turns, bright and dark moments. Sometimes, we go through exhilarating moments when we make a break through and get a nice idea working. We get to look at what we have created, and get filled with the hope, excitement, and great sense of achievement. However, there are also many more moments when the ideas simply fail. Those moments are the dark moments of this journey, when we need the light, some hope to make it through, to keep us moving on. Finishing this journey would have been impossible, without the incredible people that provided me with support and strength.

First of all, I would like to thank my dearest family, my father Abdollah, my mother Zhila, and my brother, Hadi for their compassion and love. I would like to specially thank my parents for teaching me to be strong through the difficulties of this life, and never lose hope no matter how difficult are the times we go through.

I am also very grateful to my principal coordinating supervisor, Dr Irene Moser and my associate supervisor Prof Timos Sellis for providing me with strong and continuous support throughout my PhD, and lightening up the dark turns of my PhD journey, with the insights and encouragements that they have given me. I am in particular very thankful to Dr Irene Moser for believing in me, for all the time and the hard effort that she put on this research, and for the invaluable advice and the brainstorming meetings that helped me in shaping the ideas around this PhD and progressing through them. I really appreciate her encouraging me and providing me with the motivation and strength to carry on with the thesis, especially at the difficult times of my PhD.

Finally, I would like to thank my dear friends and colleagues at Swinburne University of Technology, for supporting me through this journey and cheering me up at the times of frustration and for sparking hope inside me when I needed it most.
Declaration

This is to certify that this thesis contains no material which has been accepted for the award of any other degree or diploma and that to the best of my knowledge this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Farhad Zafari
January 11, 2019
List of Publications

Journals:


Conference Proceedings:


# Contents

## 1 Introduction

1.1 Research questions ................................................. 4
  1.1.1 Accurate user preference modelling .......................... 4
  1.1.2 Efficient user preference modelling ......................... 5
  1.1.3 Explainable user preference modelling ..................... 5

1.2 Research objectives ............................................... 6

1.3 Thesis outline ..................................................... 7

1.4 Approach .......................................................... 9

1.5 Methodology ........................................................ 11
  1.5.1 Implementation ................................................ 11
  1.5.2 Evaluation metrics ............................................ 11
  1.5.3 Datasets ....................................................... 12

1.6 Contributions ...................................................... 14

## 2 Research Background ........................................... 15

2.1 Modelling user preferences .................................... 15
  2.1.1 Preference modelling based on individual intelligence ...... 15
  2.1.2 Preference modelling based on collective intelligence ...... 16

2.2 Explaining user preferences .................................. 37
  2.2.1 Collaborative filtering explanations ......................... 37
  2.2.2 Content-based explanations ................................ 39
  2.2.3 Case-based explanations .................................... 40
  2.2.4 Knowledge and utility-based explanations .................. 40

2.3 Collaborative filtering based on deep learning ............... 40

2.4 Summary .......................................................... 45
## Feature-Aware Modelling of Preferences

### 3.1 Feature-aware PMF (FAPMF)

### 3.2 Experiments

#### 3.2.1 Results

#### 3.2.2 Effect of parameter \( r \) in FAPMF

### 3.3 Summary of key points

### 3.4 Conclusion

## Modelling Conditional Feature Value Preferences

### 4.1 Motivating example

### 4.2 Conditional preferences model

#### 4.2.1 Mathematical details

#### 4.2.2 Absence of model of feature values in PMF

#### 4.2.3 Ability of the model to capture preferences over feature values

#### 4.2.4 Correctness of CFVSVD model

#### 4.2.5 Illustrative example

### 4.3 Experiments

#### 4.3.1 Datasets

#### 4.3.2 Results

### 4.4 Summary of key points

### 4.5 Conclusion

## Modelling Socially-Influenced Preferences in Recommender Systems

### 5.1 Introduction

#### 5.1.1 Motivating example

### 5.2 Biased probabilistic matrix factorisation with socially-influenced preferences over feature values

#### 5.2.1 Model training of TFVSVD

### 5.3 Biased probabilistic matrix factorisation with socially-influenced conditional preferences over feature values

#### 5.3.1 Model training of CTFVSVD

### 5.4 Experiments

#### 5.4.1 Comparisons

#### 5.4.2 Setup

### 5.5 Results
5.6 Summary of key points .................................................. 110
5.7 Conclusion ................................................................. 110

6 Modelling Drifting Preferences in Recommender Systems 113

6.1 Introduction ............................................................... 113

6.2 Modelling time-aware preference aspects in CTFVSVD ............. 114

6.2.1 Aspect interactions and high-level view of the model ............. 115

6.2.2 TCTFVSVD model formulation ..................................... 117

6.2.3 TCTFVSVD model training ........................................... 121

6.2.4 TCTFVSVD algorithm .................................................. 128

6.2.5 Computational complexity analysis ................................ 130

6.3 Experiments ............................................................... 133

6.3.1 Setup ........................................................................ 133

6.3.2 Comparisons ............................................................... 133

6.3.3 Discussion ................................................................. 134

6.3.4 Statistical analysis ....................................................... 137

6.3.5 Dynamic aspects ......................................................... 138

6.3.6 Effect of the size of the training dataset ......................... 139

6.3.7 Run time analysis ......................................................... 146

6.4 Summary of key points .................................................. 148

6.5 Conclusion ................................................................. 148

7 A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems 151

7.1 Introduction ............................................................... 151

7.2 Adaptive training of preference aspects and interactions ............ 152

7.2.1 EA ........................................................................ 152

7.2.2 ATCTFVSVD ............................................................. 155

7.2.3 Analysing the interaction effects using ATCTFVSVD ........... 155

7.3 Experiments ............................................................... 156

7.3.1 Experimental setting ................................................... 156

7.3.2 Results ................................................................. 156

7.4 Summary of key points .................................................. 161

7.5 Conclusion ................................................................. 161
8 An Integrated Architecture for Preference Model Representation and Explanation

8.1 Introduction ........................................................................................................... 163
8.2 Recommendation and explanation framework ......................................................... 164
  8.2.1 Explicit model .................................................................................................. 165
  8.2.2 Latent and explicit models combined ............................................................... 168
  8.2.3 Correlations between latent and explicit features ............................................. 170
8.3 Extracting explanations ........................................................................................... 172
  8.3.1 Rating-based explanation model .................................................................... 172
  8.3.2 Review-specific: Word Clouds ....................................................................... 177
  8.3.3 Review-specific: Topic contributions ............................................................... 178
  8.3.4 User-specific aspects ...................................................................................... 179
  8.3.5 Community-specific aspects ......................................................................... 183
8.4 Summary of key points ............................................................................................ 187
8.5 Conclusion .............................................................................................................. 187

9 Conclusions and Future Directions ........................................................................... 189

9.1 Research contributions ............................................................................................ 189
  9.1.1 Contribution 1: feature-awareness in recommendations ............................... 190
  9.1.2 Contribution 2: conditional feature value preferences in recommendations ...... 190
  9.1.3 Contribution 3: social influence on the preference aspects ......................... 191
  9.1.4 Contribution 4: temporal dynamics of preference aspects ............................ 191
  9.1.5 Contribution 5: interaction effects of preference aspects ................................ 192
  9.1.6 Contribution 6: providing intuitive visual explanations for the users and businesses .................................................... 193
9.2 Answers to research questions .................................................................................. 193
9.3 Future research avenues .......................................................................................... 196
List of Figures

1.1 The abstract representation of problem space in preference modelling ........ 3
1.2 The graphical demonstration of the project outline ............................. 7

2.1 Bayesian network of PMF [79] (grey circles represent known matrices) ..... 22
2.2 Architecture of DMF [104] .......................................................... 42
2.3 Architecture of CoNN [124] .......................................................... 44

3.1 The Lamé oval used in line 8 in algorithm 2 for different values of \( r \) \( (r = 0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) \) .............. 49
3.2 Performance of FAPMF with respect to parameter \( r \) ........................ 52
3.3 Performance of FAPMF with respect to parameter \( r \) for each factor in Filmtrust dataset .............................................................. 54
3.4 Performance of FAPMF with respect to parameter \( r \) for each factor in Movie-lens100k dataset .............................................................. 55

4.1 Bayesian network of CFVSVSVD ...................................................... 61
4.2 Lemma 2.1 definitions and formulation .............................................. 67
4.3 The possible cases for the binary string \( S \) and the position of \( X \) in the associated real string in each case ........................................... 68
4.4 Matrices \( P, W, Z, Q \) and \( Y \), and vectors \( bu \) and \( bi \) for the illustrative example .............................................................. 71
4.5 The individual preference functions for each one of 6 features in the example given in the example .............................................................. 73

5.1 The relationship between the models ............................................... 82
5.2 Bayesian network of TFVSVSVD ..................................................... 85
5.3 Bayesian network of CTFVSVSVD .................................................. 91
5.4 The social network graph in the illustrative example .......................... 95
5.5 The matrices \( y, \omega, P, Q, W, Z, Y, R, R \), and \( T \) and the vectors \( bu \) and \( bi \) in the illustrative example .............................................................. 96
5.1 Table 5.1: Performance comparison of TFVSVD on Epinions dataset for a) all users, and b) cold-start users 

5.2 Table 5.2: Performance comparison of TFVSVD on Filmtrust dataset for c) all users, and d) cold-start users 

5.3 Table 5.3: Performance comparison of TFVSVD on Ciao dataset for c) all users, and d) cold-start users 

5.4 Table 5.4: Number of users, items and trust relationships on the Epinions, Filmtrust, and Ciao datasets 

5.5 Table 5.5: General statistics of the Epinions, Filmtrust, and Ciao datasets 

5.6 Comparison of the performance of latent factor models in terms of different number of factors on Epinions dataset, for a) MAE and b) RMSE for all users, and c) MAE and d) RMSE for cold-start users 

5.7 Comparison of the performance of latent factor models in terms of different number of factors on Filmtrust dataset, for a) MAE and b) RMSE for all users, and c) MAE and d) RMSE for cold-start users 

5.8 Comparison of the performance of latent factor models in terms of different number of factors on Ciao dataset, for a) MAE and b) RMSE for all users, and c) MAE and d) RMSE for cold-start users 

5.6 Table 5.6: t-test results for a) TFVSVD, and b) CTFVSVD against the best performing state of the art models 

5.7 Performance of CTFVSVD and TFVSVD compared with the state of the art on Epinions dataset, in terms of a) MAE, b) RMSE 

5.8 Performance of CTFVSVD and TFVSVD compared with the state of the art on Filmtrust dataset in terms of a) MAE, b) RMSE 

5.9 Performance of CTFVSVD and TFVSVD compared with the state of the art on Ciao dataset in terms of a) MAE, b) RMSE 

5.10 Box plots of CTFVSVD and TFVSVD compared with the state of the art on Epinions dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users 

5.11 Box plots of CTFVSVD and TFVSVD compared with the state of the art on Filmtrust dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users
5.12 Box plots of CTFVSVD and TFVSVD compared with the state of the art on Ciao dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users

6.1 The preference aspects and their interplay in TCTFVSVD

6.2 a) The high-level representation of TCTFVSVD and b) its flow chart

6.3 Bayesian network of TCTFVSVD

6.4 An example of drifting preferences in Eq. 6.5 and Eqs. 6.7 to 6.9 for a) positive $\alpha$ values and b) negative $\alpha$ values

6.5 Box plots of the TCTFVSVD’s combinations and CTFVSVD versus TrustSVD on the Ciao dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL)

6.6 Box plots of the TCTFVSVD’s combinations and CTFVSVD versus TrustSVD on the Epinions dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL)

6.7 Box plots of the TCTFVSVD’s combinations and CTFVSVD versus TrustSVD on the Flixster dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL)

6.8 Comparisons of the MAE values of TCTFVSVD’s combinations in a,b) Ciao, c,d) Epinions, and e,f) Flixster datasets for all users (MAE-ALL) and cold-start users (MAE-CS)

6.9 Comparisons of the MAE values of TCTFVSVD’s combinations in a,b) Ciao, c,d) Epinions, and e,f) Flixster datasets for all users (RMSE-ALL) and cold-start users (RMSE-CS)

6.10 Effect of the training amount on error on Flixster dataset for a) MAE for all users, b) MAE for cold-start users, c) RMSE for all users, d) RMSE for cold-start users

6.11 Effect of the training amount on error on Ciao dataset for a) MAE for all users, b) MAE for cold-start users, c) RMSE for all users, d) RMSE for cold-start users

6.12 Run time of TCTFVSVD with respect to the number of factors on the Ciao dataset

6.13 Comparison of TCTFVSVD’s run time against state of the art models on a) Ciao dataset, and b) Epinions dataset
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>The schematic representation of ATCTFVSD</td>
<td>153</td>
</tr>
<tr>
<td>7.2</td>
<td>The schematic representation of ATCTFVSD</td>
<td>154</td>
</tr>
<tr>
<td>7.3</td>
<td>The results of model combinations for a) MAE and b) RMSE</td>
<td>160</td>
</tr>
<tr>
<td>8.1</td>
<td>The processing of explicit feedback in ReEx</td>
<td>166</td>
</tr>
<tr>
<td>8.2</td>
<td>The ReEx Framework</td>
<td>169</td>
</tr>
<tr>
<td>8.3</td>
<td>An example of the constructed preference model for 5 sample users over 4 movies</td>
<td>170</td>
</tr>
<tr>
<td>8.4</td>
<td>The distribution of the correlations between the four top rated restaurants using the latent model, and the explicit model for users in the Yelp restaurants dataset</td>
<td>171</td>
</tr>
<tr>
<td>8.5</td>
<td>Aspect contributions for this user, Yelp restaurants dataset</td>
<td>174</td>
</tr>
<tr>
<td>8.6</td>
<td>Development of popularity of the four recommendations over time</td>
<td>176</td>
</tr>
<tr>
<td>8.7</td>
<td>The result of semantic clustering on the Yelp restaurants dataset</td>
<td>177</td>
</tr>
<tr>
<td>8.8</td>
<td>The composition of the intrinsic feature preferences of the user (left) and the feature values of the top recommended item (item 94)</td>
<td>179</td>
</tr>
<tr>
<td>8.9</td>
<td>Influential friends (number of friends: 3)</td>
<td>180</td>
</tr>
<tr>
<td>8.10</td>
<td>Aspect contributions</td>
<td>181</td>
</tr>
<tr>
<td>8.11</td>
<td>Similarity of preferences between the user and his community and the user and people outside his community</td>
<td>183</td>
</tr>
<tr>
<td>8.12</td>
<td>The influential friends of this community, among a total of 20 friends, and the average influence of the community of this community, and the outsiders</td>
<td>184</td>
</tr>
<tr>
<td>8.13</td>
<td>Aspect contributions</td>
<td>185</td>
</tr>
<tr>
<td>8.14</td>
<td>Model temporal drift</td>
<td>185</td>
</tr>
<tr>
<td>8.15</td>
<td>Community-specific explanation diagrams for the user in Yelp restaurants dataset</td>
<td>186</td>
</tr>
</tbody>
</table>
List of Tables

2.1 The key notations and symbols used throughout the thesis ........................................... 18
2.2 Classification of the recommendation methods. The areas in which the contribu-
    tions of this thesis fall are yellow-coloured .......................................................... 19
3.1 Performance comparison of FAPMF in a) Filmtrust and b) Movielens datasets .......................... 50
4.1 Example matrix $R$ with 5 users rating 4 movies. 0.00 denotes missing rating .......................... 71
4.2 Estimated ratings matrix $\hat{R}$ as a result of applying CFVSVD .................................. 72
4.3 Comparison of CFVSVD, FVSVD and SVD++ on the CiaoDVD dataset .............................. 75
4.4 Comparison of CFVSVD, FVSVD and SVD++ on the LastFM dataset .................................. 76
4.5 Significances of difference measured using the t-test for pairwise comparisons of models (CFVSVD - SVD++) .......................................................... 77
4.6 Significances of difference measured using the t-test for pairwise comparisons of models (FVSVD - SVD++) .......................................................... 77
4.7 Significances of difference measured using the t-test for pairwise comparisons of models (CFVSVD - FVSVD) .......................................................... 78
6.1 The t values and p values for TCTFVSVD’s combinations vs TrustSVD in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS) .......................................................... 140
6.2 The t values and p values for TCTFVSVD’s combinations vs CTFVSVD in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS) .......................................................... 141
6.3 Comparison of average run times for different models on the Ciao and Epin-
    ions datasets ........................................................................................................... 147
7.1 The results of model combinations in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures .......................................................... 157
8.1 Confusion matrix for the explicit rating prediction using sentiment analysis with IBM’s AlchemyAPI tool
Chapter 1

Introduction

Recommender systems suggest items (movies, books, music, news, services, etc.) that appear most likely to interest a particular user. Matching users with the most desirable items helps enhance user satisfaction and loyalty. In the area of recommender systems, the ability to model users accurately is critical for online retailers and businesses. Most of the online businesses that use recommendations as a sales strategy make a profit from matching customers with the products/services they provide. There are usually an overwhelming number of products/services that are offered by these businesses. Therefore, it is not possible for the users to visit all the products/services to make a right purchase decision, and efficient tools are needed to match the right customers to the right offerings in real time. On the other hand, the recommendations should be understandable for the users and convincing enough to initiate a purchase decision. They should provide the reasons why the offered products/services are the right choices for a particular user. Therefore the retailers have increasingly been investing in improving their recommender systems, and many e-commerce leaders such as Amazon and Netflix have made recommender systems a salient part of their websites.

Recommender systems are usually based on collaborative filtering (CF), where the preferences of a user are predicted by collecting rating information from other similar users or items. Among the CF systems, latent factor models have become popular mainly due to their high prediction accuracy and efficiency. These models explain the ratings by transforming both items and users on a shared latent feature space, which is inferred from the rating patterns. Most of the latent factor models are based on matrix factorisation. Latent factor models are very flexible and enable the incorporation of additional feedback information such as social network data, user and item biases, temporal information, contextual information, and user demographics.
These models rely on modelling latent (hidden) trends and patterns in data, and capturing more of these patterns results in better recommendations, which translates into better customer-offering matching and consequently, more profit for the businesses. This fact motivated us to investigate the preferences thoroughly, and understand what aspects are present, and how they can be captured by the latent factor models to improve the quality of recommendations.

We identify five major aspects to the preferences besides preferences over features that are captured by the basic matrix factorisation models. These aspects consist of feature value preferences, social influence, temporal dynamics, conditional preferences, and user and item biases. Feature value preferences refer to the relative favourability of each one of the item feature values, social influence refers to the influence of social relationships on the preferences of a user, temporal dynamics means the drift of the preferences over time, conditional preferences refers to the dependencies between item features and their values, and user and item biases refer to the systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others [51].

Figure 1.1 illustrates the different aspects present in the data which are captured by the model. For example, this figure shows that Rowan Atkinson (denoted by $f_{v1}$) has starred in the movie Johnny English (denoted by $I_4$). Rowan Atkinson is a value for the 'actor' feature (denoted by $f_1$), and favoured by user $u_1$. User $u_1$ is a friend of user $u_2$, and has influence on her. Therefore, user $u_2$ also becomes a fan of Rowan Atkinson’s due to the social influence of $u_1$. User $u_2$ has highly dynamic preferences and has a tendency to change her preferences a lot over time.

Most of the popular models that are very accurate rely on complex models from which it is difficult to extract reasoning. Therefore, recommender systems practitioners usually encounter a dilemma, which makes them choose between an understandable/explainable simple algorithm while sacrificing accuracy, or choosing an accurate complex model while sacrificing explainability [118, 114, 118]. As we push towards higher accuracy, models become more complex and the explainability becomes more difficult [30]. Although each one of the preference aspects have been investigated in isolation, there is currently no general architecture that unifies these aspects into a single model. Developing an architecture that provides intuitive explanations while ensuring high quality recommendations is a great challenge. Developing such a comprehensive architecture not only helps us achieve the highest recommendation quality, but also potentially enables us to provide intuitive reasons for recommendations. Furthermore, we speculate that there are interaction effects...
of such preference aspects. Developing a unified architecture that incorporates all these aspects would also enable us to further improve the quality of the recommendations, by understanding, and taking advantage of the existing interaction effects. Understanding the preference aspects and their interactions enables us to provide convincing explanations to the users. Explainability of the recommendations is considered an important functionality of recommender systems, since it improves transparency and instills trust in the users and plays a major role in the users’ decision making process. This leads to the formulation of the following research problem:
Chapter 1. Introduction

We lack a general architecture to efficiently mine domain-dependent preferences from different data sources, analyse, and exploit preference aspects, in order to achieve accurate, and explainable recommendations.

1.1 Research questions

The advantages of having a component-based architecture for preference modelling in terms of accuracy and explainability has motivated us to follow the research aim below throughout the construction of the model.

Our main purpose in this project is building an accurate and efficient preference model using the additional information provided by contextual factors and user reviews, and explaining the resulting recommendations and the preference models to the users and businesses in an intuitive and precise manner.

In line with the research aim above, in this research we are mainly interested in the following general research question:

How can we use ratings, reviews, and contextual information to build an accurate, efficient, and easily explainable model of user preferences over items, given the data sparsity problem?

We divide this general question into a few sub-questions. The following sections elaborates on these sub-questions.

1.1.1 Accurate user preference modelling

As shown in Figure 1.1 at least six aspects can be distinguished in human preferences. These aspects interact with each other, and the final preference of a user over an item is the result of the contributions of these aspects and their interplay. A great challenge in
modelling user preferences in recommendation systems is that the ratings matrix is usually sparse, and there is usually not enough information to model preferences of all users from ratings data. The question is, how to accurately model the user preferences from the sparse ratings. Therefore, we articulate the first research question as follows:

**Research Question I**

How can we accurately model the user preferences with different aspects, given the data sparsity problem and interactions between aspects?

1.1.2 **Efficient user preference modelling**

Usually accuracy comes at the cost of efficiency. More accurate models are complex and take a lot of time to train. Furthermore, modelling user preferences in recommendation systems is naturally a difficult problem. The ratings are sparse, and the preferences of millions of users over thousands of items have to be modelled, and the solutions should scale to such large datasets. Therefore, a great challenge faced by recommender systems is how to efficiently model the users while maintaining the accuracy. Thus, the second research question we are interested in is as follows:

**Research Question II**

How can we extract the preference patterns from ratings and social network data in a way that the model is efficient and scalable to large datasets?

1.1.3 **Explainable user preference modelling**

As mentioned, a challenge that recommender systems practitioners face is the dilemma between accuracy and interpretability. They could either choose a simple less accurate model that is explainable, or construct highly accurate but complex models, from which it is difficult to extract recommendations. Explaining recommendations is a critical functionality in recommender systems, and can improve the trust and system usability on the user side. On the business side, it can help the businesses better understand their customers and design more efficient marketing strategies. Therefore, it is important for these
systems to be able to explain the recommendations and represent the models to the users and businesses that are employing them. Finally, we articulate the third research question as follows:

Research Question III

How can we model the preference patterns from rating and social network data and user reviews, in a way that the resulting model and recommendations can be easily explained and represented to the users?

1.2 Research objectives

In this thesis, we are trying to achieve three main objectives, which are represented by the two pillars and the foundation of the imaginary building in Figure 1.2.

- **Efficiency/Accuracy:** Normally, recommender systems are operated by the giant electronic commerce companies such as Yahoo, and Netflix, to model the preferences of millions of users over thousands of items. This means that they process large amount of input data, and consequently, they should be scalable enough to be applicable. Moreover, as we show in Chapter 2, most of the state of the art methods focus on improving the overall prediction accuracy of the constructed preference model. Improving the preference model accuracy results in higher likelihoods of the consumer accepting the suggestion of the product/service, and consequently, increases the revenue of the companies employing the recommendation systems. Evidence suggests that even slight improvements in recommendation accuracy can result in great financial benefits in practice [51, 52]. Therefore, making improvements in recommendation accuracy while keeping the model efficient are two important objectives for recommender systems. Chapters 3 to 7 mostly concentrate on these objective.

- **Explainability:** As we explain in Chapter 2, most of the state of the art models ignore the explainability of the recommendations, in a quest for higher accuracy. The accuracy/interpretability dilemma suggests that improving accuracy usually hampers the explainability of the recommendations [118, 44, 53]. Therefore, the current solutions mostly improve the accuracy at the cost of sacrificing accuracy. As the last objective, in this thesis we aim at proposing a framework that not only improves the
Chapter 1. Introduction

Figure 1.2: The graphical demonstration of the project outline. Grey rectangles represent the research objectives, and the white rectangles and the black triangle represent the steps to construct the preference model and the explanation module respectively. Latent factor models lay the basis for the proposed framework and most of the popular latent factor models (Chapter 2). The framework supports the modelling of the major aspects of preferences, i.e., item features (Chapter 3), feature values and their conditional dependencies (Chapter 4), influence of social relationships on preferences (Chapter 5), and the temporal properties of preferences (Chapter 6), their interactions (Chapter 7), and explaining the recommendations in terms of the modelled aspects (Chapter 8).

accuracy, but also paves the way for providing intuitive explanations. Chapter 9 is more focused on this objective.

1.3 Thesis outline

We start by explaining the basic terminologies and concepts in the area of recommender systems and preference modelling in Chapter 2. We also introduce the main challenges and problems in preference modelling and recommender systems. Then we review and provide a systematic classification of the state of the art models, and point out the advantages and disadvantages of each model, and how each one of these methods go about addressing the main challenges.
In **Chapter 3** we focus on the item features, and discuss how efficient modelling of item features matters in achieving higher accuracies. In particular, we propose a feature-aware extension of the matrix factorisation, to take into account the relevance of item features.

In **Chapter 4**, we concentrate on the values of the item features (e.g. price value), and how they can be efficiently and accurately modelled with a latent factor model. We also look at the dependencies that might exist between the features of items and their values in different preference domains, and how they can possibly be captured. We propose a method to model the conditional preferences over feature-values in latent factor models, and provide mathematical justifications to the formulations used to capture such properties. In particular, we show how such preferences can be extracted without the use of any information other than simple ratings.

In **Chapter 5** we elaborate on the role of social factors in shaping and changing user preferences. First, using experimental analysis on real preference data, we show how the social factors influence the user preferences. Then we address the problem of mining the preference patterns in social networking data. We extend the model in Chapter 4 and propose a novel latent factor model to extract socially-influenced conditional preferences over feature-values, in which all the aspects of human preferences are assumed to be subject to social influence.

In **Chapter 6**, according to the research approach described in Section 1.4, we first experimentally show that preferences are subject to strong dynamicity, and that we can potentially achieve great accuracy improvements by efficiently modelling the temporal properties of preferences. Then we describe how we extend the model proposed in Chapter 5 with the ability to learn personalised temporal patterns for each user, and then use the extracted patterns to improve the accuracy of the modelled preferences. We specifically analyse the temporal properties of each one the preference aspects in isolation, and when they interact with one another.

In **Chapter 7**, we analyse the interactions between preference aspects modelled in Chapter 6.

In **Chapter 8**, we address the challenging problem of the explaining the recommendations and the resulting model to the users. In this chapter we show how the different preference aspects mined from user reviews, ratings, and social network data can be translated into intuitive explanations through visual graphs and diagrams to benefit both the user and the business sides.

And finally in **Chapter 9** we summarise the main results and findings, provide answers
to the research questions, and suggest some of the interesting possible future directions of the present work.

1.4 Approach

In order to solve the research problem in this thesis (Section ??) we followed a basic problem solving approach. We followed an incremental and component-based approach in building the proposed architecture. This approach consisted of the following four major steps:

- **Literature Review:** We carried out a systematic and comprehensive literature review of the existing preference modelling techniques in the area of recommender systems, their advantages and shortcomings with a focus on accuracy and explainability. This resulted in the identification of the most promising methods that could form a basis for the proposed architecture, and the major aspects of preferences, and how they have been approached by different researchers in the literature.

- **Experimental Analysis:** Based on the potential preference aspects and existing models, we designed and implemented experiments on publicly available benchmark datasets. These experiments provided insights into the preference aspects and how they contribute to the preference patterns in the users and items data.

- **Incremental Modelling:** We constructed the model using a component-based approach, where the aspects were added to the model incrementally. The model was designed using a multi-component architecture, in which every component was logically responsible for capturing one preference aspect, and the architecture incorporated the functionality of turning off any components arbitrarily. At each step, the model was extended in a way that made it possible to revert to the previous model by switching off the component that captured the respective aspect. This component-based approach enabled us to understand and analyse the aspects separately as well as their interactions with each other. It also helped us achieve the accuracy and explainability objectives.

- **Model Evaluation and Deployment:** After constructing the model, the interaction effects of the model aspects were analysed by comparing the accuracy of the models using different aspects. Our approach here was designing efficient methods to make sure that optimal model aspects were employed in each domain and dataset. The
performance of the model was compared with the state of the art models in comprehensive experimental settings, and the computational efficiency of the proposed architecture was investigated.
1.5 Methodology

The methodology involves choice of the implementation tools (Section 1.5.1), metrics used to evaluate the models (Section 1.5.2), and benchmark datasets (Section 1.5.3).

1.5.1 Implementation

To implement all the methods proposed in this thesis, we use LibRec \[29\] which is an open source and comprehensive java library for recommender systems. The repository of LibRec includes more than 70 state of the art recommendation algorithms.

1.5.2 Evaluation metrics

Throughout this thesis, two standard and popular measures are used to measure and compare the performance of our methods compared with the state of the art. These metrics are Mean Absolute Error (MAE) and Root Means Square Error (RMSE) and defined using Eqs. 1.1 and 1.2 respectively.

\[
MAE = \frac{1}{|R_{test}|} \sum_{u=1}^{N} \sum_{j=1}^{M} | R_{uj} - \hat{R}_{uj} | \tag{1.1}
\]

\[
RMSE = \sqrt{ \frac{1}{|R_{test}|} \sum_{u=1}^{N} \sum_{j=1}^{M} ( R_{uj} - \hat{R}_{uj} )^2 } \tag{1.2}
\]

where \(|R_{test}|\) denotes the number of known ratings in test matrix, and \(\hat{R}_{uj}\) and \(R_{uj}\) respectively denote the predicted and known ratings given by user \(u\) to item \(j\), \(N\) is the number of users and \(M\) is the number of items.

Eqs. 1.1 and 1.2 measure the accuracy for 'all users’. Throughout the experiments, we also evaluate models based on the accuracy of 'cold-start' users. Cold-start users are defined as the users who have less than 5 ratings in the ratings matrix \(R\). Therefore, \(MAE\) and \(RMSE\) values for cold-start users denoted by \(MAE_c\) and \(RMSE_c\) are obtained according to Eqs. 1.3 and 1.4 (\(U^c\) is the set of all cold-start users).

\[
MAE_c = \frac{1}{|R_{test}|} \sum_{u \in U^c} \sum_{j=1}^{M} | R_{uj} - \hat{R}_{uj} | \tag{1.3}
\]

\[
RMSE_c = \sqrt{ \frac{1}{|R_{test}|} \sum_{u \in U^c} \sum_{j=1}^{M} ( R_{uj} - \hat{R}_{uj} )^2 } \tag{1.4}
\]
1.5.3 Datasets

In this thesis, we collected and used a set of popular publicly available datasets in recommendation systems to test the methods. These datasets are as follows:

- **Epinions** dataset [60, 61, 10] consists of 664,824 ratings from 40,163 users on 139,738 products of different categories (games, movies, music, books, ...). Ratings are integer values between 1 and 5, and data density is 0.011%. Epinions also enables the users to issue explicit trust statements about other users. This dataset includes 487,183 trust ratings. The density of the trust network is 0.03%.

- **Filmtrust** [28, 31, 25, 46] is a small dataset crawled from the FilmTrust website in June, 2011. The dataset includes 35,497 ratings given by 1,508 users on 2,071 movies. The ratings are real values between 0.5 and 4, with lower values given for less favourable movies. The ratings matrix is extremely sparse and only 1.14% of the ratings in the user ratings matrix are known. Similar to Epinions, in the Filmtrust dataset the users can also express their trust in each other. Therefore, this dataset also includes 1,853 directed trust ratings. Data density is 1.13%, and the density of the trust network is 0.08%.

- **Movielens** [33] dataset is collected and made available from the Movielens [33] website by GroupLens Research. This dataset consists of 100,000 ratings from 943 users on a total of 1,682 movies. The rating scale is [1,5], 5 for the most preferred movie, and 1 for the least preferred one, and each user in the dataset has rated at least 20 movies, and the density of the rating matrix is 6.30%.

- **Ciao** [38, 87] is a dataset crawled from the ciao.co.uk website. This dataset includes 35,835 ratings given by 2,248 users over 16,861 movies. Similar to the Epinions and Filmtrust, Ciao also allows the users to establish social relations (i.e. trust relationships) with others. The number of trust relationships in Ciao is 57,544. Therefore the dataset density of ratings and trust relationships are 0.09% and 1.14% respectively. The ratings are integer values between 1 and 6.

- **CiaoDVD** [30] is a dataset crawled from the entire category of DVDs from the dvd.ciao.co.uk website in December 2013. This dataset includes 72,345 ratings given by 17,615 users over 16,121 movies.

- **LastFM** [100] dataset includes 821,011 ratings on 107,398 songs given by 992 users,
and is collected through the music website, www.last.fm. The ratings in both datasets are integer values between 1 and 5.

- **Flixster** [42] is a social movie site which allows users to rate movies and share the ratings with each other, and become friends with others with similar movie tastes. Flixster dataset which is collected from Flixster website includes 8,196,077 ratings issued by 147,612 users on 48,794 movies. The social network also includes 7,058,819 friendship links. The density of the ratings matrix and social network matrix are 0.11% and 0.001% respectively. The ratings are between 1 and 5.

- **Yelp** [106] dataset is a comprehensive and popular dataset presented by Yelp.com for the Yelp dataset Challenge. This dataset includes 2,167,077 ratings given by 544,818 users on 76,236 items. The items in this dataset cover a broad range of businesses such as restaurants, clubs and pubs, hotels, fast food, golf, fitness and instruction, shopping, home services, internet services, etc. The Yelp dataset also includes an embedded social network which allows the users to be connected with one another. This social network includes 3,563,817 social relationships. Each rating is also associated with a review in natural language.
1.6 Contributions

The most important contributions of this thesis are listed below. Each contribution is explained in details in chapter 9, where we summarise the research findings and conclude this thesis.

1. Incorporating feature-awareness into the latent factor models to achieve higher accuracy (Chapter 3).

2. Incorporating conditional preferences over feature values into the latent factor models to achieve higher accuracy (Chapter 4).

3. Incorporating the social influence of conditional preferences over feature values into the recommender systems to improve accuracy and explainability (Chapter 5).

4. Analysing the temporal drift of the preference aspects and exploiting such drifting behaviours to achieve higher accuracy (Chapter 6).

5. Analysing the interactions between preference aspects in order to achieve higher accuracy and explainability (Chapter 7).

6. Translating the modelled preference aspects and resulting recommendations into intuitive visual explanations for users and businesses (Chapter 8).

7. Introducing an efficient general component-based architecture enabling explainable and accurate recommendations to the users (chapters 4, 5, 6, and 8).
Chapter 2

Research Background

2.1 Modelling user preferences

The following sections provide an overview of the state of the art in the area of user preference modelling. The existing preference modelling methods are broadly classified into the systems based on individual intelligence (Section 2.1.1), and systems based on collective intelligence (Section 2.1.2).

2.1.1 Preference modelling based on individual intelligence

There are currently a group of preference modelling methods that only rely on the preference data from a single user. We refer to these methods as individual intelligence methods. In the following section, we briefly introduce a few of the systems in this category. Since our contributions in this thesis fall into collective intelligence systems category (Section 2.1.2), we do not intend to provide a comprehensive review of the individual intelligence systems here. We just provide a brief introduction of content-based filtering in Section 2.1.1.1 Then we elaborate on the collaborative filtering methods in Section 2.1.2

2.1.1.1 Content-based filtering methods

Content-based filtering, or cognitive filtering, originates in information retrieval and text processing, and recommends items based on a comparison between the content of the items and a user profile. These methods analyse the content of the items that the user has liked before, build a user profile, and recommend items that have similar descriptions to the user profile. Content-based methods are more suitable for items that can be described well in natural language.

Content-based filtering uses different types of models to calculate the similarity between
item descriptions in order to generate recommendations. Probabilistic models such as neural networks, decision trees, and Bayesian learning or vector space model such as term frequency inverse document frequency (TF-IDF) can be used to train the content-based recommender systems [75, 39].

Content-based recommender systems have a number of disadvantages that limit their applicability in industry. The accuracy of the recommendations generated by these techniques greatly depends on the item descriptions. Furthermore, the item descriptions do not include all the information regarding how the users decide. Another problem is that content-based systems will not find anything new to suggest to the user, because they only match the item descriptions with the user’s shopping history. This problem is referred to as over-specification. [7]. Furthermore, they need an in-depth knowledge and description of the features of the items in the profile [39].

2.1.2 Preference modelling based on collective intelligence

Collaborative Filtering (CF) approaches collect preference information from many users, and use the collected data to model the preferences of a user. CF approaches are usually preferred to content-based methods because of their impressive performance [22], and are extensively researched in academia and widely used in industry [104].

These methods are widely adopted to build recommender systems and can be broadly classified into memory-based and model-based approaches.

Memory- or instance-based learning methods (e.g. Item knn [29]) predict the user preferences based on the preferences of other users or the similarity of the items rated by other users. Item-based approaches in memory-based CF [20] calculate the similarity between the items, and recommend the items similar to the items that the user has liked in the past.

User-based approaches (e.g. User knn [29]) recommend items that have been liked by similar users [65]. The accuracy of these methods is hampered by data sparsity. Also, these methods do not scale well for real-world applications and demand on the resources increases with increased data volume. Consequently, state of the art algorithms are either pure model-based algorithms or a hybrid of some pre-computation combined with a memory-based approach [2]. These models are generally more accurate than the content-based filtering approaches [12].

Model-based CF learns the parameters of a model and only stores those parameters. Algorithms in the category of model-based CF include the clustering, aspect and latent
Chapter 2. Research Background

17

factor models [65, 43, 2]. Latent factor models as an example of model-based collaborating filtering try to explain the ratings by characterising both users and items on a number of latent factors which are inferred from the rating patterns. Recently, latent factor models based on matrix factorisation have gained much popularity as they usually outperform traditional memory-based methods, and have achieved higher performance in some benchmark datasets [52]. Latent factor models are believed to be the most popular approach among CF method due to their accuracy, scalability, and flexibility [104].

We review the state of the art from the following perspectives. First we briefly introduce the models that capture the users’ preferences over different item features (Section 2.1.2.1). Then we review the perspective of current work on feature-awareness in latent factor models (Section 2.1.2.2). We look at the models that consider the dependencies between item features (Section 2.1.2.3). We consider the works that address the problem of mining preferences over feature values (Section 2.1.2.4), and then we look into some of the models that incorporate the social network information into recommender systems (Section 2.1.2.5), and finally we review the time-aware recommendation systems (Section 2.1.2.6).

The classification of the models reviewed in sections 2.1.2 and 2.2 is given in Table 2.2. The areas coloured in yellow show where the contributions in this thesis fall. A summary of the list of the most important variables is also given in Table 2.1.

2.1.2.1 Feature preference modelling techniques

The basic latent factor methods capture the users’ preferences over different item features. For example, in general, the price of an item may be more important for the users than the warranty or customer service. All methods that model the users’ preferences over different item features are memory-based latent factor models based on matrix factorisation. Several matrix factorisation methods have been proposed in different problem settings, such as Probabilistic Matrix Factorisation (PMF) [79, 118], Biased Probabilistic Matrix Factorisation (BiasedMF) [52], Singular Value Decomposition (SVD) [80] and SVD++ [51, 49], Bayesian Probabilistic Matrix Factorisation [78], and Non-negative Matrix Factorisation (NMF) [54].

Probabilistic Matrix Factorisation (PMF)

PMF [79] has become the foundation for many of recent latent factor methods. In rating-based recommender systems, the observed ratings are represented by the ratings matrix $R$, \[ R \in \mathbb{R}^{m \times n} \], where $m$ and $n$ are the number of users and items, respectively.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>number of users</td>
</tr>
<tr>
<td>$M$</td>
<td>number of items</td>
</tr>
<tr>
<td>$D$</td>
<td>number of latent factors</td>
</tr>
<tr>
<td>$t_{uj}$</td>
<td>the time at which user $u$ rated item $j$</td>
</tr>
<tr>
<td>$P_{uf}(t)$</td>
<td>dynamic preference of user $u$ over latent feature $f$</td>
</tr>
<tr>
<td>$Q_{jf}$</td>
<td>value of feature $f$ for item $j$</td>
</tr>
<tr>
<td>$W_{uf}(t)$</td>
<td>dynamic gradient value to capture the preference of user $u$ over value of feature $f$</td>
</tr>
<tr>
<td>$Z_{uf}(t)$</td>
<td>dynamic intercept value to capture the preference of user $u$ over value of feature $f$</td>
</tr>
<tr>
<td>$y_{ij}$</td>
<td>implicit feedback of the users regarding latent feature $f$ of item $j$</td>
</tr>
<tr>
<td>$Y_{jf}$</td>
<td>feature-specific dependency matrix entry, to capture conditional preferences</td>
</tr>
<tr>
<td>$T_{uv}$</td>
<td>trust value between user $u$ and user $v$</td>
</tr>
<tr>
<td>$</td>
<td>T_u</td>
</tr>
<tr>
<td>$I_u$</td>
<td>the vector of ratings given by user $u$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>the social influence of user $u$ on the other users according to the latent factor model</td>
</tr>
<tr>
<td>$\mu$</td>
<td>the average ratings given by all users to all items</td>
</tr>
<tr>
<td>$bu_{u}(t)$</td>
<td>user $u$’s dynamic rating bias</td>
</tr>
<tr>
<td>$b_{ij}(t)$</td>
<td>item $j$’s dynamic rating bias</td>
</tr>
<tr>
<td>$R_{uj}$</td>
<td>the real rating value given by user $u$ on item $j$</td>
</tr>
<tr>
<td>$R_{uj}(t)$</td>
<td>the predicted rating value given by user $u$ on item $j$ at time $t$</td>
</tr>
<tr>
<td>$\xi_{uj}(t)$</td>
<td>dynamic intrinsic preferences of user $u$ over item $j$</td>
</tr>
<tr>
<td>$\theta_{uj}(t)$</td>
<td>dynamic socially-influenced preferences of user $u$ over item $j$</td>
</tr>
<tr>
<td>$\kappa_{uj}(t)$</td>
<td>other dynamic factors influencing user preferences, e.g. conditional preferences</td>
</tr>
<tr>
<td>$w$</td>
<td>a word used in user reviews</td>
</tr>
<tr>
<td>$P_{uw}$</td>
<td>the estimated sentiment of user $u$ towards feature $w$ according to the explicit model</td>
</tr>
<tr>
<td>$Q_{jw}$</td>
<td>the estimated sentiment towards value of feature $w$ for item $j$ according to the explicit model</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>the collection of words used in the reviews</td>
</tr>
<tr>
<td>$\Psi_u$</td>
<td>the collection of words used by user $u$ in his reviews</td>
</tr>
<tr>
<td>$\Psi_j$</td>
<td>the collection of words in the reviews on item $j$</td>
</tr>
<tr>
<td>$Q_{jf}'$</td>
<td>the average preferences of all users over value of feature $f$ for item $j$</td>
</tr>
<tr>
<td>$c$</td>
<td>a cluster of words extracted from reviews, which represents one topic</td>
</tr>
<tr>
<td>$C(\Psi)$</td>
<td>the collection of clusters of the words used in the reviews</td>
</tr>
<tr>
<td>$P_{uc}'$</td>
<td>the estimated sentiment of user $u$ towards feature $c$ according to the explicit model</td>
</tr>
<tr>
<td>$Q_{jc}'$</td>
<td>the estimated sentiment towards value of feature $c$ for item $j$ according to the explicit model</td>
</tr>
<tr>
<td>$a(w)$</td>
<td>the total number of times word $w$ has appeared in user reviews</td>
</tr>
<tr>
<td>$b(w)$</td>
<td>the sentence-based sentiment of word $w$</td>
</tr>
<tr>
<td>$A_{uw}$</td>
<td>the frequency of word $w$ in the reviews by user $u$</td>
</tr>
<tr>
<td>$B_{jw}$</td>
<td>the frequency of word $w$ in the reviews on item $j$</td>
</tr>
<tr>
<td>$q_w$</td>
<td>the normalised frequency of word $w$ considering both users and items dimensions</td>
</tr>
<tr>
<td>$E$, $E_u$</td>
<td>total model error, and model error for user $u$</td>
</tr>
<tr>
<td>$E_R$, $E_T$</td>
<td>total model rating error, and total model social error</td>
</tr>
<tr>
<td>$C_x$</td>
<td>contribution value for aspect $x$</td>
</tr>
<tr>
<td>$\pi_x$</td>
<td>model parameters and hyper-parameters for aspect $x$</td>
</tr>
<tr>
<td>$F_{uv}, F_{uv}'$</td>
<td>social influence of user $v$ on user $u$, normalised social influence of user $v$ on user $u$</td>
</tr>
</tbody>
</table>

Table 2.1: The key notations and symbols used throughout the thesis
<table>
<thead>
<tr>
<th>Pref. Modelling</th>
<th>Feature Pref. Modelling</th>
<th>Model-based</th>
<th>[80, 54, 52, 119, 79, 49, 52, 51, 87]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Pref. Modelling</td>
<td>Model-based</td>
<td>107, 61, 12, 27, 11, 56, 91, 13</td>
<td></td>
</tr>
<tr>
<td>Feature value Pref. Modelling</td>
<td>Model-based</td>
<td>113, 95</td>
<td></td>
</tr>
<tr>
<td>Social Pref. Modelling</td>
<td>Model-based</td>
<td>Co-factorisation</td>
<td>65, 65, 65, 63, 43, 105</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ensemble</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Regularisation</td>
<td>12, 66, 63, 62, 122, 23, 121, 76, 120</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Probabilistic</td>
<td>34, 123, 59, 125</td>
</tr>
<tr>
<td>Dynamic Pref. Modelling</td>
<td>Model-based</td>
<td>Memory-based</td>
<td>92, 28, 98, 41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Models based on PMF</td>
<td>50, 47, 50, 62, 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Models based on BPMF</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Models based on TMF</td>
<td>45, 57, 74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Models based on BTMF</td>
<td>103</td>
</tr>
<tr>
<td>Pref. Explanation</td>
<td>CF</td>
<td>Memory-based</td>
<td>[55]</td>
</tr>
<tr>
<td></td>
<td>Model-based</td>
<td>Rating-based</td>
<td>[14, 4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Review-based</td>
<td>118, 24, 17</td>
</tr>
<tr>
<td></td>
<td>Memory-based</td>
<td>Item-based</td>
<td>80, 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User-based</td>
<td>36, 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-based</td>
<td>84, 88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case-based</td>
<td>90, 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledge-based</td>
<td>70, 97</td>
</tr>
</tbody>
</table>

Table 2.2: Classification of the recommendation methods. The areas in which the contributions of this thesis fall are yellow-coloured.
in which the element $R_{uj}$ is the rating given by the user $u$ to the item $j$. Usually, $R_{uj}$ is a 5-point integer, 1 point means very bad, and 5 points means excellent. Let $P \in \mathbb{R}^{N \times D}$ and $Q \in \mathbb{R}^{M \times D}$ be latent user and item feature matrices, with vectors $P_u$ and $Q_j$ representing user-specific and item-specific latent feature vectors respectively ($N$ is the number of users, $M$ is the number of items, and $D$ is the number of item features).

In PMF, the ratings matrix $R$ is usually split into two matrices, one for training and the other for testing. The model performance is normally measured by computing the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) on the test set. In PMF, a probabilistic linear model with Gaussian observation noise is adopted. In this method, the conditional distribution over the observed ratings is defined in Eq. 2.1.

$$p(R|P, Q, \sigma^2) = \prod_{u=1}^{N} \prod_{j=1}^{M} [N(R_{uj}|\hat{R}_{uj}, \sigma^2)]^{I_{uj}}$$  \hspace{1cm} (2.1)

where $N(x|\mu, \sigma^2)$ is the probability density function of Gaussian distribution with mean $\mu$ and variance $\sigma^2$, $\hat{R}_{uj}$ denotes the estimated rating of item $j$ given by user $u$, and $I_{uj}$ is the indicator matrix that is equal to 1 if user $u$ has rated item $j$ and 0 otherwise.

In PMF, $R_{uj}$ is estimated by the inner product of latent user feature vector $P_u$ and latent item feature vector $Q_j$, that is $\hat{R}_{uj} = P_uQ_j^T$. The goal of matrix factorisation is to factorise a matrix into two matrices such that by multiplying the factorised matrices, the ratings matrix can be approximated, and the missing ratings can be obtained. This concept is graphically depicted in Fig. 2.1. According to the Eq. $\hat{R}_{uj} = P_uQ_j^T$, a user preference value over an item is a linear combination of the user-specific and item-specific latent feature values. This means that in PMF, it also is assumed that the user preferences follow the additive independence assumption [13]. Most of the state of the art preference learning methods assume a user’s preferences over one feature to be independent of their preferences for other features.

Also, the assumption is that the prior probabilities of the user and item matrices follow a Gaussian distribution with mean zero.

$$p(P|\sigma^2_P) = \prod_{u=1}^{N} [N(P_u|0, \sigma^2_P)]$$  \hspace{1cm} (2.2)

$$p(Q|\sigma^2_Q) = \prod_{j=1}^{M} [N(Q_j|0, \sigma^2_Q)]$$  \hspace{1cm} (2.3)

By using Bayesian inference, the posterior probability of the user and item matrices $P$ and $Q$ can be obtained by Eq. 2.4.
Chapter 2. Research Background

\[ p(P, Q|R, \sigma, \sigma_P, \sigma_Q) \simeq p(R|P, Q, \sigma)p(P|\sigma_P)p(Q|\sigma_Q) \] (2.4)

\[ p(P, Q|R, \sigma, \sigma_P, \sigma_Q) \simeq \prod_{u=1}^{N} \prod_{j=1}^{M} [N(R_{uj}|\bar{R}_{uj}, \sigma^2)]^{I_{uj}} \times \prod_{u=1}^{N} [N(P_u|0, \sigma^2_P)] \times \prod_{j=1}^{M} [N(Q_j|0, \sigma^2_Q)] \] (2.5)

The problem of modelling user preferences over items is now reduced to maximising the posterior probability given by Eq. 2.5. Maximising the probability in Eq. 2.5 is equivalent to minimising the log-posterior over the user and item latent feature matrices with ratings matrix and fixed parameters as shown in Eq. 2.6.

\[ \arg\min_{P, Q} \ln p(P, Q|R, \sigma, \sigma_P, \sigma_Q) = \ln p(R|P, Q, \sigma) + \ln p(P|\sigma_P) + \ln p(Q|\sigma_Q) + C \] (2.6)

where \( C \) is a constant that is not dependent on \( P \) and \( Q \). \( \sigma_P, \sigma_Q, \) and \( \sigma \) are standard deviations of matrix entries in \( P, Q, \) and \( R \) respectively. Minimising the log-posterior probability in Eq. 2.6 is equivalent to minimising the error function in Eq. 2.7.

\[ \arg\min_{P, Q} E = \frac{1}{2} \sum_{u=1}^{N} \sum_{j=1}^{M} I_{uj}(R_{uj} - \hat{R}_{uj})^2 + \frac{\lambda_P}{2} \sum_{u=1}^{N} \|P_u\|_{Frob}^2 + \frac{\lambda_Q}{2} \sum_{j=1}^{M} \|Q_j\|_{Frob}^2 = \{P^*, Q^*\} \] (2.7)

where \( \| . \|_{Frob} \) denotes the Frobenius norm, and \( \lambda_P = \frac{\sigma^2}{\sigma_P^2} \) and \( \lambda_Q = \frac{\sigma^2}{\sigma_Q^2} \), and \( P^* \) and \( Q^* \) denote the optimal values for model parameters \( P \) and \( Q \). Stochastic Gradient Descent and Alternating Least Squares are usually employed to solve the optimisation problem in Eq. 2.7. Using these methods, the accuracy of the method measured on the training set is improved using Eq. 2.7 and the optimal matrices \( P^* \) and \( Q^* \) are obtained.

Eq. 2.4 can be directly obtained by applying the chain rule of probability theory \[18\] to the Bayesian network in Fig. 2.1. Using the chain rule and the dependencies defined between the model parameters and hyper-parameters defined in Fig. 2.1 leads to Eqs. 2.8 and 2.9.

\[ p(P, Q, R, \sigma, \sigma_P, \sigma_Q) = p(P, Q|R, \sigma, \sigma_P, \sigma_Q)P(R, \sigma, \sigma_P, \sigma_Q) \] (2.8)

\[ p(P, Q, R, \sigma, \sigma_P, \sigma_Q) = p(R|P, Q, \sigma)p(P|\sigma_P)p(Q|\sigma_Q)p(\sigma)P(\sigma_P)P(\sigma_Q) \] (2.9)
In Eqs. 2.8 and 2.9 as we mentioned before, $\sigma_P$, $\sigma_Q$, and $\sigma$ are the model hyper-parameters and assumed to be zero, therefore they are known in advance. This means that their probabilities $P(\sigma_P)$, $P(\sigma_Q)$, and $P(\sigma)$ would be equal to one. $P(R, \sigma, \sigma_P, \sigma_Q)$ is also equal to one, since $R$, $\sigma$, $\sigma_P$, $\sigma_Q$ are assumed to be known, hence they are not variables. We also assume that the model parameters $P$ and $Q$ depend on the model hyper-parameters $\sigma$, $\sigma_P$, $\sigma_Q$ and the rating matrix $R$. Using these assumptions, Eq. 2.8 would be translated to Eq. 2.4.

Some of the other variants of matrix factorisation include Max-margin Matrix Factorisation (MMMF) [82], and Localised Matrix Factorisation (LMF) [119].

**Biased Matrix Factorisation (BiasedMF)**

This method [52] is analogous to PMF but also includes user bias and item bias values. The estimated ratings are obtained as $\hat{R}_{u,j} = \mu + bu_u + bi_j + P_u Q^T_j$ where $\mu$ is the global average rating, $bu_u$ is the user $u$’s bias, and $bi_j$ is the item $j$’s bias. The error function and the gradients are accordingly updated to incorporate the user and item bias vectors.

**Singular Value Decomposition (SVD)**

SVD is a well-established technique for identifying latent semantic factors in information retrieval. In SVD, the estimated ratings matrix is obtained using $\hat{R} = P.S.Q^T$, where $S$ is a diagonal matrix of size $D \times D$ having all singular values of matrix $R$ as its diagonal entries [80, 52]. Therefore, SVD can be considered a generalisation of PMF. In the area of recommender systems, the terms PMF and SVD are usually used interchangeably.
Chapter 2. Research Background

Singular Value Decomposition with Implicit Feedback (SVD++)

SVD++ [51, 49] extends BiasedMF by adding another set of item factors to account for the implicit feedback that the users provide over items. When a user rates an item, he/she implicitly tells us about his/her preferences by choosing to voice his opinion and vote a rating. Koren, Bell, and Volinsky [49] have found that incorporating this kind of implicit data significantly improves prediction accuracy.

Bayesian Probabilistic Matrix Factorisation (BPMF)

BPMF [78] extends basic PMF by assuming Gaussian-Wishart priors on the user and item hyper-parameters, the mean and standard deviation values for user and item matrices. In this method, the model is trained using Markov chain Monte Carlo methods and Gibbs sampling.

Non-Negative Matrix Factorisation (NMF)

NMF [54] is similar to basic PMF, with the limitation that all the values in the user and item matrices have to be positive. The model is trained by alternatively fixing one matrix and learning the other. A Bayesian extension of NMF (BNMF) has also recently been proposed by Hernando, Bobadilla, and Ortega [37].

2.1.2.2 Feature-aware preference modelling techniques

In the current methods based on latent factor models, the number of features that capture the user preferences is assumed to be known in advance as one of parameters of the model (e.g. [52, 12, 60, 66, 121]). For example, the current models assume that the users’ preferences over different movies can be explained by, say, 100 features, then train a model that explains the ratings using those 100 latent features. Intuitively, there are a limited number of features that explain most of the users’ preferences. Therefore, assuming too many item features in latent factor models would make the model learn many irrelevant features with less contribution to the ratings, and hence would degrade the accuracy.

The degrading effect of training latent factor models with too many latent features has been emphasised by some researchers in the literature [118, 43]. For example, in order to find the most attractive features for the users, Zhang et al. [118] conducted some analytical experiments and showed that around 15-45 features are sufficient to capture the users’ preferences in two datasets. Their experiments showed that incorporating more features in the proposed latent factor model would introduce noise into the recommendation
procedure. This observation was consistent with the recent work by McAuley and Leskovec [69]. Similarly, Jiang et al. [43] argued that if a small number of factors is used, the recommender system will not be able to distinguish between any users or items. Conversely, if the number of features is too large, users and items will be too unique for the system to calculate their similarities, and the complexity will considerably increase. Therefore, although the importance of feature-awareness has been emphasised and discussed in the literature, to the best of our knowledge, there is no solution for the degrading effect of irrelevant features in latent factor models.

2.1.2.3 Conditional preference modelling techniques

Although conditional preferences have been extensively researched in CP-Nets [12], GAI-Networks [27], and other preference representation models (e.g. [11], [13], [96], [94]), there are only a few recommender systems that take conditional preferences into consideration. Most of the preference models based on matrix factorisation, such as PMF [78], Bayesian Probabilistic Matrix Factorisation [79], and General Probabilistic Matrix Factorisation (GPMF) [81] use linear functions to represent user preferences. Therefore, in these models the dependencies between features are ignored.

Some of the latent factor models which incorporate conditional preferences are based on domain knowledge. For example, Yu et al. [107] proposed a quantitative model to capture conditional preferences in recommender systems. They assume that the conditional preferences can be defined as a set of conditional preference rules. For example, in travel domain a preference rule like "If Time = Summer and Destination = Okinawa, then Activity = Swimming, S=1" means that user likes a trip with swimming as activity if the trip is to Okinawa in summer. They also showed how such preferences can be translated into single rating scores on items (preference inference), and how the preferences over several users can be grouped into a single score (preference merging). The main disadvantage of the model proposed by Yu et al. is that it relies on domain knowledge as input to the model.

Liu et al. [61] on the other hand, proposed a latent factor model that does not require domain knowledge, and rather directly captures the conditional preferences from the user ratings. Using the movielens dataset, they showed that most of the users’ preferences in rating-based recommender systems are conditional. Then they proved that a quadratic polynomial can model the conditional preferences that cannot be captured by the linear function used in conventional latent factor models based on matrix factorisation. They
showed how to integrate the proposed quadratic approximation model of conditional preferences into ListPMF \cite{60} in order to obtain more accurate results.

2.1.2.4 Feature value preference modelling techniques

Zhang et al. \cite{118}, D’Addio and Manzato \cite{20}, and Wang, Pan, and Chen \cite{95} proposed models to capture the users’ preferences over feature values from user reviews. We divide the models in this category into two sub-categories, the memory-based and model based approaches.

Model-based feature value preference modelling techniques

Zhang et al. \cite{118} used phrase-level sentiment analysis on user reviews to extract explicit item features and user opinions (sentiments) and proposed EFM. The explicit preferences extracted were incorporated into basic PMF in order to improve the recommendation accuracy and explainability. The advantage of this method over other methods based on basic matrix factorisation is that it takes the preferences over feature values into account, therefore it improves the accuracy. However, its disadvantage is that it operates based on user reviews and in cases when this information is not available, this method cannot be applied. Also, it is based on phrase-level sentiment analysis, which ignores the context of the sentence in which a term is used.

Another work to model the user preferences over item feature values is proposed by Wang, Pan, and Chen \cite{95}. Similar to the previous models, they also use sentiment analysis on the user reviews. Therefore, the disadvantage of this model is that it needs the user reviews to extract the feature value preferences. While Zhang et al. aim at capturing the preferences over feature values for the general purpose of improving recommendation accuracy, Wang, Pan, and Chen mainly use the derived preferences over feature values to address the problem of cold-start users. In Wang, Pan, and Chen’s model, user preferences are derived in the form of a triple $<\text{attribute}, \text{opinion}, \text{specification}>$ as extracted from reviews, where opinion is the sentiment expressed by the user about the feature that is mapped to the attribute, and specification shows the attribute’s specification value. For example, $<\text{weight}, +1, 200g>$ denotes that the reviewer has expressed a positive opinion (+1) about the phone’s weight which is 200g. In some recommender systems, a new user is initially asked about their preferences over feature values by specifying their criteria for the variables. However, the preferences elicited are usually incomplete, because most users are not really able to state their full preferences (i.e. over all variables) due to cognitive
limitations and/or unfamiliarity with the product domain. Therefore, the challenge is how to complete the partial preferences, that is, to predict a user’s missing preferences for an unstated attribute. In Wang et al.’s model, the users’ similarities are used to complete the partial preferences of the new users. Unlike other models ([20, 118]) which partly rely on user reviews, their method relies entirely on the user reviews to model a user’s preferences.

Memory-based feature value preference modelling techniques

Another model that tries to capture the user preferences by considering feature values has recently been proposed by D’Addio and Manzato [20]. They also use sentiment analysis to extract the users’ sentiments towards specific item feature values from the unstructured reviews. Unlike Zhang et al., D’Addio and Manzato use sentence-level sentiment analysis as opposed to phrase-level sentiment analysis. D’Addio and Manzato’s memory-based model as an item-based model is easier to explain than latent factor models [118]. Another advantage of D’Addio and Manzato’s model is that it is based on sentence-level sentiment analysis so it considers the context of a word. However, unlike Zhang et al.’s model which extracts the preferences over feature values, this model only extracts the users’ overall sentiments towards each item feature. Therefore, the differences between different users in preferring an item feature value is not captured by their model, hence the accuracy could degrade by ignoring preferences over feature values.

2.1.2.5 Social preference modelling techniques

As mentioned before, the most popular recommender systems in practice are based on collaborative filtering, and most of the recommendation models that also take social information into account are based on collaborative filtering techniques. The existing social recommender systems take two sets of information as input. These inputs include the user-item ratings matrix, which includes the ratings given by users to the items, and social information, which includes how the users are connected to each other in a social network. Accordingly, these systems are comprised of a basic collaborative filtering component, and a social component. These systems can be classified into two broad categories according to their collaborative filtering component [88]. These categories are memory-based social recommender systems and model-based social recommender systems.
Model-based social preference modelling

Latent factor models based on matrix factorisation are widely employed in the social recommendation methods based on model-based approaches. The rationale behind these methods is that the users’ preferences are similar to or influenced by users whom they are socially connected to. Intuitively, the users with strong relationships are more likely to share similar preferences than those with weak ties. Therefore, the users in social networks are treated separately, based on the strength of their relationships. In general, the social recommendation methods minimise the error using Eq. 2.10.

\[
\arg\min_{\zeta, \Omega}[E = L^E(R, \zeta) + \alpha S^E(T, \Omega)]
\]

where \(\zeta\) denotes the set of parameters that define user preferences in the latent factor models, \(R\) denotes the user-item ratings matrix, and \(T\) denotes the social network matrix which shows the user-user social relations, and \(\Omega\) denotes the set of parameters learnt from the social information, and \(L^E\) and \(S^E\) are the model error for the latent and social aspects respectively. According to the definition of \(S(T, \Omega)\), the methods in this categories are divided into four sub-categories: Co-factorisation methods, Ensemble methods, Regularisation methods, and Probabilistic methods. In turn, we elaborate on the models in each one of these categories.

Co-factorisation methods. The underlying assumption in these systems is that a user should share the same preference vector in the rating space and the social space. A shared preference vector can explain both the user preferences in the rating space and the user relationships in the social space. Therefore, these models perform a co-factorisation in the user-item ratings matrix and the user-user social relation matrix by sharing the same user preference latent matrix. In the category of co-factorisation social recommenders, Ma et al. [65], Tang et al. [87], Guo et al. [31], Jiang et al. [43], and Yang et al. [105] propose methods to capture the effect of social relationships in user preferences.

Ma et al. incorporated the social network information into the basic matrix factorisation and proposed SoRec. In SoRec, it is assumed that the trust relations between users in the social network can be factorised into a user-specific latent factor matrix \(P\) and another user-specific latent factor matrix \(Z\). Furthermore, they assumed that the user ratings in the user-item ratings matrix can also be explained by the user-specific latent factor matrix \(P\), and the item-specific latent factor matrix \(Q\). They formulated an optimisation problem, in which the difference between the estimated trust and rating values using the matrices
Chapter 2. Research Background

$P$, $Q$, and $Z$, and the real trust and rating values in the matrices $R$ and $T$ is minimised.

Similarly, in the model proposed by Tang et al. (LOCALBAL), the user preferences defined by the matrix $P$ are derived from a co-factorisation on the user-item ratings and user-user social matrices. However, LOCALBAL is different from SoRec in the sense that the preferences of correlated users are correlated via a correlation matrix.

Guo et al. proposed a model called TrustSVD, in order to capture the socially-influenced user preferences. TrustSVD is similar to SoRec in the sense that the social relations matrix ($T$) is factorised into two latent user-specific matrices. However, it is different from SoRec in the sense that implicit feedback from user ratings and trust relations is also considered in calculating the missing user ratings.

Jiang et al. also proposed a method called ContextMF, to capture the user preferences given the considerations of social influence. In ContextMF, the user preferences are assumed to be explainable by individual preference as well as interpersonal influence. More specifically, the user-item ratings matrix is factorised into two intermediary matrices: user-item influence matrix, which shows how much each user is influenced by his/her social connections to adopt a particular item, and user-item preference matrix, which shows the user’s individual preference when adopting an item.

TrustMF, which was proposed by Yang et al. is very similar to SoRec. The only difference is that in TrustMF, the vectors are penalised based on the number of ratings given by a user, the number of ratings given to an item, and the number of trusted users by a user. However, in SoRec, the values of all vectors are equally penalised. Furthermore, unlike SoReg, in TrustMF different models are used for the truster and trustee. The truster model shows how a user’s preferences are affected by the users in his social network, whereas the trustee model shows how a user’s opinions affect the preferences of other users who trust that user.

**Ensemble methods.** The underlying assumption in these systems is that a user should have similar ratings to those of their friends, and the user rating for a missing item can be obtained by a linear combination of the user’s preference model, as well as the preference model of her friends in the social network. Formally speaking, in these models the user ratings are obtained according to Eq. 2.11

$$\hat{R}_{uj} = L_{uj}(R, \zeta) + \beta S_{uj}(T, \Omega)$$

(2.11)

where $R_{uj}$ is the rating given by user $u$ to item $j$, $L_{uj}$ is the rating produced by the preference model of the user, and the $S_{uj}$ is the rating produced by the preferences of
the users connected with the user in the social network, and $\hat{R}_{uj}$ is the estimated rating of user $u$ on item $j$. In Eq. 2.11, $\beta$ controls the contribution of social network information to the user preferences. Ma et al. 64 and Tang, Gao, and Liu 85 propose methods that fall into the category of Ensemble models.

In the model proposed by Ma et al., which is referred to as Social Trust Ensemble (STE), the value of $SM_{uj}$ is calculated as a weighted sum of the predicted ratings from the user’s social network of friends/trustees/followers. In STE, the predicted ratings of the friends in the user’s social network are obtained by the model that is trained for them by matrix factorisation.

Similarly, the model proposed by Tang et al., which is referred to as mTrust, also uses the user’s social network in predicting the rating of an item. However, it is different in the sense that the value of $SM_{uj}$ is calculated based on the ratings of the user’s friends, as well as the strength of influence from friends which is learnt automatically from the data.

Regularisation methods. Regularisation models assume that a user’s preference model is similar to that of their social friends. These models are similar to co-factorisation models in two ways. In both categories, the preference model of the user’s friends do not contribute in the calculation of the predicted ratings. Therefore, the value of $\beta$ in Eq. 2.11 in the models of both categories is fixed at zero. However, unlike co-factorisation models which assume that the user preference vector is shared in the rating and social space, the regularisation models merely try to minimise the difference between the preference vector of the user and the preference vector of their friends.

This category includes the models proposed by Jamali and Ester 42, Ma et al. 66, Sun et al. 83, and Liu and Aberer 32, Zhao et al. 122, Fazeli et al. 23, and Zhao et al 121, Qian et al. 76, Zhao, Qian, and Feng 120.

In Jamali and Ester’s model, which is abbreviated to SocialMF, a social regularisation term is added in the prediction error function, so that a user’s preference is forced to approach the average preference of the user’s social network. The advantage of SocialMF is that it also addresses the transitivity of trust, or trust propagation in a social network. In reality, the user is directly connected to some users, who in turn are connected to other users, who are connected to other users and so on. Therefore, the user is actually influenced by the entire social network, not just the users with whom she is directly connected. The social regularisation term forces the user’s preferences to approach her friends’ preferences, which are forced to approach their friends’ preferences. Therefore, by adding social regularisation, the propagation of trust in social networks is also taken into consideration.
Another model in this category is the method proposed by Ma et al., which is referred to as SoReg. Similar to SocialMF, in SoReg, the user’s preference model is assumed to be similar to the preference model of his social contacts. However, unlike SocialMF, in SoReg the user’s preference model is made similar to the preference model of each one of his social friends. The degree to which the user’s preference vector should become close to that of his friends is controlled by the similarity based on the previous ratings.

Sun et al. proposed a social regularisation method called RSBOSN. This method builds on SoReg. However, it is different from SoReg in three ways. In RSBOSN, it is assumed that the user indicates his preferences by giving tags to the items, and multiple tags can be given to one item to describe the user preference for that item.

A clustering approach is used to find similar users in the social network, and the similarities between the users are calculated based on the clusters they belong to, and finally, the users and items are mapped to the tag space, and the similarity between users and items are calculated.

Then the user’s preference model is made similar to that of his friends, based on two similarities: the similarity between the user and his friends from the clustering, and the similarity between the user’s friend and an item for which the rating is to be predicted. Therefore, the influence that a user gets from other users not only depends on his similarity to those users, but also the similarity of his friends to the item that is being considered. The rationale for this is probably that the users who have more similarity with an item, would be more willing to recommend it to others, and consequently they would have more influence on their friends.

Another regularisation model to capture socially-influenced preferences is SoCo, proposed by Liu and Aberer [62]. Similar to SoReg, a user’s preferences are forced to become similar to those of her social friends. The extent to which the preferences become similar also depends on the user similarities calculated using the rating history. In SoReg and SoCo, if the rating similarities for two users are high, their preference models become more similar. However SoCo is different from SoReg in the sense that the contextual information (e.g. time of the ratings) are also taken into consideration.

Qian et al. [76] fused four factors, personal interest, individual preference, interpersonal similarity, and interpersonal influence into a unified latent factor model based on PMF, and proposed Personalised Recommendation Model (PRM). The items are classified into categories and sub-categories, and then based on the number of items that the user has rated in each category, the user’s interest in each category is estimated. Then the general
interest of a user in the category of an item is calculated as a matrix. This model is similar
to ContextMF in the sense that both factors of social influence and social similarity between
users are taken into consideration. However, it is different from ContextMF in two ways.
First, PRM is a co-factorisation model, while ContextMF is a regularisation model. Second,
PRM also takes the personal interests into consideration, while ContextMF only considers
individual preference indicated by the ratings. The model proposed by Zhao, Qian, and
Feng [120] is exactly the same as PRM except that it does not include the personal interest
factor.

The advantage of the models in this category is that the incorporation of social regu-
larisation term enables these models to indirectly consider the trust propagation in social
networks.

**Probabilistic methods.** There is another category of methods that use probabilis-
tic approaches to model socially-influenced user preferences. Some of the models in this
category are proposed by Liu, Wu, and Liu [59], He and Chu [34], Zhao, McAuley, and
King [123], Jun and Fekri [125].

Liu, Wu, and Liu [59] proposed a trust-aware and content-aware probabilistic method,
abbreviated as BPMFSRIC. BPMFSRIC modifies BPMF and fuses item contents and
social relations with the user ratings into a unified model. While all methods reviewed so
far build on PMF, this method is based on BPMF. One of the problems with the methods
based on PMF is that if the regularisation parameters are not tuned carefully, the model
is prone to over-fitting because it finds a single point estimate of the parameters [78].
Therefore, in basic matrix factorisation, it is assumed that the regularisation parameters
are known in advance, and they are fed into the model as inputs. In order to address
this problem, Salakhutdinov and Mnih [78] proposed an extension to PMF by assuming
Gaussian-Wishart priors on the user and item regularisation parameters. In BPMF, these
parameters are also learnt along with other model parameters. Liu, Wu, and Liu argue that
the posterior distribution of user hyper-parameters should be conditioned on the preference
vectors of the trusted users. Similarly, the posterior distribution of user hyper-parameters
are dependent on the feature vectors of linked items. Unlike BPMF in which uniform
hyper-parameters are assumed for different users and items, in BPMFSRIC, different hyper-
parameters are used for each item and user.

Another method that considers social information is SNRS proposed by He and Chu.
They assume that the user rating for an item is a result of his individual preferences, as well
as his knowledge about that item. They acquire this knowledge either through public media
such as newspapers, TV, and Internet, or through the recommendations they get from their social friends. Therefore, three factors affect user ratings: user preferences, general acceptance of items, and influence from social friends. He and Chu used the Bayesian theorem and proposed a probabilistic model to make personalised recommendations from such information. In SNRS, the probability that the user $u$ gives the rating $k$ to the item $i$ is conditional on the item features, user features, and the ratings given by user $u$’s immediate friends to the item $i$. In SNRS, the set of users correlated to a given user is simply obtained by considering their circle of friends in the social network. Then this probability is defined as the multiplication of three probabilities. The probability that the user $u$ gives the rating $k$ to the item $i$, given that she has the features (user preference), the probability that the user $u$ gives the rating $k$ to the item $i$, given that the item $i$ has the features (item acceptance), and the probability that the user $u$ gives the rating $k$ to the item $i$, given the ratings that her immediate friends have given to that item (social influence). Then using Bayesian theorem, each one of these probabilities are estimated. SNRS also incorporates the influences that the users get from distant friends. Therefore, similar to Regularisation methods, the propagation of social influence in the network is also considered.

**Memory-based social preference modelling**

The models in this category are usually user-based, and follow two basic steps. First, they obtain the related users for a given user, and then as in traditional user-based collaborative filtering, they aggregate the ratings from the related users and estimate the missing ratings for the user [88]. Victor et al. [91] Victor, De Cock, and Cornelis [92], Golbeck [26], Massa and Avesani [68], and Jamali and Ester [41] incorporate the social network information into a traditional user-based approach.

In the methods proposed by Victor et al. and Victor, De Cock, and Cornelis, the set of correlated users of a user is simply assumed to be the user’s directly related users.

Golbeck proposed a method TidalTrust to estimate the trust scores between the users that are not directly connected in the social network. Based on the estimated trust values, the set of correlated users for each user is obtained and used in a traditional user-based collaborative filtering approach. Similarly, Massa and Avesani proposed MoleTrust to estimate the trust between related and unrelated users in the social network and use the obtained social information in estimating user ratings.

In the method proposed by Jamali and Ester, which is referred to as TrustWalker,
Chapter 2. Research Background

the rating given by user \( u \) to item \( i \) is estimated by querying user \( u \)'s direct and indirect friends’ ratings for item \( i \), as well as similar items by performing a random walk in the social network. The Random Walk model employed in this method enables the exploitation of item-oriented and trust-based user-oriented approaches. Suppose that the goal is to estimate the rating of user \( u \) on item \( i \). At each step of this process, the random walk first starts from the node representing the user \( u \) in the social network. At each current node \( v \), if the user representing that node has rated the item \( i \), it stops the walk and returns the rating \( R_{vi} \). Otherwise, it does one of the followings: 1) it stops at the current node \( v \) and randomly selects one of the items \( j \) similar to the item \( i \) rated by the user \( v \) and returns \( R_{vj} \). 2) it moves to a node connected to the current node in the social network. Therefore, in this method, the propagation of trust in social network is also implicitly taken into consideration.

All the latent factor models introduced in Section 2.1.2.5, which consider the influence of social friends in user preferences, define the user preferences as a vector of weight values for item features. Therefore, the differences between users in preferring item feature values is totally ignored.

2.1.2.6 Dynamic preference modelling techniques

The time-dependent collaborative filtering models are also classified into memory-based time-aware recommenders and model-based time-aware recommenders [102].

Model-based time-aware recommenders

The models in this category usually fall into four classes: 1) models based on PMF, 2) models based on BPMF, and 3) models based on Probabilistic Tensor Factorisation (PTF), and 4) models based on Bayesian Probabilistic Tensor Factorisation (BPTF).

Models based on probabilistic matrix factorisation. Modelling the drifting preferences using a model-based approach based on PMF has first been considered by Koren [50] in TimeSVD++. TimeSVD++ builds on the previous model SVD++ [52], in which the user preferences are modelled through a latent factor model that incorporates the user bias, item bias, and also the implicit feedback given by the users. For each one of these preference components, Koren [50] used a time-dependent factor to capture both transient and long-term shifts. They showed TrustSVD++ achieves significant improvements over SVD++ on a daily granularity [102].
Another model-based time-aware recommendation model was proposed by Koenigstein, Dror and Koren [47]. In this model, the authors used session factors to model specific user behaviour in music learning sessions. Unlike TimeSVD++ which is domain-independent, this method was developed especially for the music domain. First, it enhances the bias values in SVD++, by letting the item biases share components for items linked by the taxonomy. For example, the tracks in a good album may all be rated higher than the average, or a popular artist may receive higher ratings than the average for items. Therefore, shared bias parameters are added to different items with a common ancestor in the taxonomy hierarchy of the items. Similarly, the users may also tend to rate artists or genres higher than songs. Therefore, the user bias is also enhanced by adding the type of the items. It is also assumed that unlike in the movies domain, in music it is common for the users to listen to many songs, and rate them consecutively. Such ratings might be rated similarly due to many psychological phenomena. The advantage of the models proposed by Koenigstein, Dror and Koren [47] and Koren [50] that extend SVD++ is that they enable the capturing of dynamicity of the preference aspects with a high granularity for aspects that are assumed to be more subject to temporal drift. Furthermore, as shown by Koenigstein, Dror and Koren [47], domain-dependent temporal properties of the preferences and their individual aspects can also be taken into consideration.

Jahrer, Toscher and Legenstein [40] split the rating matrix into several matrices, called bins, based on their time stamps. For each bin, a separate time-unaware model is trained by producing an estimated rating value that is obtained using the ratings given for that bin. Each one of the bins is assigned a weight value, and the final rating is obtained by combining the ratings that are obtained through the models trained on each bin. Therefore, using this approach, they combined multiple time-unaware models into a single time-aware model. The disadvantage of this model is that the ratings matrix is usually sparse as it is, and it even becomes sparser, when the ratings are split into bins, since each model that is trained for a bin receives even less training data.

A similar approach is followed in the model proposed by Liu and Aberer [62]. They systematically integrated contextual information and social network information into a matrix factorisation model to improve the recommendations. To overcome the sparsity problem of training separate models based on their time-stamps, they applied a random decision trees algorithm, and created a hierarchy of the time-stamps. For example, the ratings can be split based on year on the first level, month on the second level, day on the third level, and so on. Liu and Aberer argue that the ratings that are given at similar time
intervals are better correlated with each other, and therefore such clustering is justified. They also added the influence of friends to the model, using a context-aware similarity function. In this function, users who give similar ratings to those of their friends in similar contexts get higher similarity values. Consequently, in this model, the role of time on the social influence is also indirectly taken into consideration.

Baltrunas, Ludwig and Ricci [6] argued that methods based on tensor factorisation can improve the accuracy when the datasets are large. Tensor factorisation requires the addition of a large number of model parameters that must be learned. When the datasets are small, simpler models with fewer parameters can perform equally well or better. In Baltrunas, Ludwig and Ricci’s method, a matrix is added to capture the influence of contextual factors (e.g. time) on the user preferences by modelling the interaction of contextual conditions with the items. Although the model is quite simple and fast, it does not include the effect of time on individual preference aspects. Unlike the models proposed by Koenigstein, Dror and Koren [47] and Koren [50], it cannot capture fine-grained and domain-specific dynamicities.

Models based on Bayesian probabilistic matrix factorisation. As mentioned before, BPMF extends the basic matrix factorisation [78] by assuming Gaussian-Wishart priors on the user and item regularisation parameters and letting the hyper-parameters be trained along with the model parameters. Dynamic BPMF (dBPMF) is a non-parametric Bayesian dynamic relational data modelling approach based on the Bayesian probabilistic matrix [63]. This model imposes a dynamic hierarchical Dirichlet process (dHDP) prior over the space of probabilistic matrix factorisation models to capture the time-evolving statistical properties of modelled sequential relational datasets. The dHDP was developed to model the time-evolving statistical properties of sequential datasets, by linking the statistical properties of data collected at consecutive time points via a random parameter that controls their probabilistic similarity.

Models based on probabilistic tensor factorisation. In tensor factorisation methods, the context variables are modelled in the same way as the users and items are modelled in matrix factorisation techniques, by considering the interaction between users, items, and context. In tensor factorisation methods, the three-dimensional user-item-context ratings are factorised into three matrices, a user-specific matrix, an item-specific matrix, and a context-specific matrix. A model in this category was proposed by Karatzoglou et al. [45], who used Tensor Factorisation with CP-decomposition, and proposed
multi-verse recommendation, which combines the data pertaining to different contexts into a unified model. Therefore, similar to the model proposed by Baltrunas, Ludwig and Ricci [6], other contextual information besides time (e.g. user mode, companionship) can also be taken into consideration. However, unlike Baltrunas, Ludwig and Ricci [6], they factorise the rating tensor into four matrices, a user-specific matrix, an item-specific matrix, a context-specific matrix, and a central tensor, which captures the interactions between each user, item, and context value. Then the original ratings tensor, which includes the ratings given by users to items in different contexts (e.g. different times) can be reconstructed by combining the four matrices back into the ratings tensor. Other models in this category are the models proposed by Li et al. [57] and Pan et al. [74].

Models based on Bayesian probabilistic tensor factorisation. There is a class of dynamic models that are based on Bayesian Probabilistic Tensor Factorisation (BPTF) [103]. BPTF generalises BPMF by adding tensors to the matrix factorisation process. A tensor extends the two dimensions of the matrix factorisation model to three or more dimensions. Therefore, besides capturing the user-specific and item-specific latent matrices, this model also trains a time-specific latent matrix, which captures the latent feature values in different time periods. The models based on tensor factorisation are similar in introduction of the time-specific matrices into the factorisation process. However, they are different in the way they factorise the ratings matrix into the user, item, and time matrices, and also the way they train the factorised matrices. Similar to BPMF, BPTF uses Markov Chain Monte Carlo with Gibbs sampling to train the factorised matrices.

2.1.2.7 Memory-based time-aware recommenders

Some simple time-dependent collaborative filtering models have been proposed by Lee, Park and Park [55]. The models use item-based and user-based collaborative filtering, and exploit a pseudo-rating matrix, instead of the real rating matrix. In the pseudo-rating matrix the entries are obtained using a rating function, which is defined as the rating value when an item with launch time $l_j$ was purchased at time $p_j$. This function was inspired by two observations, 1) more recent purchases better reflected a user’s current preferences, and 2) recently launched items appealed more to the users. If the users are more sensitive to the launch time of the items, the function gives more weight to new items, and if the user’s purchase time is more important in estimating their current preference, the function assigns more weight to recent purchases. After obtaining the pseudo-rating matrix, the neighbours are obtained as in the traditional item-based or user-based approaches, and the
items are recommended to the users.

2.2 Explaining user preferences

There has been increasing awareness in recommender systems research of the need to make transparent recommendations to the users [24]. However, explainability has received less attention than improving the recommendation accuracy. In this section, we review studies on explaining the modelled preferences to the users. The explanation systems are broadly classified into four categories: Collaborative filtering based systems, content-based systems, case-based systems, and knowledge-based and utility-based systems.

2.2.1 Collaborative filtering explanations

The explanations in collaborative filtering can be extracted from traditional models (memory-based) or latent factor models (model-based).

2.2.1.1 Traditional model explanations

The explanations in this category are not based on latent factor models. These models can either be item-based or user-based.

Item-based models

In item-based recommendations, items similar to the item that is being recommended are used to explain the recommendation of an item. For example, "users who bought this item, also bought ..." which is employed by Amazon.com [80] and system proposed by Bilgic and Mooney [9] are examples of item-based explanation [89].

User-based models

In user-based explanations, similar users are retrieved based on the ratings and used to explain a recommendation. Demonstrating the distribution of the ratings of that item by the neighbours of a user is an example of user-based explanation style [36]. As in the classic user-based collaborative filtering, the user’s neighbours are defined based on the ratings given by users to the items. The demographic approach is similar to user-based collaborative filtering, in the sense that it also calculates a user’s similarity with other users [1]. However, instead of ratings, the user demographics data is used to calculate the user similarities.
Chapter 2. Research Background

2.2.1.2 Latent model explanations

The models in this category are classified into rating-based and review-based. Rating-based explanations use the ratings to generate explanations, while review-based explanation systems also use reviews to generate more comprehensive explanations.

Rating-based models

Brun et al. [14] proposed an approach for explaining recommendations resulting from matrix factorisation, by interpreting the item features in the feature matrix as representative users. Such a mapping is claimed to be useful in explaining the recommendations to users as well as alleviating the new item cold-start problem without the need for any information about the content of the items. There are two disadvantages with their approach. This method requires the representative users to provide ratings for new items. Also it is not clear how mapping the item features to the representative users helps improve the explainability of the recommendations. The principle of the method is to assume that if an item scores highly on feature $x_1$, $x_2$, and $x_3$ which are preferred by representative users $u_1$, $u_2$, and $u_3$, and a fourth user’s preferences are close to these, the item is also preferred by the fourth user. The major advantage of this method is helping alleviate cold-start problem with new items, assuming the representative users are asked to rate the new items if they have not rated them already.

Unlike Brun et al., Aleksandrova et al. [4] map the features by calculating the Euclidean distance between the feature vector from user feature matrix, and the corresponding canonical vector to the features. The experiments on this method show that the method discovers interpretable features and does not significantly decrease the accuracy on tests with 10 and 15 latent factors. However, it does not provide a mechanism to explain the latent features.

Review-based models

Zhang et al. [118] proposed the Explicit Factor Model (EFM) to generate explainable recommendations, while keeping a high prediction accuracy. In this paper, the explicit product features are extracted with feature extraction techniques in natural language processing, and then the users’ opinions about those features are extracted using phrase-level sentiment analysis on user reviews. This system states the reasons why an item is recommended, as well as reasons for not recommending an item to a user. Compared to the methods proposed by Aleksandrova et al., and Brun et al., this method provides more understandable explanations to the users based on actual feature interpretations. However,
Chapter 2. Research Background

39

the explanations are mainly focused on the latent features, and preference aspects are not modelled and explained.

Gedikli et al. [24] address the question of how explanations can be communicated to the users in a more effective way. The users are provided with different types of explanations, and the effects of explanations on different dimensions are measured. The explanations are illustrated using existing visualisations, as well as two novel interfaces based on tag clouds. The dimensions used to evaluate the explanations are transparency, efficiency, effectiveness, and satisfaction. The experiments in Gedikli’s research illustrate the benefits of explanations in recommender systems. However, the main focus of their study was designing effective explanation interfaces and thus improving the quality dimensions, and they do not investigate how the explanations can be extracted from complex models in the first place.

Chen and Wang [17] proposed a novel explanation interface, emphasising the explanation of the trade-off properties within a set of recommendations in terms of both the static specifications and the feature sentiments extracted from product reviews. This model is novel in the sense that in addition to static specifications, it also extracts feature sentiments from product reviews to produce tradeoff-oriented explanations. In this approach, a simple linear function captures the user’s sentiments towards item features in addition to the static specifications of the items. First, the top k ranked items are calculated using the static specifications and the sentiments extracted. If the user cannot make a choice among the recommendations provided, he is engaged in an interactive process with the system, in which his preferences are refined. The process finishes when the user makes the final choice. They found that the explanation interface helps improve the users’ knowledge about the products, preference certainty, perceived information usefulness, recommendation transparency and quality, and purchase intention.

2.2.2 Content-based explanations

In content-based explanations, a user’s ratings on all items are given to the system as input. These ratings are used to train a classifier which is used for recommendation and explanation. The explanations are produced based on the item properties. For example, by considering the item properties for a user in the movies domain, the system detects that Harrison Ford is the user’s favourite actor. Therefore, a movie item starring that particular actor is recommended to the user, and this reason is given as the explanation [84, 93, 89].
2.2.3 Case-based explanations

Case-based reasoning explanations are similar to content-based explanations, but focus primarily on similar items that are used to make the recommendation. The explanations generated by the Find Me system proposed by Viappiani et al. \cite{90} and the explanations provided by Netflix.com \cite{8} are examples of case-based style \cite{89}.

2.2.4 Knowledge and utility-based explanations

These systems use a knowledge base and solve problems using rules in an inference engine. Case-based reasoning is a particular type of knowledge-based systems, in which the examples of similar cases are used to produce a solution. In these systems, the match between a user and an item is inferred using the inference engine \cite{70, 97, 89}.

2.3 Collaborative filtering based on deep learning

Recently there has been a line of research on employing deep learning methods in recommender systems in conjunction with latent factors models. Some of the papers reviewed completely replaced matrix factorisation with a deep learning module. Others either have combined the two, or have extracted latent user/item representations using neural networks. These methods mostly strive to improve the recommendation accuracy by exploiting the additional information about users (e.g. user demographics) and items (e.g. item descriptions) or user-item interactions (e.g. user reviews on items), or capturing non-linear user-item interactions using deep neural network structures. In particular efficiency and explainability are two major challenges with the deep learning methods, and integration of deep networks with matrix factorisation seem to be a good way of improving accuracy while achieving explainability. In the following, we review some of the papers in the area of collaborative filtering using deep learning.

One of the methods based on deep learning for collaborative filtering was proposed by Dong et al. \cite{22}. They propose to use deep representation learning for the additional information and collaborative filtering for the ratings matrix. They use an extended version of stacked denoising auto-encoder (SDAE), which is a specific form of neural network. The network is comprised of one encoder component and one decoder component. The encoder takes a given input and maps it to a hidden representation, while the decoder maps the hidden representation back to the reconstructed input. Then the learning tries to minimise the error in reconstructing the input. Using this mechanism, this method is able to map the
additional data about the users and items, into latent feature representations. A similar method was proposed by Zhang et al. [117] called AutoSVD++. They also feed additional data about users and items to a deep network and extract the hidden representation of the features. However, instead of SDAE networks, they use contractive auto-encoders (CAE) for the deep learning module. While Dong et al. fuse the hidden feature representations into the prediction error function, Zhang et al. only use the feature representations in the rating estimation function. The experimental results show improvements over SVD++, into which AutoSVD++ is integrated.

The additional information about users and items are given to two SDAE components, and latent features are extracted. Then these features are integrated into a latent factor model based on PMF. Therefore, the latent features in PMF are trained simultaneously, using both user ratings and the features that are trained using deep learning modules. The experiments show that learning the latent features by the deep learning components improves the accuracy of PMF. Although Dong et al.’s method shows improvements in accuracy, it only trains preferences over features and ignores the other aspects introduced in Chapter 1. Furthermore, since explaining the recommendations to the users is still difficult.

Another method based on deep learning for recommendations is proposed by Xue et al. [104]. Similar to Dong et al., Xue et al. also propose to use deep learning for training latent features and introduce Deep Matrix Factorisation (DMF). The architecture of DMF is illustrated in Figure 2.2. DMF is different from Dong et al.’s method in several ways. Unlike Dong et al., Xue et al. do not use additional information to train the latent features. In Dong et al.’s method, the users and items latent matrices are combined linearly to calculate the estimated ratings. Therefore it does not account for the conditional dependencies that might exist between features.

Xue et al. train the latent user and item feature vectors using a deep network. This network receives vectors for user $u$ and item $j$ from the ratings matrix as input, and produces the user-specific and item-specific latent feature vectors for user $u$ and item $j$. The estimated rating is then calculated by finding the similarity between the user and item feature vectors. Unlike Dong et al. in which gradient descent is used to train the latent features, in DMF the latent features are trained using the deep network. DMF also exploits the implicit feedback to obtain higher accuracies. If a rating by a user for an item is not known in the ratings matrix, it is assumed to be zero. The ratings matrix is then fed to the deep network for training. The experiments also reveal accuracy improvements
over some of the state of the art classic and deep learning methods.

Similar to Dong et al.’s method, DMF also does not consider different preference aspects. Furthermore, calculating the similarity between latent feature vectors still does not address the dependencies that might exist between features. These two methods are similar to the older method, Wide and Deep [18] proposed by Chen et al. for app recommendations. In Wide & Deep, user data and app impression data are fed into a neural network that includes bode wide and deep paradigms. The output of probability that a user would like an app. They show that a neural network that is both wide and deep outperforms a network that is wide or deep. Stochastic back propagation is then used to train the neural network. This method also suffers from recommendation explainability. The neural network receives all the data that is available and produces the estimated ratings like a black box.

Wei et al. [99] propose a deep CF method called Integrated Recommendation Collaborative Deep (IRCD). In IRCD the item features are learnt from a deep learning architecture using the descriptions of items that are retrieved online. The features are then integrated
into a latent factor model to improve the accuracy. The item descriptions are first parsed using natural language processing, by removing stop words, applying term frequency inverse document frequency (tf-idf), and normal vectors are obtained. Then these normalised vectors are fed into the deep network and hidden representations of the item features are trained. The latent features are then fused into a latent factor model based on matrix factorisation using a regularisation term. In the regularisation term, the error between the latent features trained by latent factor models and the latent representations learnt by the deep network is minimised. Similar to Dong et al.’s method, IRCD also uses SDAE networks for extracting item latent feature vectors. However, unlike Dong et al. who use both items and users data, they only use items data for the deep learning component.

Although the recent work has exploited neural deep networks for recommendations, most of them use the inner product of item features and user features to estimate the ratings. To resolve this issue, He et al. have proposed a neural architecture called (Neural Collaborative Filtering) NCF, in which arbitrary interactions between user features and item features can be learnt. In NCF, the user-item interactions are learnt using multi-layer perceptrons. Similar to DMF, NCF also uses deep networks to model the user and item features and estimate ratings. Both methods use both explicit and implicit feedback data to train the network. However, unlike DMF in which the similarity of user and item latent features learnt by the neural network is used to estimate the ratings, NCF does not separate the user and item features. It uses multi-layer perceptrons in the neural network to train the arbitrary interactions and produce the final rating as output. Similar to the other neural learning methods reviewed so far, this method also ignores different preference aspects. It is not clear what aspects contribute to a rating, since they are all automatically trained by the neural deep networks.

Zheng, Noroozi, and Yu proposed a method called (Cooperative Neural Networks) CoNN for modelling users and items using review texts. In CoNN, two coupled convolutional neural networks work together to maximise the prediction accuracy. One of the networks receives the user’s review texts as input, and models the user’s behaviour as a hidden latent feature vector. The other network translates the item’s review texts into a feature vector that describes item properties. The learnt user and item latent features are then used to predict the rating for the corresponding user and item in a layer that resides on top of the two neural networks. In order to translate the natural language reviews into numbers that can be fed into the neural networks, pre-trained word embeddings from Google are used. The architecture of CoNN is shown in Figure 2.3. Similar to DMF, CoNN
also diffuses user and item latent representations using deep learning networks. However, unlike DMF which receives ratings matrix as input, CoNN receives review texts.

![Figure 2.3: Architecture of CoNN](image)

Recently, Deng et al. [21] have proposed a trust-aware model based on deep learning for recommendations. They argue that the initialisation of the latent feature matrices can greatly influence the performance of these models. A good initialisation can lead to a better local minimum and improve the learning process. Then they suggest that we can utilize a deep learning method, i.e. autoencoder, to reduce the dimensionality of the latent feature vectors. This would make it easier to deal with the initialisation problem for the latent feature matrices. To overcome this challenge, they propose a matrix factorisation based approach for trust-aware recommendations called DLMF. The deep network is used to pretrain the initial values of the parameters of the matrix factorisation model.
2.4 Summary

The state of the art models are broadly classified into individual intelligence and collective intelligence methods. Individual intelligence methods (e.g. CP-Nets) model a user’s preferences using the feedback data about that individual user. Collective intelligence or collaborative filtering methods on the other hand use the feedback data from many other users to collaboratively model the users’ preferences, and exploit the existing correlations between users’ preferences.

Among collaborative filtering methods, latent factor models have increased in popularity in academia and industry, owing to their simplicity, efficiency, scalability and accuracy. These methods try to capture different aspects in preferences using latent factors. Different aspects in preferences include feature preferences, feature value preferences, conditional dependencies between features, social influence, time, and bias. All the latent factor methods model preferences over features as the most basic preference aspect that can be captured from the user ratings data. Moreover, various extensions to the latent factor models are also proposed to capture other preference aspects besides feature preferences. These methods usually capture only one of two other aspects, and ignore the others. Some of the existing work on the other hand has focused on explaining the resulting recommendations to the users. These methods are broadly classified into collaborative filtering, content-based, case-based, and knowledge-based explanation systems, depending on the underlying model that is used to generate recommendations.

Another research area that has become increasingly popular recently in the field is collaborative filtering systems that apply deep learning. Although these methods are promising and show improved accuracy, they are usually inefficient and difficult to explain. Some of existing work tries to make a tradeoff between these three objectives by fusing deep networks into latent factor models, or at least structuring the deep networks in a way that user-specific and item-specific latent factors can be explicitly extracted by the network. Therefore, we speculate that in the future, the accuracy and explainability of latent factor models will probably be complemented by the additional accuracy obtained through deep networks.
Chapter 3

Feature-Aware Modelling of Preferences

In the current methods based on latent factor models, the number of features that capture the user preferences is assumed to be known in advance as one of parameters of the model (e.g. 52, 12, 60). For example in these methods, it is assumed that the users’ preferences over different movies can be explained by, say, 100 features, which are used to train a model. Since there are a limited number of features that explain most of the users’ preferences, assuming too many item features leads to overfitting the model and degrades the accuracy. The degrading effect of training latent factor models with too many latent features have been emphasised by some researchers in the literature 118, 43. For example, in order to find the features the users care most about, Zhang et al. 118 carried out some analytical experiments and showed that around 15 features would be sufficient to capture the users’ preferences in two datasets 69.

Similarly, Jiang et al. 43 argued that if a small number of factors was used, the recommender system would not be able to distinguish between users or items. Conversely, if the number of features is too large, users and items will be too unique for the system to calculate their similarities, and the complexity will increase considerably. Therefore, the major research question that we are interested in in this chapter is: how many item features define the user preferences, and in particular how can the system avoid the degrading effect of irrelevant features? The work in this chapter proposes a feature-aware extension of PMF called FAPMF to address this issue. FAPMF modifies the learning process in PMF, so that the latent features are trained incrementally, and helps with overcoming the challenge of degrading effect of irrelevant features in PMF.

The rest of this chapter is organised as follows: In Section 3.1 we introduce FAPMF.
In Section 3.2 we first explain the experimental setup, and then report on the results of FAPMF. In Section 3.2.2 we give more analysis on the results obtained in Section 3.2.1 and finally we conclude the chapter in Section 3.4 by highlighting the main findings.

3.1 Feature-aware PMF (FAPMF)

To address the problem of irrelevant features in matrix factorisation, we extend PMF by incrementally adding features in the gradient descent process. In Algorithm 1, first we calculate and store the learning ratios for each of the features (line 3). As studies have shown (e.g. [118, 43]), only a limited number of factors are responsible for most of the observed user rating patterns. Therefore, we believe that learning the features incrementally with more learning iterations dedicated to the first features would better capture the user rating patterns. In other words, first we let the model learn the most important feature by iterating over the user ratings and estimating the matrices $\mathbf{P}$ and $\mathbf{Q}$ with only one factor. In the next iteration, we add one more feature to the model (line 6) and go over another learning iteration (line 9). The ratio of the number of iterations to the total number of iterations when each feature is added, is determined by a Lamé oval (factor ratio). In a Lamé oval, the values of the independent variable $x$ and the dependent variable $y$ are calculated according to the parametric function with $x = \sin(t) \frac{2}{r}$ and $y = \cos(t) \frac{2}{r}$. This function for different values of $r$ ($r = 0.2, 0.4, 0.6, 0.8, 1$) is depicted in Figure 3.1. Using smaller values in this function means that the model uses more iterations on features that are learnt earlier in the process.

**Algorithm 1 FAPMF: Train Model**

```
1: double TrainModel(Matrix P, Matrix Q, int D, double r, int maxIter, double γ)
2: {
3:   list factorRatios ← GetFactorRatios(D, r);
4:   int f ← 1;
5:   totalNumFactors ← D;
6:   for f ≤ totalNumFactors do
7:     D ← f;
8:     int l ← 1;
9:     for i ≤ factorRatios.get(f) × totalNumFactors × maxIter do
10:        int u ← l;
11:        for u ≤ N do
12:          int j ← f;
13:          for j ≤ M do
14:            Update $P_{ij}$ and $Q_{ij}$ using gradient descent derivatives and with learning rates γ;
15:            Recalculate and update the model error;
16:            $t ← t + 1$;
17:            end for
18:        u ← u + 1;
19:      end for
20:      Adjust the learning rate γ based on the model error;
21:      l ← l + 1;
22:    end for
23:    f ← f + 1;
24:  end for
25: }
```

This model is a heuristic version of stochastic gradient descent search that is employed in PMF. In Algorithm 1 we start the search process by only considering the group of model
Algorithm 2 FAPMF: Get Factor Ratios

1: List<int> GetFactorRatios(int numFactors, double r)
2: {
3:     List<double> FactorRatios;
4:     int f = 0;
5:     double sum = 0;
6:     for f ≤ numFactors do
7:         double x = \frac{1 - r^f}{1 - r};
8:         FactorRatios.set(f, (1 - x^f)^2);
9:         sum = sum + FactorRatios.get(f);
10:     f ← f + 1;
11:     end for
12:     f = 1;
13:     for f ≤ numFactors do
14:         FactorRatios.set(f, FactorRatios.get(f) / sum);
15:         f ← f + 1;
16:     end for
17:     return FactorRatios;
18: }

parameters that belong to the first factor (line 9). By doing this, in fact we aggregate most of the pattern in the first factors, and avoid the degrading effect of the less relevant factors which are trained using less number of iterations.

![Figure 3.1: The Lamé oval used in line 8 in algorithm 2 for different values of r (r = 0.1, 0.2, ..., 1, 2, ..., 10)](image)

3.2 Experiments

In order to evaluate the effectiveness of FAPMF, we conducted a series of experiments on two benchmark recommendation datasets, Movielens and Filmtrust.

In order to show the effectiveness of FAPMF, we compared the results against the recommendation quality of three popular methods for discovering latent factor models, that is, PMF, BPMF, and NMF. In this paper, 80% of the ratings are randomly chosen for training and the remaining 20% are used for validation. We use 0.001 for both user and item regularisation parameters (\( \lambda_P \) and \( \lambda_Q \)) and 0.01 for the learning rate (\( \gamma \)). We
### Table 3.1: Performance comparison of FAPMF in a) Filmtrust and b) Movielens datasets

<table>
<thead>
<tr>
<th>Model</th>
<th>D</th>
<th>MAE Avg</th>
<th>Stdev</th>
<th>RMSE Avg</th>
<th>Stdev</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>0.6864</td>
<td>0.0073</td>
<td>0.9295</td>
<td>0.0120</td>
<td></td>
</tr>
<tr>
<td>FAPMF</td>
<td>100</td>
<td>0.6851</td>
<td>0.0023</td>
<td>0.9264</td>
<td>0.0026</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.6828</td>
<td>0.0109</td>
<td>0.9281</td>
<td>0.0115</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.6755</td>
<td>0.0064</td>
<td>0.9209</td>
<td>0.0090</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8522</td>
<td>0.0051</td>
<td>1.1331</td>
<td>0.0066</td>
<td>19.46 17.97</td>
</tr>
<tr>
<td>PMF</td>
<td>100</td>
<td>0.8309</td>
<td>0.0039</td>
<td>1.1165</td>
<td>0.009</td>
<td>17.55 17.02</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.854</td>
<td>0.0101</td>
<td>1.1479</td>
<td>0.0118</td>
<td>20.05 19.15</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.7874</td>
<td>0.014</td>
<td>1.1779</td>
<td>0.0192</td>
<td>22.77 21.82</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.7624</td>
<td>0.0112</td>
<td>1.0100</td>
<td>0.0120</td>
<td>9.97 7.97</td>
</tr>
<tr>
<td>BPMF</td>
<td>100</td>
<td>0.8019</td>
<td>0.0106</td>
<td>1.0666</td>
<td>0.0145</td>
<td>14.57 15.06</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8208</td>
<td>0.0101</td>
<td>1.0927</td>
<td>0.0134</td>
<td>16.81 15.06</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.7407</td>
<td>0.0063</td>
<td>0.9905</td>
<td>0.0078</td>
<td>7.33 6.16</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.7079</td>
<td>0.0038</td>
<td>0.9478</td>
<td>0.0051</td>
<td>3.22 2.26</td>
</tr>
<tr>
<td>NMF</td>
<td>100</td>
<td>0.7548</td>
<td>0.0047</td>
<td>0.9584</td>
<td>0.0054</td>
<td>13.89 14.38</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.7457</td>
<td>0.0043</td>
<td>0.9542</td>
<td>0.0047</td>
<td>8.95 8.46</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.7437</td>
<td>0.0035</td>
<td>0.9530</td>
<td>0.0047</td>
<td>7.17 6.27</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.0106</td>
<td>0.0074</td>
<td>1.3062</td>
<td>0.0082</td>
<td>24.89 26.35</td>
</tr>
<tr>
<td>PMF</td>
<td>100</td>
<td>0.8765</td>
<td>0.0029</td>
<td>1.1931</td>
<td>0.0029</td>
<td>13.89 14.38</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8190</td>
<td>0.0051</td>
<td>1.0424</td>
<td>0.0056</td>
<td>8.95 8.46</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8006</td>
<td>0.0043</td>
<td>1.0167</td>
<td>0.0063</td>
<td>7.11 6.27</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8046</td>
<td>0.0066</td>
<td>1.0375</td>
<td>0.0079</td>
<td>5.65 7.26</td>
</tr>
<tr>
<td>BPMF</td>
<td>100</td>
<td>0.8182</td>
<td>0.0067</td>
<td>1.0544</td>
<td>0.0083</td>
<td>7.75 9.11</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8254</td>
<td>0.0065</td>
<td>1.0611</td>
<td>0.0088</td>
<td>9.66 10.07</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8381</td>
<td>0.0072</td>
<td>1.0773</td>
<td>0.0095</td>
<td>11.27 11.54</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8742</td>
<td>0.0048</td>
<td>1.1332</td>
<td>0.0049</td>
<td>13.17 15.1</td>
</tr>
<tr>
<td>NMF</td>
<td>100</td>
<td>0.8694</td>
<td>0.0056</td>
<td>1.1250</td>
<td>0.0093</td>
<td>13.18 14.81</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8550</td>
<td>0.0051</td>
<td>1.1025</td>
<td>0.0054</td>
<td>12.79 13.45</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8392</td>
<td>0.0074</td>
<td>1.0807</td>
<td>0.0080</td>
<td>11.38 11.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average Improvement</td>
<td>12.80 11.53</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Model</th>
<th>D</th>
<th>MAE Avg</th>
<th>Stdev</th>
<th>RMSE Avg</th>
<th>Stdev</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAPMF</td>
<td>50</td>
<td>0.7591</td>
<td>0.0023</td>
<td>0.9620</td>
<td>0.0022</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.7545</td>
<td>0.0047</td>
<td>0.9584</td>
<td>0.0054</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.7457</td>
<td>0.0043</td>
<td>0.9542</td>
<td>0.0047</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.7437</td>
<td>0.0035</td>
<td>0.9530</td>
<td>0.0047</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.0106</td>
<td>0.0074</td>
<td>1.3062</td>
<td>0.0082</td>
<td>24.89 26.35</td>
</tr>
<tr>
<td>PMF</td>
<td>100</td>
<td>0.8765</td>
<td>0.0029</td>
<td>1.1931</td>
<td>0.0029</td>
<td>13.89 14.38</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8190</td>
<td>0.0051</td>
<td>1.0424</td>
<td>0.0056</td>
<td>8.95 8.46</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8006</td>
<td>0.0043</td>
<td>1.0167</td>
<td>0.0063</td>
<td>7.11 6.27</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8046</td>
<td>0.0066</td>
<td>1.0375</td>
<td>0.0079</td>
<td>5.65 7.26</td>
</tr>
<tr>
<td>BPMF</td>
<td>100</td>
<td>0.8182</td>
<td>0.0067</td>
<td>1.0544</td>
<td>0.0083</td>
<td>7.75 9.11</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8254</td>
<td>0.0065</td>
<td>1.0611</td>
<td>0.0088</td>
<td>9.66 10.07</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8381</td>
<td>0.0072</td>
<td>1.0773</td>
<td>0.0095</td>
<td>11.27 11.54</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8742</td>
<td>0.0048</td>
<td>1.1332</td>
<td>0.0049</td>
<td>13.17 15.1</td>
</tr>
<tr>
<td>NMF</td>
<td>100</td>
<td>0.8694</td>
<td>0.0056</td>
<td>1.1250</td>
<td>0.0093</td>
<td>13.18 14.81</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>0.8550</td>
<td>0.0051</td>
<td>1.1025</td>
<td>0.0054</td>
<td>12.79 13.45</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.8392</td>
<td>0.0074</td>
<td>1.0807</td>
<td>0.0080</td>
<td>11.38 11.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average Improvement</td>
<td>11.64 12.39</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b)

Table 3.1: Performance comparison of FAPMF in a) Filmtrust and b) Movielens datasets

We tested these models on multiple datasets using different parameter values and observed that these values generally result in better accuracies. We also set 200 as maximum number of learning iterations (\(maxIter\) in Algorithm 1). Through experiments, we also noticed that using \(\frac{1}{5}\) for \(r\) yields better accuracy for our model, so we used this value throughout our experiments. Each model training and testing is repeated for 5 times to eliminate the randomness in the results and therefore to assure that the results are more reliable.

#### 3.2.1 Results

The evaluation results for the aforementioned latent factor models as well as FAPMF on Filmtrust and Movielens datasets are shown in Tables 3.1a and 3.1b respectively.

From Table 3.1, we can see that FAPMF outperforms all three latent factor models in almost all settings. The percentages on the right side of each table show the amount
Chapter 3. Feature-Aware Modelling of Preferences

of improvement that FAPMF makes over the corresponding latent factor model. FAPMF yields extensive improvements over basic PMF in both datasets. We can also see that the accuracy of FAPMF in both datasets consistently decreases as the number of factors increases. This is probably because adding more features further complicates the solution space, by adding irrelevant features with less or even negative contribution to the accuracy. Therefore, the model fails to find the optimal solutions. In FAPMF, this problem is alleviated by learning the factors in an incremental way. In this model, first the model is trained with one factor for a specific number of iterations. Then a second factor is added and the training is done for less number of iterations. In Filmtrust dataset, on average FAPMF improves the MAE accuracy by 19.96%, 15.42%, and 3.01% relative to PMF, BPMF, and NMF respectively. The average improvement of FAPMF for RMSE is 19%, 13.72%, and 1.89% with respect to PMF, BPMF, and NMF respectively. In Movielens dataset, FAPMF improves 13.71%, 12.63%, and 8.58% over PMF, BPMF, and NMF respectively. The improvements for RMSE in this dataset are 13.87%, 13.8%, and 9.5%, respectively.

3.2.2 Effect of parameter $r$ in FAPMF

As explained in Section 3.1 FAPMF alleviates the negative effect of irrelevant features by training the model in a feature-aware manner. In this method, the first features are given more importance than the latter features, by doing more learning iterations over them. The parameter $r$ in Algorithm 1 controls the number of iterations over each feature. As Figure 3.1 shows, if $r < 1$, the algorithm performs more iterations over the first features and few iterations over the later features. If $r > 1$, the training iterations would be equally distributed over all features. To demonstrate the effect of the parameter $r$ on FAPMF’s performance, we depict the change in RMSE with respect to learning iterations for each separate factor with two different values for $r$ on the Filmtrust and Movielens datasets.

We observe that the value of $r$ has a major impact on the recommendation accuracy. Interestingly, we also notice that when the value of $r$ is 10, the accuracy results are close to those of basic PMF. In this case, all the features are treated equally, i.e. all features get the same number of training iterations. Consequently, the accuracy of FAPMF approaches the accuracy of basic PMF. However, even in this case, the model achieves higher accuracies than basic PMF. The reason is that when the search is carried out in a feature-aware manner, the underlying additive independence assumption makes it easier to find the optimal solution. As we observe in this figure, $r = 0.01$ yields the best results for the
Chapter 3. Feature-Aware Modelling of Preferences

Figure 3.2: Performance of FAPMF with respect to parameter $r$

MovieLens dataset with both 100 and 150 factors and for the Filmtrust dataset with 150 factors, and $r = 0.1$ yields the best results for the Filmtrust dataset with 100 factors. We also notice that in general the model performs better with 150 than 100 factors.

To demonstrate the effect of the parameter $r$ on FAPMF’s performance, in Figures 3.3 and 3.4 we depict the change in RMSE with respect to learning iterations for each separate factor with two different values for $r$ on the Filmtrust and MovieLens datasets. We assume $r = 1$ and $r = 1000$, and we set the number of factors to 12. These two values for $r$ result in different numbers of iterations for each factor. However, to be able to show both cases in one diagram, we set all iteration numbers to 400 and plot the accuracies for an equal number of iterations for both models. To ensure that the results are not subject to randomness, each experiment is repeated 10 times, and the average errors of the iterations are used.

In Figures 3.3 and 3.4, the diagram on the top-left corner of each figure belongs to the first factor, and the diagram on the bottom-right corner of each figure belongs to the 12th factor. We can observe that when $r = 1000$, 200 iterations are used for all factors,
since the errors for all factors stop at iteration 200. However, when $r = 1$, the first factors receive more iterations. The deteriorating effect of later factors can be clearly seen in both datasets. We observe that in almost all factors, for a specific number of iterations the accuracy keeps improving before it deteriorates. Specifically we observe how with $r = 1$ the model avoids the degrading effect of irrelevant factors by limiting the number of learning iterations over that factor. For example, the second factor stops training at iteration 200 when $r = 1000$, whereas it stops at iteration 300 when $r = 1$. Consequently, a lower value of parameter $r$ in both datasets always results in better accuracies than a higher $r$ value. We observe that as more factors are added, the gap in accuracy between the models with $r = 1$ and $r = 1000$ becomes larger. We also clearly see that the first factors reduce the error to a larger extent than the later factors (and this effect is additive). This actually means that the first factors learnt by this model capture more of the rating patterns than the later factors and therefore they are more relevant. Therefore, by incremental training of the factors, FAPMF is able to avoid the degrading effect of training irrelevant factors, and improves the accuracy.

### 3.3 Summary of key points

1. Deciding the right number of factors to use in latent factor models is a problem with these models.

2. Using too many factors considerably increases the training effort and can prevent the model from finding optimal solution.

3. A small number of features on the other hand might not be sufficient for the model to properly represent the users and items.

4. A possible way of overcoming this issue is to incrementally add the factors through the training iterations.

5. This method improves accuracy by stopping the training for each factor before the model overfits for that factor.

### 3.4 Conclusion

In this chapter, we addressed the negative effect irrelevant features have in latent factor models, which has been observed by researchers. In order to tackle this problem, we proposed a feature-aware latent factor model based on PMF. In FAPMF, the user preferences
Figure 3.3: Performance of FAPMF with respect to parameter \( r \) for each factor in Filmtrust dataset
Figure 3.4: Performance of FAPMF with respect to parameter $r$ for each factor in Movie-lens100k dataset
are learnt by incrementally iterating over each feature, and minimising the prediction error by using stochastic gradient descent. We used the Lamé oval to control the proportion of learning iterations over individual features. Using the Filmtrust and MovieLens datasets with different numbers of features, the experiments showed that FAPMF considerably improves on basic PMF. The experiments also showed that FAPMF outperforms two other popular latent factor models based on Matrix Factorisation (BPMF and NMF). Therefore, we empirically showed that it is possible to avoid the degrading effect of irrelevant features by incrementally training the latent factor models. Incorporating feature-awareness to the latent factor models was the first step we took in line with the research objective of improving the recommendation accuracy. The following chapters focus on capturing different aspects and explaining them to the users to achieve the research objectives.
Chapter 4

Modelling Conditional Feature Value Preferences

Despite their popularity and good accuracy, LFMIs have so far disregarded some aspects of information present in the rating data, potentially at the expense of accuracy. For instance, by modelling latent features, but not latent feature values, they implicitly assume that all values for item features are equally preferred by all users. For example, it is likely that one of the latent features in typical product ratings would describe price, but since feature values (low price, high price) are not captured in the model, an average-based model implicitly assumes that users generally prefer cheap items over expensive ones, or, say, that a movie with a specific level of action is equally preferred by all users, which is clearly not true. Modelling feature values is not easy when ratings are sparse, as is typically the case. Nonetheless, we speculate that capturing these aspects could improve the accuracy of recommendations. Also, leading matrix factorisation methods such as (PMF) \[79\], Bayesian Probabilistic Matrix Factorisation (BPMF) \[78\] and Biased Matrix Factorisation \[52\] (BiasedMF) assume that the user preferences can be represented by a linear function. However, conditional preferences naturally exist in human preferences \[61\]. For example, a user may prefer a hotel near the sea in summer, while preferring a hotel in the mountains in winter. Therefore, the user preferences over feature values actually depend on other features and their values, and modelling such dependencies is likely to enhance the accuracy of recommendations.

In this chapter, we address the research question, how can we efficiently model preferences over different item feature values given the sparsity of user ratings and how much would the accuracy improve as a consequence? Although this question has been addressed before \[95\ \[118\], all methods proposed to date need additional information (e.g. user re-
Chapter 4. Modelling Conditional Feature Value Preferences

views) to extract user preferences over feature values. In this study, we capture feature values from rating data alone by adding additional matrices to PMF \[79\]. Specifically, in this chapter we make the following contributions:

1. We show how the classic matrix factorisation techniques can be extended to capture the conditional preferences over feature values.
2. We mathematically prove the suitability of the techniques that were used to capture conditional preferences over feature values.
3. We experimentally demonstrate the effectiveness of the proposed techniques to capture conditional preferences over feature values.

The rest of the chapter is organised as follows: first in Section 4.1, we provide a simple example that illustrates the limitations of the existing latent factor models based on matrix factorisation, in capturing conditional preferences over feature values. Then in Section 4.2, we introduce the proposed models, biased matrix factorisation with conditional preference over feature values (denoted by CFVSVF) and present the mathematical details and the supporting theorems. In Section 4.3, we first explain the experimental setup, and then report on the results of CTFVSVF. Finally we conclude the chapter in Section 4.5, by summarising the main findings.

To the best of our knowledge, the current work is the first attempt at incorporating user conditional preferences over feature values into the latent factor models without the need for any additional source of feedback (e.g. user reviews). It has the following advantages over previous work:

1. It relies solely on user ratings.
2. The feature values are extracted as latent features, which enables us to include features that are not definable or interpretable \[52\].
3. The model also captures interactions between features and feature values.

4.1 Motivating example

In the following, we show that the formulation of current latent factor models is not representative enough to capture the preferences of these users. In the example, there are two users who show their preferences over three items by giving ratings between 1 and 5. This results in a user-item ratings matrix \( R \). In all the methods based on the basic matrix
factorisation, the user-item ratings matrix is factorised into two matrices, a user-specific matrix \( P \) and an item-specific matrix \( Q \).

First, let us assume the feature values of the items are equal to the values represented in matrix \( Q \) (the values of matrix \( W \) are all one, and the values of matrix \( Z \) are all zero), and there are conditional dependencies \[12\] between the two item features, \( f_1 \) and \( f_2 \). In matrix factorisation, the ratings are predicted by multiplying \( P \) and \( Q^T \). This yields the estimated ratings matrix \( \hat{R} \).

We assume that user \( u_1 \) prefers item \( i_1 \) over item \( i_2 \), and item \( i_4 \) over item \( i_3 \). There are also conditional dependencies between feature \( f_1 \) and feature \( f_2 \), which means that Eq. \[4.1\] holds \[12\].

\[ u_1 : R_{11} > R_{12}, R_{14} > R_{13} \]  

\[4.1\]

In PMF, this results in the two inequalities \( ea + gb > fa + gb \) and \( fa + hb > ea + hb \), which obviously contradict each other. Therefore, this example shows that the simple linear function in the basic matrix factorisation is not representative enough to capture the preferences that include conditional dependencies between item features.

Let us assume that user \( u_1 \) prefers item \( i_1 \) over item \( i_2 \), while user \( u_2 \) has the opposite preference over these two items. This results in the following preferences for users \( u_1 \) and \( u_2 \):

\[ u_1 : R_{11} > R_{12} \]  

\[ u_1 : R_{11} > R_{12} \]  

\[4.2\]

\[ u_2 : R_{21} > R_{22} \]  

\[4.3\]

Replacing the values of \( R_{11}, R_{12}, R_{21}, \) and \( R_{22} \) from the matrix \( \hat{R} \) results in Eq. \[4.4\]
Let us assume user $u_2$ prefers item $i_3$ over item $i_4$ and user $u_1$ has the opposite preference. Then Eqs. 4.5 and 4.6 would be true for the ratings given by the users over items $i_3$ and $i_4$.

\[ u_1 : R_{13} < R_{14} \]  
\[ u_2 : R_{23} > R_{24} \]

This yields Eq. 4.7 which clearly contradicts Eq. 4.4.

\[ ea + fc < fa + ec \]  
\[ (4.7) \]

Therefore, as we see, the classic matrix factorisation cannot capture simple conditional preferences as above.

So far, we assumed that the values of matrices $W$ and $Z$ are all one and zero respectively. We assume that there are differences between the users $u_1$ and $u_2$ in preferring item feature values, and take the values of matrix $W$ to be 1 for user $u_1$, and -1 for user $u_2$. This means that these users have opposite preferences for the values of the items in matrix $Q$. This results in the following matrix $\hat{R}$:

\[
\begin{array}{cccc}
  & i_1 & i_2 & i_3 & i_4 \\
 u_1 & ea + gb & fa + gb & ea + hb & fa + hb \\
 u_2 & -(ec + gd) & -(fc + gd) & -(ec + hd) & -(fc + hd) \\
\end{array}
\]

If user $u_1$ prefers $i_1$ over $i_2$, we have:

\[-(ec + gd) > -(fc + gd) \]

\[ (4.8) \]

However, according to matrix factorisation we have $ec + gd > fc + gd$ which contradicts Eq. 4.8. Therefore, matrix factorisation cannot capture user preferences when they have different preferences over feature values.
4.2 Conditional preferences model

The model proposed in the following section extends SVD++. As mentioned in Chapter 2, SVD++ has been proposed by Koren and Bell [51] to incorporate the implicit feedback and user and item bias values into PMF. In SVD++, the rating given by user \( u \) to item \( i \) \( \hat{R}_{ui} \) is estimated using the Eq. 4.9

\[
\hat{R}_{ui} = \mu + b_u + b_i + \sum_{f=1}^{D} (P_{uf} + \sum_{y_i \in I_u} y_{if}) Q_{jf} \tag{4.9}
\]

In Eq. 4.9, \( \mu \) denotes the average rating given by all users to all items, \( I_u \) is the set of items rated by user \( u \), \( y \in \mathbb{R}^{M \times D} \) is an item-specific latent matrix to capture the implicit feedback.

To address the problem of capturing preferences over feature values in matrix factorisation, we extend the basic matrix factorisation by adding matrices \( W \) and \( Z \) to learn the preferences over item feature values, and matrix \( Y \) to capture the dependencies between features. The proposed method is abbreviated as CFVSVD. The graphical model of this method is depicted in Figure 4.1.

4.2.1 Mathematical details

Similar to PMF, in CFVSVD the user preferences are modelled as a Bayesian network [48]. Figure 4.1 shows the topology or the structure of the Bayesian network for user preferences that are modelled by CFVSVD. According to this figure, the probabilities of the matrices \( P, Q, W, Z, Y, y \) and vectors \( b_u \) and \( b_i \) are dependent on the hyper-parameters \( \sigma_P, \sigma_Q, \sigma_W, \sigma_Z, \sigma_Y, \sigma_y, \sigma_{b_u} \)
Chapter 4. Modelling Conditional Feature Value Preferences

and \(\sigma_{bi}\) respectively. Likewise, the probability of the ratings in matrix \(R\) is conditional upon the matrices \(P, Q, W, Z, Y, y\) and vectors \(bu\) and \(bi\). In this method, we use a quadratic polynomial function to capture the conditional preferences. In this function, matrix \(W\) is used to capture the "gradient" values, matrix \(Z\) is used to learn the "intercept" values, and matrix \(Y\) is used to capture the "interactions" between features. As Figure 4.1 shows, gradient and intercept matrices have the same dimensions as the user matrix \(P\). In CFVSVD, the matrix value \(P_{uf}\) denotes the general importance attributed to item feature \(f\) by user \(u\). Matrices \(W\), \(Q\), and \(Z\) on the other hand are used to model the user preferences over item feature values. Similar to the PMF, matrix \(Q\) shows the degree to which an item has a particular feature. For example, assuming the existence of a latent feature for "price", \(Q_{j}\) takes a high value for expensive items and a low value for cheap ones. Similar to SVD++, \(y\) is the item-specific latent matrix to capture implicit feedback.

According to the Bayesian network of the user preferences in Figure 4.1 in the proposed model, the posterior probability of user and item matrices is calculated based on Eq. 4.10 which incorporates user bias \(bu\) and item bias \(bi\) and the conditional preferences over feature values captured by matrices \(W\), \(Q\), \(Z\), and \(Y\):

\[
p(P, Q, W, Z, Y, y, bu, bi | R, \sigma, \sigma_P, \sigma_Q, \sigma_W, \sigma_Z, \sigma_Y, \sigma_y, \sigma_{bu}, \sigma_{bi}) \approx p(R | P, Q, W, Z, Y, y, bu, bi, \sigma) \\
\times p(P | \sigma_P) \times p(Q | \sigma_Q) \times p(y | \sigma_y) \\
\times p(W | \sigma_W) \times p(Z | \sigma_Z) \times p(Y | \sigma_Y) \\
\times p(bu | \sigma_{bu}) \times p(bi | \sigma_{bi})
\]

(4.10)

As in PMF, we assume that the probabilities in Eq. 4.10 follow a Gaussian distribution. Using this assumption Eq. 4.10 yields Eq. 4.11

\[
p(P, Q, W, Z, Y, y, bu, bi | R, \sigma, \sigma_P, \sigma_Q, \sigma_W, \sigma_Z, \sigma_Y, \sigma_y, \sigma_{bu}, \sigma_{bi}) \\
\approx \prod_{u=1}^{N} \prod_{j=1}^{M} [N(R_{uj} | \hat{R}_{uj}, \sigma^2)]^{\hat{R}_{uj}} \\
\times \prod_{u=1}^{N} [N(P_u | 0, \sigma_P^2)] \times \prod_{j=1}^{M} [N(Q_j | 0, \sigma_Q^2)] \times \prod_{j=1}^{M} [N(y_j | 0, \sigma_y^2)] \\
\times \prod_{u=1}^{N} [N(W_u | 0, \sigma_W^2)] \times \prod_{u=1}^{N} [N(Z_u | 0, \sigma_Z^2)] \times \prod_{f=1}^{D} [N(Y_f | 0, \sigma_Y^2)] \\
\times \prod_{u=1}^{N} [N(bu_u | 0, \sigma_{bu}^2)] \times \prod_{j=1}^{M} [N(bi_j | 0, \sigma_{bi}^2)]
\]

(4.11)

Maximising Eq. 4.11 would be equivalent to minimising the error according to Eq. 4.12.
argmin_{P,Q,W,Z,Y,y,bu,bi}[E = \frac{1}{2} \sum_{u=1}^{N} \sum_{j=1}^{M} I_{uj}(\hat{R}_{uj} - R_{uj})^2 \\
+ \frac{\lambda_P}{2} \sum_{u=1}^{N} \|P_{u}\|_{Frob}^2 + \frac{\lambda_Q}{2} \sum_{j=1}^{M} \|Q_{j}\|_{Frob}^2 + \frac{\lambda_b}{2} \sum_{u=1}^{N} \|y_{u}\|_{Frob}^2 \\
+ \frac{\lambda_W}{2} \sum_{u=1}^{N} \|W_{u}\|_{Frob}^2 + \frac{\lambda_Z}{2} \sum_{j=1}^{M} \|Z_{j}\|_{Frob}^2 \\
+ \frac{\lambda_Y}{2} \sum_{j=1}^{M} \|Y_{j}\|_{Frob}^2 + \frac{\lambda_{bu}}{2} \sum_{u=1}^{N} \|b_{u}\|_2^2 + \frac{\lambda_{bj}}{2} \sum_{j=1}^{M} \|b_{j}\|_2^2]
] (4.12)

where \(\lambda_W = \frac{\sigma_w^2}{\sigma_W^2}, \lambda_Z = \frac{\sigma_z^2}{\sigma_Z^2}, \lambda_Y = \frac{\sigma_y^2}{\sigma_Y^2}, \lambda_y = \frac{\sigma_y^2}{\sigma_y^2}, \lambda_{bu} = \frac{\sigma_{bu}}{\sigma_{bu}}, \lambda_{bj} = \frac{\sigma_{bj}}{\sigma_{bj}}\). \(\mu\) denotes the global average of the observed ratings, and \(bu_u\) and \(bi_j\) denote biases for user \(u\) and item \(j\) respectively. The interaction matrix \(Y \in \mathbb{R}^{D \times D}\) in the Eq. 4.12 is a symmetrical square matrix and is added to capture the dependencies or interactions between item features. Using stochastic gradient descent, the gradient values for the model parameters in CFVSVD are obtained according to Eqs. 4.13 through 4.19.

\[
\frac{\partial E}{\partial bu_u} = e_{uj} + \lambda_{bu} bu_u 
\] (4.13)
\[
\frac{\partial E}{\partial bi_j} = e_{uj} + \lambda_{bi} bi_j
\] (4.14)
\[
\frac{\partial E}{\partial P_{uf}} = e_{uj}(W_{uf}Q_{jf} + Z_{uf}) + \lambda_P P_{uf}
\] (4.15)
\[
\frac{\partial E}{\partial Q_{jf}} = W_{uf}(e_{uj}(P_{uf} + 2 \sum_{f'=1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'}Y_{f'})) + \lambda_Q Q_{jf}
\] (4.16)
\[
\frac{\partial E}{\partial W_{uf}} = Q_{jf}(e_{uj}(P_{uf} + 2 \sum_{f'=1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'}Y_{f'})) + \lambda_W W_{uf}
\] (4.17)
\[
\frac{\partial E}{\partial Z_{uf}} = e_{uj}(P_{uf} + 2 \sum_{f'=1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'}Y_{f'})) + \lambda_Z Z_{uf}
\] (4.18)
\[
\frac{\partial E}{\partial Y_{f'f'}} = e_{uj}(W_{uf}Q_{jf} + Z_{uf})(W_{uf}Q_{jf'} + Z_{uf'}) + \lambda_Y Y_{f'f'}
\] (4.19)

\(\forall i \in I_u: \frac{\partial E}{\partial y_{if}} = e_{uj}(W_{uf}Q_{jf} + Z_{uf}) + \lambda_y y_{if}
\] (4.20)
\[
\frac{\partial E}{\partial Y_{f'f'}} = \frac{\partial E}{\partial Y_{f'f'}}
\] (4.21)

where \(\gamma\) denotes the learning rate in stochastic gradient descent. The error \(e_{uj}\) is calculated according to Eq. 4.22.
\[ e_{u_j} = \hat{R}_{u_j} - R_{u_j} \quad (4.22) \]

In CFVSVD, the estimated rating of the item \( j \) given by user \( u \), \( \hat{R}_{u_j} \) is calculated based on Eq. 4.23.

\[
\begin{align*}
\hat{R}_{u_j} &= \mu + b_u + b_{ij} + \sum_{f=1}^{D} \sum_{i \in I_u} (\sum_{f \in D} y_{if} + P_{uf})(W_{uf}Q_{jf} + Z_{uf}) + \\
&\quad \sum_{f' = 1}^{D} \sum_{f=1}^{D} Y_{ff'}(W_{uf}Q_{jf} + Z_{uf})(W_{uf'}Q_{jf'} + Z_{uf'}) 
\end{align*}
\quad (4.23)
\]

Eq. 4.23 means that the user \( u \)’s rating over item \( j \) can be estimated by calculating the average rating \( \mu \) that the users normally give to the items, as well as the user and item biases (\( b_u \) and \( b_{ij} \)) and the user’s preference which is captured by matrices \( P, Q, W, Z, \) and \( Y \) and the implicit feedback matrix \( Y \). When all entries in gradient matrix \( W \) are fixed at one and the entries in intercept matrix \( Z \) and interaction matrix \( Y \) set to zero, the proposed CFVSVD method is reduced to SVD++ \[51\][52] which is an extension of PMF with user and item biases.

### 4.2.2 Absence of model of feature values in PMF

In this study, we propose an extension to PMF \[79\], which models user preferences over feature values. In the first instance, we use a lemma to demonstrate that the user preferences over feature values are not covered in PMF.

**Lemma 1.** Eq. 4.9 in Section 4.2.1, which represents the PMF model, does not include the user preferences over feature values.

**Proof.** In PMF, user \( i \)’s preferences over an item \( j \) are estimated by the product of vector \( P_u \) and vector \( Q_j \) (\( \hat{R}_{u_j} = P_uQ_j^T \)). Vector \( P_u \) is user-specific and models the relative importance of each item feature \( f \), while vector \( Q_j \) is item-specific and models the degree to which each item has an item feature. The linear combination employed in PMF suggests that it follows additive independence in multi-attribute theory \[13\]. According to additive independence, user \( u \)’s utility (preference) over item \( j \) with the features \( Q = (Q_{j1}, Q_{j2}, ..., Q_{jD}) \) is given by Eq. 4.24.

\[
\forall u \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M\} : e^{u,j}(Q_j) = e^{u,j}(Q_{j1}, Q_{j2}, ..., Q_{jD}) = \sum_{f=1}^{D} P_{uf}g^{u,f}(Q_{jf}) 
\quad (4.24)
\]
Chapter 4. Modelling Conditional Feature Value Preferences

$g_{uf}(.)$ as the component value function denotes user $u$’s preferences over the value of feature $f$ in an item. $e_{uf}$ is the utility value item $j$ for user $u$. Intuitively, $P_{uf}$ denotes user $u$’s preferences over item feature $f$, and the function $g_{uf}(.)$. As we can see, in PMF $g_{uf}(Q_{jf}) = Q_{jf}$, which means that the user’s preferences over feature values in PMF are independent of the preferences of user $u$ and are only a function of the item $j$’s features $(Q_{j1}, Q_{j2}, ..., Q_{jD})$. In other words, the differences between users in preferring item feature values are not taken into consideration in PMF.

4.2.3 Ability of the model to capture preferences over feature values

Apart from demonstrating, in an experimental evaluation, that the inclusion of a model for feature values adds information, and therefore accuracy, to the rating estimates, we also provide proof that even in the worst case, the additional model of feature values adds a small amount of information not captured in PMF.

**Theorem 1.** If there are $N$ users in the system whose ratings diverge, at least $\frac{1}{N}$ of the existing differences of user preferences for feature values can be captured by CFVSVD.

**Proof.** Theorem 1 postulates that if there are $N$ users in the system whose ratings diverge, at least $\frac{1}{N}$ of the existing differences of user preferences for feature values can be captured by CFVSVD.

In CFVSVD, a user $u$’s value function for item $j$, denoted $g_{uf}(Q_{jf})$ with $Q_{jf}$ denoting the general preference for feature $f$ of item $j$ is defined as shown in Eq. (4.25). The user preference for a feature is adjusted by a gradient value $W_{uf}$ and an intercept $Z_{uf}$.

\[ \forall u \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M\} : g_{uf}(Q_{jf}) = W_{uf}Q_{jf} + Z_{uf} \tag{4.25} \]

The CFVSVD extension only adds information to the model if there are differences in user preferences.

Let us assume that the two arbitrary users $u_1$ and $u_2$ have different feature value preferences. This means that for any item feature $Q_{jf}$, user $u_1$ has either a larger or smaller preference than user $u_2$. In order words, there is difference between these two users in preferring an item feature value $(Q_{jf})$. Taking price as an example, user $u_1$ can have a larger or smaller preference value than user $u_2$, for $\$1000$ as the value of the price feature of a laptop. Eq. (4.26) assumes user 1’s preference $g_{u1f}$ for an item feature $Q_{jf}$ is larger ($\succ$) than that of user 2. Eq. (4.27) states the opposite.

\[ \forall (j \in \{1, 2, 3, ..., M\}, f \in \{1, 2, 3, ..., D\}) : (g_{u1f}(Q_{jf}) \succ g_{u2f}(Q_{jf})) \tag{4.26} \]
\( \forall (j \in \{1, 2, 3, ..., M\}, f \in \{1, 2, 3, ..., D\}) : (g_{u_1}^f(Q_{jf}) \prec g_{u_2}^f(Q_{jf})) \) (4.27)

As Eq. 4.25 shows, this function only depends on the values of \( f \) and \( u \). Therefore, in the following, we simplify the notation by using the functions \( l^f \) and \( h^f \) instead of \( g_{u_1}^f \) and \( g_{u_2}^f \) respectively. The question is, how can we find the values of \( W_{u_1}^f, Z_{u_1}^f, W_{u_2}^f, Z_{u_2}^f \), so that the maximum number of differences can be captured. In other words, let’s assume that the users \( u_1 \) and \( u_2 \) have different feature value preferences for a feature \( f \) that are defined according to the Eqs. 4.28 and 4.29.

\[
\forall (u_1, u_2 \in \{1, 2, 3, ..., N\}, f \in \{1, 2, ..., D\}) : \\
(l^f(Q_{1f}) \succ h^f(Q_{1f})) \wedge \\
(l^f(Q_{2f}) \prec h^f(Q_{2f})) \wedge \\
(l^f(Q_{3f}) \succ h^f(Q_{3f})) \wedge \\
\cdots \\
(l^f(Q_{Mf}) \prec h^f(Q_{Mf}))
\]

\[
\forall (u_1, u_2 \in \{1, 2, 3, ..., N\}, f \in \{1, 2, ..., D\}) : \\
(Q_{1f} \succ X) \wedge \\
(Q_{2f} \prec X) \wedge \\
(Q_{3f} \succ X) \wedge \\
\cdots \\
(Q_{Mf} \prec X)
\]

where \( X = \frac{Z_{u_2}^f - Z_{u_1}^f}{W_{u_2}^f - W_{u_1}^f} \) and each predicate in Eq. 4.29 denotes one difference between two users in preferring the value of feature \( f \) for an item \( j \in \{1, 2, ..., M\} \). Regardless of the value of \( Q_{jf} \), we can obviously find the values of \( W_{u_1}^f, Z_{u_1}^f, W_{u_2}^f, Z_{u_2}^f \) such that the respective predicate in Eq. 4.29 holds for item \( j \). This means that for an item \( j \), the model can find the gradient and intercept values so that the feature value preference differences between users \( u_1 \) and \( u_2 \) for the item \( j \) are captured.

We can show that using the function in Eq. 4.25 at least half of the differences for all items can be captured by CFVSVD for the two users \( u_1 \) and \( u_2 \). Generally speaking, if there are \( N \) users and \( M \) items, the total number of possible differences would be equal to \( \frac{N(N-1)M}{2} \). Of these possible differences, the model can capture \( \frac{(N-1)M}{2} \). Therefore, \( \frac{1}{N} \)
of the total differences can be captured by the model. Interestingly, this is independent of the number of items and only depends on the number of users. For example, if the model is trained for 100 users, and there are 1000 items, the total number of possible differences is equal to 4950000, 49500 of which can be captured at least.

The values of $Q_{jf}$ in Eq. 4.29 can be ordered, and translated into an associated real string and binary string based on the following rule. The real string includes the $Q_{jf}$ values in ascending order, and the $\succ$ sign for each $Q_{jf}$ in the binary string is translated to zero (meaning a satisfied difference), and $\prec$ is translated to one (meaning a dissatisfied difference). For example, if there are 4 items with $(Q_{1f} = 1) \succ X$, $(Q_{2f} = 2) \prec X$, $(Q_{3f} = 3) \prec X$, and $(Q_{4f} = 4) \succ X$, the associated real string and binary string would be 1234 and 0110 respectively. Finding the value of $X$ in Eq. 4.29 is equivalent to finding its position in the real string (which is in turn equivalent to partitioning the associated binary string in two parts) in a way that the number of zeros in the left partition plus the number of ones in the right partition is greater than or equal to half the length of the string. For instance, for the sample strings 1234 and 0110, if we choose $X$ to be 2.5, there will be one zero in the left partition (01), and one 1 in the right partition (10) in the associated binary string 01 10. In formal terms, we need to prove Lemma 2.1 shown in Figure 4.2. There are 4 possible cases for the binary string $S$. These possible cases and the positioning of the binary string by positioning $X$ in the associated real string in each case are shown in Fig. 4.3.

As we see in Figure 4.3, in all cases we can position $X$ in a way that the equation $Z_1 + O_2 \geq \frac{M}{2}$ in Lemma 2.1 is satisfied, so we have proved Lemma 2.1. Therefore, for every pair of users, $\frac{M}{2}$ differences can be captured. Since finding the values of $X$ for the user
Chapter 4. Modelling Conditional Feature Value Preferences

\[ X_{\alpha_1} = M_{\alpha_1} \]

\[ X_{\alpha_2} = M_{\alpha_2} \]

\[ X_{\alpha_3} = M_{\alpha_3} \]

(a)

(b)

(c)

(d)

Figure 4.3: The possible cases for the binary string \( S \) and the position of \( X \) in the associated real string in each case

4.2.4 Correctness of CFVSV model

In this part, we establish the suitability of the extensions Eq. 4.23 adds to Eq. 4.9 for estimating the expected ratings of items.

**Theorem 2.** The user preferences can be estimated using Eq. 4.23.

**Proof.** We apply the Taylor-Maclaurin expansion to show that the rating estimates can be approximated using Eq. 4.23.

For simplicity, we introduce a function that produces an estimation of a user \( u \)'s preference values for an item \( j \) from the features \( Q_j \) of the item (\( \epsilon_u^j \)) and a function that returns a user \( u \)'s preference for an item \( j \) based on the user preferences for features of an item \( y^u \) and the features \( Q_j \) (\( \epsilon_u^y \)). The difference between \( \epsilon_u^j \) and \( \epsilon_u^y \) therefore captures the difference between Eq. 4.23 and Eq. 4.9.
\( \forall (u \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M\}, \epsilon_1^u \Rightarrow \exists \epsilon_2^u) : \)
\[
\epsilon_1^u(Q_{j1}, Q_{j2}, ..., Q_{jD}) = \epsilon_2^u(g^{u1}(Q_{j1}), g^{u2}(Q_{j2}), ..., g^{uD}(Q_{jD}))
\]

(4.30)

According to Eq. (4.30), for every arbitrary utility function \( \epsilon_1^u \) that maps the feature values of item \( j \) to a respective preference value, there is at least one function \( \epsilon_2^u \) that maps the user \( u \)'s preferences over feature values of item \( j \) to the same preference value.

Replacing the short notation for user preferences over feature values \( g^{uf} \) with the gradient \( W^{uf} \) and intercept values \( Z^{uf} \) that describe the linear functions of user preferences over feature values, we obtain Eq. (4.31).

\[
\forall (u \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M\}, \epsilon_1^u \Rightarrow \exists \epsilon_2^u) : \]
\[
\epsilon_1^u(Q_{j1}, Q_{j2}, ..., Q_{jD}) = \epsilon_2^u(W_u^1 Q_{j1} + Z_u^1, W_u^2 Q_{j2} + Z_u^2, ..., W_u^D Q_{jD} + Z_u^D)
\]

(4.31)

Applying a linear coordinate system transformation leads to Eq. (4.32).

\[
\epsilon_2^u(Q_{j1}, Q_{j2}, ..., Q_{jD}) = \epsilon_1^u\left(\frac{Q_{j1} - Z_u^1}{W_u^1}, \frac{Q_{j2} - Z_u^2}{W_u^2}, ..., \frac{Q_{jD} - Z_u^D}{W_u^D}\right)
\]

(4.32)

This proves that there is at least one function \( \epsilon_2^u \) as in Eq. (4.32) to satisfy Eq. (4.30).

Eq. (4.33) defines \( \epsilon_2^u(.) \) as Eq. (4.23), the equation whose suitability for estimating ratings we intend to prove.

\[
\epsilon_2^u(g^{u1}(Q_{j1}), g^{u2}(Q_{j2}), ..., g^{uD}(Q_{jD})) \approx C_{uj} + \sum_{f=1}^{D} \left( \sum_{i \in I_u} y_{if} + P_{uf} \right) g^{uf}(Q_{jf}) + \sum_{f=1}^{D} \left( \sum_{f'=1}^{D} Y_{ff'} g^{uf}(Q_{jf}) g^{uf'}(Q_{jf'}) \right)
\]

(4.33)

In Eq. (4.33) \( C_{uj} = \mu + bu_u + bi_j \). Therefore, to prove Eq. (4.23), we need to prove that Eqs. (4.30) and (4.33) hold.

We apply the Taylor-Maclaurin series expansion, which can estimate any arbitrary function using polynomials, to Eq. (4.33) and obtain Eq. (4.34). Eq. (4.34) is an approximation because the Taylor-Maclaurin series is an infinite sum of terms.
\[ \epsilon_2^u(g^{u1}(Q_{j1}), g^{u2}(Q_{j2}), \ldots, g^{uD}(Q_{jD})) \approx C_{uj} \]
\[ + \sum_{f=1}^{D} A_f^u g^{uf}(Q_{jf}) + \sum_{f=1}^{D} B_f^u g^{uf}(Q_{jf})^2 + \sum_{f=1}^{D} C_f^u g^{uf}(Q_{jf})^3 + \ldots \]
\[ + \sum_{f=1}^{D} \sum_{f'=1}^{D} A_{ff'}^u g^{uf}(Q_{jf}) g^{uf'}(Q_{jf'}) + \sum_{f=1}^{D} \sum_{f'=1}^{D} B_{ff'}^u g^{uf}(Q_{jf})^2 g^{uf'}(Q_{jf'}) + \ldots \]
\[ + \sum_{f=1}^{D} \sum_{f'=1}^{D} \sum_{f''=1}^{D} A_{f'f''}^u g^{uf}(Q_{jf}) g^{uf'}(Q_{jf'}) g^{uf''}(Q_{jf''}) + \ldots \]
\[ + \ldots \] (4.34)

The first line in Eq. 4.34 captures the interactions between a feature with itself, the second line captures the interactions between two item features, and the third line captures the interactions between three item features. The coefficient parameters \( A_f^u, B_f^u, C_f^u, \ldots, A_{ff'}^u, B_{ff'}^u, \ldots \) determine whether an interaction exists, and the strength of the interaction.

If we only use the first terms in the first and second lines to estimate the interactions between features, we obtain Eq. 4.35.

\[ \epsilon_2^u(g^{u1}(Q_{j1}), g^{u2}(Q_{j2}), \ldots, g^{uD}(Q_{jD})) \approx C_{uj} + \sum_{f=1}^{D} A_f^u g^{uf}(Q_{jf}) + \sum_{f=1}^{D} \sum_{f'=1}^{D} A_{ff'}^u g^{uf}(Q_{jf}) g^{uf'}(Q_{jf'}) \] (4.35)

If we let \( A_{ff'}^u = Y_{ff'} \) and \( A_f^u = (\sum_{i \in I_u} y_{if} + P_{uf}) \), Eq. 4.33 becomes Eq. 4.35.

Therefore, we have proved that Eq. 4.23 can reasonably be used to estimate the user preferences over items in recommender systems. \( A_{ff'}^u = Y_{ff'} \) means that the interactions between item features are captured by the same matrix \( Y \) for all users. We may lose some generality due to this assumption, but it is not practical to train one separate interaction matrix \( Y \) for each user, since it would significantly increase the computation and amount of memory needed to store all the matrices. However, assuming equal interaction matrices for all the users is a good compromise between accuracy, and memory and computational complexity.

4.2.5 Illustrative example

In order to demonstrate how CFVSVD captures conditional preferences over feature values, in this section we give a simple example. We assume that five users rate a total of four
Chapter 4. Modelling Conditional Feature Value Preferences

<table>
<thead>
<tr>
<th></th>
<th>Movie 1</th>
<th>Movie 2</th>
<th>Movie 3</th>
<th>Movie 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>5.00</td>
<td>3.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>User 2</td>
<td>0.00</td>
<td>4.00</td>
<td>2.00</td>
<td>4.00</td>
</tr>
<tr>
<td>User 3</td>
<td>3.00</td>
<td>5.00</td>
<td>0.00</td>
<td>2.00</td>
</tr>
<tr>
<td>User 4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.00</td>
</tr>
<tr>
<td>User 5</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.1: Example matrix $R$ with 5 users rating 4 movies. 0.00 denotes missing rating.

movies on a scale between 1 and 5. The goal of preference modelling is to estimate the missing ratings denoted by zero in the ratings matrix $R$ in Table 4.1.

Using CFVSVD, the users are modelled using the user-specific matrices $P$, $W$, $Z$, the item-specific matrix $Q$, and the interaction matrix $Y$ (for the sake of simplicity, in this example we ignore the implicit feedback matrix $y$).

Figure 4.4: Matrices $P$, $W$, $Z$, $Q$ and $Y$, and vectors $bu$ and $bi$ for the illustrative example.
The columns in the matrices $P$, $Q$, $W$, $Z$ and $Y$ in Figure 4.4 illustrate that we model the user preferences using 6 features. The values in matrix $P$ show the importance of each feature for each user. The second feature is clearly the most important for the first user in matrix $P$, as it has the highest value for this user. The values in matrix $Q$ show how much of a particular feature each item has.

Assuming that a large part of the first factor (represented by the first column in matrix $Q$) is comprised of the price feature, these values in matrix $Q$ would mean that in general, users tend to appreciate price most in the second item.

The gradient matrix $W$ and the intercept matrix $Z$ in combination with matrix $Q$ model the users’ preferences as linear functions over the item features in $Q$. Strictly speaking, matrix $Z$ by itself depicts how attractive an item feature with the value of zero would be. Matrix $W$ shows how much the user preference over an item feature value increases/decreases, as its value increases. If feature $f$ is price, $W_{uf} = -2$ means that the user $i$’s preference over this feature decreases with a slope of -2.

The functions for user preferences over feature values are extracted from the matrices $P$, $W$, and $Z$ and are illustrated in Figure 4.5. In Figure 4.5, user 4’s preferences over the values for the first feature decreases in a sharp descent, while the preferences of user five increase clearly. Assuming that this feature mostly represents price, this would mean that user 4 prefers cheaper items (for example due to strict budget limitations), while in contrast, user 5 prefers more expensive items. It is clear that such differences between users can reasonably exist, and this example shows how CFVSVD can model such differences if they exist in the rating patterns.

Using the values in the matrices $W$ and $Z$, in Figure 4.5 we plot the individual feature value preference functions for 5 users $(g_{uf}^f(Q_{jf}) = W_{uf}Q_{jf} + Z_{uf})$. Each diagram describes one item feature. The legends in each diagram represent the users $u = 1, u = 2, ..., u = 5$. In this example, MAE and RMSE have a value of 0.52.
Chapter 4. Modelling Conditional Feature Value Preferences

Figure 4.5: The individual preference functions for each one of 6 features in the example given in the example

The values in matrix $Y$ show the dependencies between item features. For example, the small value in first row and second column, in matrix $Y$ in Figure 4.4e shows that there is a weak dependency between the first feature and the second feature, while the large value in the first row and sixth column shows a strong dependency between the first feature and the sixth feature.

The estimated bias values for users ($b_u$) and items ($b_i$) show that the first and fifth users are very biased in their ratings, meaning that they give higher and lower than average ratings respectively. Similarly, we can see that the first and third items have very biased ratings. The first item tends to receive higher ratings than the average, while the third item tends to receive lower ratings.

4.3 Experiments

In order to evaluate the effectiveness of the methods CFVSV, we conducted a series of experiments on two benchmark recommendation datasets. To implement the proposed method, we use LibRec [29] which is an open source java library for recommender systems.
4.3.1 Datasets

In order to test the performance of the proposed model, we use two popular datasets, *CiaoDVD* [30], and *LastFM* [100] (Section 1.5.3).

We compared the results of CFVSVSD with the recommendation quality of its baseline, *SVD++* [51] which similar to CFVSVSD captures bias values, but unlike CFVSVSD, does not include conditional preferences over feature values. We also compare the model with conditional preference with the case where the conditional preferences are not trained (FVSVD).

In the experiments, 80%, 60%, and 40% of the ratings are randomly chosen for training and respectively the remaining 20%, 40%, and 60% are used for validation. Each model training and testing is repeated 30 times to eliminate the randomness in the results and therefore to assure that the results are more reliable.

The datasets were used in their entirety to predict ratings for all users. We also included predictions for cold start users alone. Cold start users have given 5 ratings or fewer, therefore little is known about their preferences compared to other users. The models used include social interactions, which rely on the definition of friends and the correlations of a user’s tastes and that of her friends. The estimates for cold start users also rely on the general preference of the public for an item (matrix $Q$).

Due to the over-fitting problem, the accuracy of iterative models keeps improving for a certain number of iterations, after which it starts to degrade. Therefore to compare the models, we recorded the best accuracy values achieved by each model across the iterations, rather than using the final values at the end of a fixed number of iterations. We believe that this approach results in a fairer comparison. A t-test was conducted to establish the significance of the differences.

4.3.2 Results

CFVSVSD comprises both the conditional preferences and preferences over feature values, whereas FVSVD only models preferences over feature values. To establish which aspect contributes to the quality, we compare both of these models with SVD++, an existing technique which models neither of these aspects, but is otherwise equivalent.

We compare the results in terms of the RMSE and MAE, on the complete set of users and cold start users only. We also vary the proportion of training and testing entries. The results for the CiaoDVD dataset in Table 4.3 show that in general, CFVSVSD achieves consistently better results than FVSVD and SVD++, even for cold start users. We can
Table 4.3: Comparison of CFSVD, FVSVD and SVD++ on the CiaoDVD dataset. ALL/CS decides whether all or only cold start users are considered. The first number in the split column is the percentage of training, the latter number the percentage of test data items. The best-performing algorithm is shown in bold

also see that the differences are larger when measuring by MAE. It is also apparent that a larger training set improves the accuracy, and that including both conditional preferences and preferences over feature value is beneficial. Nonetheless, FVSVD achieves the same quality on the RMSE as CFVSVD in the 80/20 instance with all users, whereas there is a slight difference in the MAE values of the same case. The RMSE for cold start users with a 60/40 split even concludes in favour of SVD++, indicating that capturing additional aspects adds noise and deteriorates the model, although the difference is confined to the fourth decimal.

It is likely that the usefulness of modelling additional aspects in rating data depends on whether the data contains these aspects. The comparison using the LastFM dataset in Table 4.4 illustrates this notion. FVSVD consistently outperforms CFVSVD, suggesting that there are no interdependencies between features to capture.

However, since FVSVD mostly outperforms SVD++, the dataset appears to contain differences in user opinions on the values of features, such as one user preferring low price and another mid-range. SVD++ only outperforms FVSVD when the error is measured using RMSE with cold start users, and when the training set is smaller than 80%. In the case of cold start users, ratings are scarce. When small training sets are used, there is even less information to draw on, and the advantage of modelling additional aspects disap-
### Table 4.4: Comparison of CFSVD, FVSVD and SVD++ on the LastFM dataset. ALL/CS decides whether all or only cold start users are considered. The first number in the split column is the percentage of training, the latter number the percentage of test data items. The best-performing algorithm is shown in bold.

<table>
<thead>
<tr>
<th>Measure</th>
<th>ALL/CS</th>
<th>Split</th>
<th>CFVSVD</th>
<th>FVSVD</th>
<th>SVD++</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>40/60</td>
<td>0.5448</td>
<td><strong>0.5447</strong></td>
<td>0.5535</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>40/60</td>
<td>0.4670</td>
<td><strong>0.4644</strong></td>
<td>0.5734</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>60/40</td>
<td>0.5418</td>
<td><strong>0.5416</strong></td>
<td>0.5507</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>60/40</td>
<td>0.4656</td>
<td><strong>0.4528</strong></td>
<td>0.5648</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>80/20</td>
<td>0.5406</td>
<td><strong>0.5405</strong></td>
<td>0.5496</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>80/20</td>
<td>0.3595</td>
<td><strong>0.3560</strong></td>
<td>0.4312</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>40/60</td>
<td>0.6686</td>
<td><strong>0.6682</strong></td>
<td>0.6688</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>40/60</td>
<td>0.6586</td>
<td>0.6507</td>
<td><strong>0.6395</strong></td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>60/40</td>
<td>0.6664</td>
<td><strong>0.6660</strong></td>
<td>0.6665</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>60/40</td>
<td>0.6851</td>
<td>0.6633</td>
<td><strong>0.6262</strong></td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>80/20</td>
<td>0.6659</td>
<td><strong>0.6656</strong></td>
<td>0.6660</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>80/20</td>
<td>0.5759</td>
<td><strong>0.5704</strong></td>
<td>0.5953</td>
</tr>
</tbody>
</table>

Interestingly, when the training set is 80%, both CFVSVD and FVSVD outperform SVD++ by a large margin.

We tested the hypothesis of significant difference in the accuracies between pairs of algorithms. Table 4.7 confirms that using MAE, both CFVSVD and FVSVD are significantly better than SVD++ regardless of dataset. The only difference is that RMSE squares the error values, whereas MAE does not. Appraising the differences by the RMSE, we observe a few instances where SVD++ performs insignificantly better than CFVSVD or FVSVD.

The comparison between CFVSVD and FVSVD has more mixed results regarding significance, especially in the cases of the cold start users. As observed in Tables 4.3 and 4.4, generally CFVSVD performs better on the CiaoDVD dataset, whereas LastFM reverses the result. In general, predictions for cold start users entail more uncertainty, hence the models perform more similarly and the differences are often not significant on the 99% level, or even on the 95% level.
<table>
<thead>
<tr>
<th>Measure</th>
<th>Split</th>
<th>ALL/CS</th>
<th>CiaoDVD</th>
<th>LastFM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Diff.</td>
<td>Sig.</td>
<td>Diff.</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0074</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0116</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0077</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>60/40</td>
<td>-0.0109</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>80/20</td>
<td>-0.0078</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0095</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0011</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0009</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0006</td>
<td>0.0001</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>60/40</td>
<td>0.0001</td>
<td>0.8844</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>80/20</td>
<td>-0.0015</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0009</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.5: Significances of difference measured using the t-test for pairwise comparisons of models. ALL/CS decides whether all or only cold start users are considered. The first number in the split column is the percentage of training, the latter number the percentage of test data items. If the difference value is negative, it is in favour of the first algorithm, otherwise the second. Bolding indicates significant difference on the 99% level (Sig. < 0.01).

<table>
<thead>
<tr>
<th>Measure</th>
<th>Split</th>
<th>ALL/CS</th>
<th>CiaoDVD</th>
<th>LastFM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Diff.</td>
<td>Sig.</td>
<td>Diff.</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0060</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0085</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0066</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>60/40</td>
<td>-0.0084</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>80/20</td>
<td>-0.0069</td>
<td>0.0000</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0078</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0009</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0001</td>
<td>0.6906</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0005</td>
<td>0.0002</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>60/40</td>
<td>0.0007</td>
<td>0.0038</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>80/20</td>
<td>-0.0015</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0005</td>
<td>0.0328</td>
</tr>
</tbody>
</table>

Table 4.6: Significances of difference measured using the t-test for pairwise comparisons of models. ALL/CS decides whether all or only cold start users are considered. The first number in the split column is the percentage of training, the latter number the percentage of test data items. If the difference value is negative, it is in favour of the first algorithm, otherwise the second. Bolding indicates significant difference on the 99% level (Sig. < 0.01).
Table 4.7: Significances of difference measured using the t-test for pairwise comparisons of models. ALL/CS decides whether all or only cold start users are considered. The first number in the split column is the percentage of training, the latter number the percentage of test data items. If the difference value is negative, it is in favour of the first algorithm, otherwise the second. Bolding indicates significant difference on the 99% level (Sig. < 0.01)

<table>
<thead>
<tr>
<th>CFVSD - FVSD</th>
<th>Measure</th>
<th>Split</th>
<th>ALL/CS</th>
<th>MAE CiaoDVD</th>
<th>LastFM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diff.</td>
<td>Sig.</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0014</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0031</td>
<td>0.0000</td>
<td>0.0026</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0011</td>
<td>0.0000</td>
<td>0.0002</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>60/40</td>
<td>-0.0025</td>
<td>0.0000</td>
<td>0.0128</td>
</tr>
<tr>
<td>MAE</td>
<td>ALL</td>
<td>80/20</td>
<td>-0.0009</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>MAE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0017</td>
<td>0.0000</td>
<td>0.0035</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>40/60</td>
<td>-0.0002</td>
<td>0.0499</td>
<td>0.0004</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>40/60</td>
<td>-0.0008</td>
<td>0.0013</td>
<td>0.0079</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>60/40</td>
<td>-0.0001</td>
<td>0.4091</td>
<td>0.0004</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>60/40</td>
<td>-0.0006</td>
<td>0.0110</td>
<td>0.0218</td>
</tr>
<tr>
<td>RMSE</td>
<td>ALL</td>
<td>80/20</td>
<td>0.0000</td>
<td>0.9521</td>
<td>0.0003</td>
</tr>
<tr>
<td>RMSE</td>
<td>CS</td>
<td>80/20</td>
<td>-0.0004</td>
<td>0.1580</td>
<td>0.0055</td>
</tr>
</tbody>
</table>

4.4 Summary of key points

1. Current latent factor models assume that all values of latent features are equally preferable for the users, and hence, cannot capture preferences over feature values. Furthermore, conditional dependencies between features are mostly ignored.

2. Ignoring this aspect potentially degrades the accuracy of these models, because the same preference values are trained for all users.

3. A possible way of addressing this problem could be by assuming that these preferences follow linear patterns and introducing additional matrices to capture these preferences and the dependencies.

4. This method can partially capture the differences between preferences over feature values without considerably increasing the computational complexity.

5. Experiments on two datasets show that capturing these preferences using linear function that is proposed significantly improves the accuracy.
4.5 Conclusion

In this chapter, we addressed the problem of improving the recommendation accuracy of the state of the art models by learning user conditional preferences over feature values in latent factor models. In order to tackle this problem, we proposed an extension to probabilistic matrix factorisation that captures conditional preferences over feature values. In CFVSVD, two additional matrices (feature gradient matrix, $W$ and feature intercept matrix, $Z$) model a linear function for each user which captures the user’s appreciation of the values of a latent feature. As in PMF and SVD++, the prediction error is minimised by using stochastic gradient descent. The interaction matrix $Y$ was added to capture the interactions between features. Mathematical proofs show that the function that models the aspects of the data including conditional feature value preferences is suitable for the estimation of unknown ratings. The proofs also show that the classic methods based on PMF are unable to capture the differences between users in preferring item feature values. CFVSVD can capture these differences in part while balancing accuracy, computational complexity and memory requirements.

The experiments on two benchmark datasets, CiaoDVD and LastFM showed that in both datasets and for all cases and both for all users and cold-start users, CFVSVD achieved significantly higher accuracies in terms of MAE and RMSE than SVD++. This confirmed the benefit and significance of incorporating conditional preferences over feature values into latent factor models.

In the case of LastFM, a model with includes preferences over feature values but not the interactions between features (FVSVD) outperformed CFVSVD in all cases by a small but sometimes significant margin. This observation raises the question whether the use of CFVSVD can be recommended for all datasets or whether the introduction of the dependency aspect can sometimes be detrimental. It seems safe to say that in the case of LastFM, considering all users, the differences are very small, and in any case considerably smaller than those between CFVSVD and SVD++. It therefore seems safe to recommend the use of CFVSVD over SVD++ and FVSVD. However, it seems that feature value preferences and dependencies between features are sometimes present, but not always. This sparked the idea of adding a functionality to the model, so that every aspect can be arbitrarily switched off or on. In the following chapters, the aspects are added to the model in a way that they could be removed by switching off their respective component.
Chapter 5

Modelling Socially-Influenced Preferences

5.1 Introduction

According to peer selection (homophily) and social influence in the social sciences, users tend to be connected with users who have similar attributes, and social influence further enhances the similarity of members of the same social network. Therefore, the users’ tastes and preferences become similar. Incorporating social influence into recommender systems has proved to be beneficial in improving recommendation performance. The additional data provided by the social context can help alleviate the cold-start and data sparsity problems. In reality, we always turn to friends we trust for movie, music or book recommendations, and our tastes can be easily influenced by the friends we interact with. Although incorporating social influence into latent factor models has been addressed before by many researchers, the influence of social relationships on the constituting aspects of user preferences has been ignored. However, the users’ preferences over features as well as the feature values can be subject to social influence.

To incorporate the influence of friends in the social context on preferences, Guo et al. have proposed TrustSVD. This method builds on SVD++ and incorporates the explicit and implicit influence of ratings as well as trust between users in a social network into matrix factorisation. Comprehensive experimental results have shown that TrustSVD outperforms both trust- and rating-based methods in prediction accuracy. However, TrustSVD does not include conditional preferences over feature values that we considered in Chapter 4. In this chapter, we extend FVSVD (proposed in Chapter 4) and TrustSVD, and add the preferences over feature values and propose TFVSVD. We also
propose an extension to TFVSVD, called CTFVSVD, which takes the conditional feature
dependencies into consideration. Therefore, the extensions proposed in this chapter build
on both TrustSVD and FVSVD which build on SVD++. The difference between TrustSVD
and CTFVSVD is that TrustSVD does not include preferences over feature values, while
CTFVSVD does not include the social aspect. In these extensions, we add additional
matrices that capture the feature value preferences, which are related to the values of
social influence matrix. The explicit influence of social relationships on each one of the
preference aspects (feature preferences and feature value preferences) is modelled. Figure
5.1 shows how these extensions build on one another.

Figure 5.1: The relationship between the models

Therefore, the major research question that we are interested in in this chapter is,
how can we efficiently model preferences given the differences between users in preferring
item feature values, and their social connections with each other in social networks, and
how much improvement would incorporating such information make? The current work
proposes a feature-aware latent factor model based on matrix factorisation to address these
issues.

The advantage of TFVSVD compared with the existing methods is that it considers
the differences between users in preferring item feature values. Unlike the existing methods
that only apply social influence to feature preferences, this method is the first that also
models the influence of friends on feature value preferences. The only disadvantage of these
models is that the new extensions slightly increase the computational time of CTFVSVD.

The rest of the chapter is organised as follows: In Section 5.1.1 we provide a motivating
example that shows the problems with latent factor models based on PMF in capturing
social influence. Then in sections 5.2 and 5.3 we introduce the proposed models, TFVSVD and CTFVSVD to overcome the challenge of learning conditional and non-conditional socially-influenced preferences over feature values in PMF. In Section 5.4 we first explain the experimental setup, and then report on the results of the proposed model. Finally we conclude the chapter in Section 5.7, by summarising the main findings of this work.

5.1.1 Motivating example

In this section, we continue with the example provided in Section 4.1 and show the limitations of the existing latent factor models based on matrix factorisation, in capturing conditional preferences over feature values that are subject to social influence. In that example, we assumed that the matrices $P$ and $Q$ model the preferences of users $u_1$ and $u_2$ over items $i_1, i_2, i_3, i_4$.

\[
\begin{array}{cc|cc}
 & i_1 & i_2 & i_3 & i_4 \\
 u_1 & R_{11} & R_{12} & R_{13} & R_{14} \\
 u_2 & R_{21} & R_{22} & R_{23} & R_{24} \\
\end{array}
\]

\[
\begin{array}{cc|cc|cc|cc}
 & f_1 & f_2 & i_1 & e & g & i_1 & i_2 & i_3 & i_4 \\
 u_1 & a & b & i_2 & f & g & u_1 & ea + gb & fa + gb & ea + hb & fa + hb \\
 u_2 & c & d & i_3 & e & h & u_2 & ec + gd & fc + gd & ec + hd & fc + hd \\
 i_4 & f & h & & & & & & & \\
\end{array}
\]

Let us assume that these two users have the preferences defined by Eqs. 5.1 and 5.2, which can be modelled using the two factorised matrices $P$ and $Q$, using classic matrix factorisation.

\[u_1 : R_{11} > R_{12}, R_{13} > R_{14}\]  \hspace{1cm} (5.1)

\[u_2 : R_{21} < R_{22}, R_{23} > R_{24}\]  \hspace{1cm} (5.2)

Let us assume that user $u_1$ and $u_2$ become friends and due to the social influence, user $u_1$ changes his preference over item $i_3$ and $i_4$, and prefers item $i_4$ over item $i_3$. This would result in the preference order in Eqs. 4.5 and 4.5 which as we concluded, cannot be modelled using the methods based on classic matrix factorisation. This example shows the limitations of methods based on classic matrix factorisation, in modelling conditional
preferences over feature values that are subject to change by social interactions. Therefore, we need to develop a method that can capture such preferences from the user-item ratings and social network data.

To the best of our knowledge, the current work is the first attempt at modeling socially-influenced conditional preferences over feature values in latent factor models. Unlike similar models in which the feature value preferences are explicitly extracted from the user reviews, we extract the feature value preferences implicitly through a latent factor model. Due to the hidden nature of features in latent factor models, these models are also able to incorporate the user preferences over less obvious features, and even completely uninterpretable features \[52\]. Therefore, the preferences over the values of such features can also be captured CTFVSVD, while they would not be included in user reviews. All latent factor models which consider the influence of friends in user preferences define the user preferences as a vector of weight values for item features. Therefore, the differences between users in preferring item feature values are ignored. However, we speculate that ignoring such differences degrades the accuracy of latent factor models, and the socially-influenced conditional feature and feature value preferences should be captured in a model. The model proposed in this chapter considers the influence that social connections have on the users’ preferences over item features, as well as item feature values while taking into account the dependencies between item features.

5.2 Biased probabilistic matrix factorisation with socially-influenced preferences over feature values

To address the problem of capturing preferences over feature values with trust considerations in matrix factorisation, as we did in Chapter \[4\], we extend FVSVD by adding the matrix \(\omega\) for capturing social influence.

The method proposed here is abbreviated to TFVSVD. The graphical model of TFVSVD is depicted in Figure 5.2. Similar to PMF, in TFVSVD the user preferences are modelled as a Bayesian Network \[18\]. Figure 5.2 shows the topology or the structure of the Bayesian Network for user preferences that are modelled by TFVSVD. Suppose that a social network is represented by a graph \(\mathcal{V} = (\mathcal{V}, \mathcal{E})\), where \(\mathcal{V}\) includes a set of users (nodes) and \(\mathcal{E}\) represents the trust relationships among the users (edges). We denote the adjacency matrix by \(T \in \mathbb{R}^{N \times N}\), where \(T_{uv}\) shows the degree to which user \(u\) trusts user \(v\).

In TFVSVD, all aspects of preferences (i.e. preferences over features as well as prefer-
ences over feature values) are assumed to be subject to change by social interactions, and therefore the explicit influence of social relationships on each one of aspects of preferences are modelled. In TrustSVD, the rating given by user \( u \) to item \( j \) is calculated according to Eq. 5.3.

\[
\hat{R}_{uj} = \mu + bu_u + bi_j + \sum_{f=1}^{D} (P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf})Q_{jf}
\]  

(5.3)

According to this equation, the estimated rating is a result of users general taste (\( \mu \)), user bias (\( bu_u \)), item bias (\( bi_j \)), the intrinsic preferences (\( \sum_{f=1}^{D} P_{uf} \)), implicit feedback (\( \sum_{f=1}^{D} |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} Q_{jf} \)), and social influence (\( \sum_{f=1}^{D} |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf} Q_{jf} \)). Therefore, unlike SVD++ in 4.9 which does not model social influence, TrustSVD also captures social influence.

Then the error to be minimised is formulated as Eq. 5.4.

\[
\arg\min_{P,Q,W,Z,\omega, y, bu, bi} [E = \frac{\lambda_P}{2} \sum_{u=1}^{N} \sum_{v \in I_u} I_{uv}(T_{uv} - \hat{T}_{uv})^2 + \frac{1}{2} \sum_{u=1}^{N} \sum_{j=1}^{M} (R_{uj} - \hat{R}_{uj})^2 \\
+ \frac{\lambda_f}{2} \sum_{u=1}^{N} |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} ||P_{ui}||^2_{Frob} + \frac{\lambda_f}{2} \sum_{j=1}^{M} ||Q_{j}||^2_{Frob} \\
+ \frac{\lambda_s}{2} \sum_{u=1}^{N} |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} ||\omega_{ui}||^2_{Frob} + \frac{\lambda_f}{2} \sum_{j=1}^{M} ||Q_{j}||^2_{Frob} \\
+ \frac{\lambda_{bu}}{2} \sum_{u=1}^{N} |I_u|^{-\frac{1}{2}} bu_u^2 + \frac{\lambda_{bi}}{2} \sum_{j=1}^{M} |I_j|^{-\frac{1}{2}} bi_j^2]]
\]  

(5.4)
where \( y \in \mathbb{R}^{M \times D} \) and \( \omega \in \mathbb{R}^{N \times D} \) account for implicit feedback, and implicit trust between users respectively, \( \hat{T}_{uv} = P_u \omega_v^T \), and \( I_u \) is the set of items rated by user \( u \), \( T_u \) is the set of users that user \( u \) trusts, \( T_v^+ \) is the set of users that trust user \( v \), and \( U_j \) is the set of users who have rated item \( j \). According to Eq. 5.3, the user preferences over features are comprised of the explicit feedback extracted from the ratings, the implicit feedback given by user, by choosing a set of items to rate, and the implicit feedback given by users by choosing a set of users to trust. Equation \( \hat{T}_{uv} = P_u \omega_v^T \) means that the explicit trust values in matrix \( T \) are assumed to be estimable by explicit user preferences in matrix \( P \) as well as the implicit social influence matrix \( \omega \). In this section and the following section, we explain how we extend this model by incorporating socially-influenced conditional preferences over feature values.

In TFVSVD, we also assume that the user preferences over an item feature can be formulated with a linear function. In TFVSVD, the user preferences over feature values are captured by a user-dependent factorisation using the function \( W_{uf}Q_{jf} + Z_{uf} \), while in the previous methods, it was captured using \( Q_{jf} \). As we showed in Chapter 4, since \( Q_{jf} \) is an item dependent matrix, it cannot model the different users’ preferences over an item feature value, since all users’ preferences will be combined into the value of an item feature. Therefore, a user-dependent function is required to capture such differences. The simple example in Section 5.3.1.3 shows the preferences for a few users. As Figure 5.2 shows, these matrices have the same dimensions as the user matrix \( P \). According to Figure 5.2, the probabilities of the matrices \( P, Q, W, Z, \omega, y \) and vectors \( bu \) and \( bi \) are dependent on the hyper-parameters \( \sigma_P, \sigma_Q, \sigma_W, \sigma_Z, \sigma_\omega, \sigma_y, \sigma_{bu} \) and \( \sigma_{bi} \) respectively. Likewise, the probability of obtaining the ratings in matrix \( R \) is conditional on the matrices \( P, Q, W, Z, \omega, y \) and vectors \( bu \) and \( bi \). Throughout this thesis, we choose to use capital letters for the matrices that contain explicit information, and non-capital letters for the matrices that hold implicit information. According to the Bayesian Network of the user preferences in Figure 5.2, in the proposed model, the posterior probability of user and item matrices are calculated according to Eq. 5.5, which incorporates user bias \( bu \) and item bias \( bi \) and the preferences over feature values captured by matrices \( W \) and \( Z \), as well as the implicit feedback and trust captured by the matrices \( y \) and \( \omega \).

\[
p(P, Q, W, Z, \omega, y, bu, bi|R, T^{(1)}, T^{(2)}, T^{(3)}, \sigma, \sigma_T, \sigma_P, \sigma_W, \sigma_Q, \sigma_Z, \sigma_\omega, \sigma_y, \sigma_{bu}, \sigma_{bi})
\]

\[
\propto p(R|P, Q, W, Z, \omega, y, bu, bi, \sigma) \times p(T^{(1)}|\omega, P, \sigma_T) \times p(T^{(2)}|\omega, W, \sigma_T) \times p(T^{(3)}|\omega, Z, \sigma_T)
\]

\[
\times p(P|\sigma_P) \times p(P|\sigma_T) \times p(W|\sigma_W) \times p(W|\sigma_T) \times p(Q|\sigma_Q) \times p(Z|\sigma_Z) \times p(Z|\sigma_T)
\]

\[
\times p(bu|\sigma_{bu}) \times p(bi|\sigma_{bi}) \times p(\omega_v|\sigma_T) \times p(y|\sigma_T)
\]

Similar to PMF, we assume that the probabilities in Eq. 5.5 follow a Gaussian distribution. Using this assumption Eq. 5.5 results in Eq. 5.6.
\[ p(P, Q, W, Z, \omega, y, bu, bi, R, T^{(1)}, T^{(2)}, T^{(3)}, \sigma, \sigma_T, \sigma_P, \sigma_w, \sigma_Q, \sigma_z, \sigma_y, \sigma_{bu}, \sigma_{bi}) \]

\[ \approx \prod_{u=1}^{N} \prod_{j \in I_u} N(R_{u,j}, \hat{R}_{u,j}, \sigma^2) \]

\[ \times \prod_{u=1}^{N} \prod_{v \in T_u} N(T_{uv}^{(1)}|\hat{T}_{uv}^{(1)}, \sigma_T^2) \times \prod_{u=1}^{N} \prod_{v \in T_u} N(T_{uv}^{(2)}|\hat{T}_{uv}^{(2)}, \sigma_T^2) \times \prod_{u=1}^{N} \prod_{v \in T_u} N(T_{uv}^{(3)}|\hat{T}_{uv}^{(3)}, \sigma_T^2) \]

\[ \times \prod_{u=1}^{N} N\left(\frac{P_u}{T_u}|0, \sigma^2_{P}\right) \times \prod_{u=1}^{N} N\left(\frac{P_u}{T_u}|0, \sigma^2_{P}\right) \times \prod_{u=1}^{N} N\left(\frac{W_u}{I_u}|0, \sigma^2_{W}\right) \times \prod_{u=1}^{N} N\left(\frac{W_u}{I_u}|0, \sigma^2_{W}\right) \]

\[ \times \prod_{u=1}^{N} N\left(\frac{Z_u}{T_u}|0, \sigma^2_{Z}\right) \times \prod_{u=1}^{N} N\left(\frac{Z_u}{T_u}|0, \sigma^2_{Z}\right) \times \prod_{j=1}^{M} N\left(\frac{b_{ij}}{|U_j|}|0, \sigma^2_{b}\right) \times \prod_{j=1}^{M} N\left(\frac{b_{ij}}{|U_j|}|0, \sigma^2_{b}\right) \times \prod_{i=1}^{M} N\left(\frac{y_i}{|U_i|}|0, \sigma^2_{y}\right) \times \prod_{i=1}^{M} N\left(\frac{\omega_i}{|T_{iu}|}|0, \sigma^2_{\omega}\right) \]

where \( \hat{R}_{u,j}, \hat{T}_{uv}^{(1)}, \hat{T}_{uv}^{(2)}, \hat{T}_{uv}^{(3)} \) are calculated according to Eqs. 5.7 to 5.10.

\[ \hat{R}_{u,j} = \mu + bu_u + bi_j + \sum_{f=1}^{D} (P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf})(W_{uf}Q_{jf} + Z_{uf}) \] (5.7)

Eq. 5.7 means that user \( u \)'s rating over item \( j \) can be estimated by calculating the average rating that the users normally give to the items, as well as the user and item biases and the users' preferences captured by matrices \( P, Q, W, Z, \omega, \) and \( y \). The term \( P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf} \) in Eq. 5.7 means that the importance of an item feature \( f \) for user \( u \) is comprised of the explicit importance value captured through the user-item interaction \( (P_{uf}) \), the implicit feedback that user \( u \) provides for item feature \( f \) by choosing the items in set \( I_u \) \( (|I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if}) \), as well as the implicit feedback that user \( u \) provides by choosing the users in set \( T_u \) to trust \( (|T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf}) \). If user \( u \) has rated fewer items, it probably means that the user has very specific preferences over item features. Therefore the implicit feedback makes a larger contribution. This is denoted by the term \( |I_u|^{-\frac{1}{2}} \) in Eq. 5.7. Similarly, if the user \( u \) trusted fewer users, the implicit feedback given by the matrix \( \omega \) will contribute more. This is denoted by the term \( |T_u|^{-\frac{1}{2}} \).

\[ \hat{T}_{uv}^{(1)} = \sum_{f=1}^{D} P_{uf} \omega_{vf} \] (5.8)
The set of users that user $u$ trusts, $T_u^+$, is the set of users that trust user $v$, and $U_i$ is the set of users who have rated item $i$.

### 5.2.1 Model training of TFVSVD

Using stochastic gradient descent, the gradient values for the model parameters (the matrix entries and bias vector values) are obtained according to Eqs. (5.14) through (5.21).
\[
\frac{\partial E^N}{\partial b_{iu}} = \frac{\partial E^N}{\partial b_{iu}} = e^N_{ui} + \lambda_b |I_u|^{-\frac{1}{2}} b_{iu},
\]
(5.14)
\[
\frac{\partial E^N}{\partial b_{ij}} = \frac{\partial E^N}{\partial b_{ij}} = e^N_{ui} + \lambda_b |U_j|^{-\frac{1}{2}} b_{ij},
\]
(5.15)
\[
\frac{\partial E^N}{\partial P_{uf}} = e^N_{ui} (W_{uf}Q_{jf} + Z_{uf}) + \lambda_T |I_u|^{-\frac{1}{2}} P_{uf},
\]
\[
\frac{\partial E^T}{\partial P_{uf}} = \lambda_T |I_u|^{-\frac{1}{2}} P_{uf} + \lambda_T e^{(1)}_{ui} \omega_{uf}
\]
(5.16)
\[
\frac{\partial E^N}{\partial Q_{jf}} = \frac{\partial E^N}{\partial Q_{jf}} = e^N_{ui} [W_{uf}(P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i)] + \lambda_Q Q_{jf}
\]
(5.17)
\[
\frac{\partial E^N}{\partial W_{uf}} = e^N_{ui} Q_{jf} (P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i) + \lambda_W |I_u|^{-\frac{1}{2}} W_{uf},
\]
(5.18)
\[
\frac{\partial E^T}{\partial W_{uf}} = e^N_{ui} Q_{jf} |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_v + \lambda_T |T_u|^{-\frac{1}{2}} W_{uf} - \lambda_T e^{(2)}_{ui} \omega_{vf}
\]
(5.19)
\[
\forall v \in T_v : \frac{\partial E^N}{\partial \omega_{vf}} = e^N_{ui} |T_u|^{-\frac{1}{2}} (W_{uf} Q_{jf} + Z_{uf}) + \lambda_T |T_v|^{-\frac{1}{2}} \omega_{vf}
\]
(5.20)
\[
\forall i \in I_u : \frac{\partial E^N}{\partial y_{if}} = \frac{\partial E^N}{\partial y_{if}} = e^N_{ui} |I_u|^{-\frac{1}{2}} (W_{uf} Q_{jf} + Z_{uf}) + \lambda_T |U_j|^{-\frac{1}{2}} y_{if}
\]
(5.21)

where \(\gamma\) denotes the learning rate in stochastic gradient descent. \(e^N_{uij}\) is calculated according to Eq. (5.22)

\[
e^N_{uij} = R_{uij} - \tilde{R}^N_{uij}
\]
(5.22)

5.3 Biased probabilistic matrix factorisation with socially-influenced conditional preferences over feature values

In this section, we show how we extend TFVSVD proposed in Section 5.2 to incorporate the conditional preferences. We adopt a quadratic polynomial function (5.1) to capture the conditional preferences, and incorporate it into the model proposed in the previous section.
The proposed method is abbreviated as CTFVSVD. In CTFVSVD, the estimated rating of the item $j$ given by user $u$ is calculated based on Eq. 5.23.

$$
\hat{R}_{uj}^C = \mu + bu + bi + \sum_{f=1}^{D} \left( P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_{if} + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{uv} (W_{uf} Q_{jf} + Z_{uf}) + \sum_{f'=1}^{D} (\sum_{f=1}^{D} (W_{uf} Q_{jf} + Z_{uf}) Y_{jj'} (W_{uf'} Q_{jf'} + Z_{uf'}) \right)
$$

(5.23)

This equation includes all the terms in Eq. 5.7, but it also includes one the last term that captures the interactions between features.

The minimisation problem in CTFVSVD is obtained according to Eq. 5.25.

$$
E_R^C = E_R^N (\hat{R}_{uj}^C) + \frac{\lambda_Y}{2} \sum_{i=1}^{D} \|Y_i\|^2_{Frob}
$$

(5.24)

$$
\arg\min_{P,Q,W,Z,Y,\omega,\mu,b_u,b_i} [E^C = E_R^C + E_T]
$$

(5.25)

where $\lambda_Y = \frac{\sigma_Y^2}{\sigma_Y^2}$ is the hyper-parameter for the interaction matrix $Y$. The interaction matrix $Y \in \mathbb{R}^{D \times D}$ in Eq. 5.24 is a symmetrical square matrix and added to capture the dependencies or interactions between item features. The Bayesian network for CTFVSVD is depicted in Figure 5.3. As we can observe, Eqs. 5.26 to 5.28 would simply transform to Eqs. 5.17 to 5.19 if all values in matrix $Y$ were set to zero. Restricting matrix $Y$ entries to zero would mean that there are no conditional dependencies between features in user preferences. When all the entries in gradient matrix $W$ are fixed at one and the entries in intercept matrix $Z$ are fixed at zero, the proposed TFVSVD method is simply reduced to the TrustSVD [31].

### 5.3.1 Model training of CTFVSVD

Similar to TFVSVD, the user and item biases in CTFVSVD are also updated according to gradient values in Eqs. 5.14 and 5.15. However, in CTFVSVD, the model parameters are instead updated according to the gradient values in Eqs. 5.26 through 5.30.
Figure 5.3: Bayesian network of CTFVSVD

\[ \frac{\partial E^C}{\partial Q_{jf}} = \frac{\partial E^C}{\partial Q_{jf}} = e_{uj}^C[W_{uf}(P_u + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i) + 2W_{uf} \sum_{f' = 1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'})Y_{f'}] + \lambda_Q Q_{jf} \]  

(5.26)

\[ \frac{\partial E^C}{\partial W_{uf}} = e_{uj}^C[Q_{jf}(P_u + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i) + 2Q_{jf} \sum_{f' = 1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'})Y_{f'}] + \lambda_W |I_u|^{-\frac{1}{2}} W_{uf} \]  

\[ \frac{\partial E^T}{\partial W_{uf}} = e_{uj}^CQ_{jf}[T_u |^{-\frac{1}{2}} \sum_{v \in I_u} \omega_v + \lambda_T |T_u|^{-\frac{1}{2}} W_{uf} - \lambda_T e_{uv}^{(2)} \omega_{uf}] \]  

(5.27)

\[ \frac{\partial E^C}{\partial Z_{uf}} = e_{uj}^C[(P_u + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i) + 2 \sum_{f' = 1}^{D} (W_{uf'}Q_{jf'} + Z_{uf'})Y_{f'}] + \lambda_Z |I_u|^{-\frac{1}{2}} Z_{uf} \]  

(5.28)

\[ \frac{\partial E^T}{\partial Z_{uf}} = e_{uj}^C[T_u |^{-\frac{1}{2}} \sum_{v \in I_u} \omega_v + \lambda_T |T_u|^{-\frac{1}{2}} Z_{uf} + \lambda_T e_{uv}^{(3)} \omega_{uf}] \]  

\[ \frac{\partial E^C}{\partial Y_{f'}} = \frac{\partial E^C}{\partial Y_{f'}} = e_{uj}^N(W_{uf}Q_{jf} + Z_{uf})(W_{uf'}Q_{jf'} + Z_{uf'}) - \lambda_Y Y_{f'} \]  

(5.29)

(5.30)

5.3.1.1 CTFVSVD Algorithm

The algorithm for CTFVSVD is given in Algorithm 3. This algorithm describes the details of a gradient descent method that is used to train the model parameters \( (bu, bi, P, Q, W, Z, Y, \omega, y, \text{ and } Y) \). The algorithm receives the model hyper-parameters \( (\lambda_{bu}, \lambda_{bi}, \lambda_P, \lambda_Q, \lambda_W, \lambda_Z, \lambda_Y, \gamma_Y) \) and learning rates \( (\gamma_{bu}, \gamma_{bi}, \gamma_P, \gamma_Q, \gamma_W, \gamma_Z, \gamma_y, \gamma_W, \gamma_Y) \) as input,
and as output trains the model parameters according to the Bayesian approach described in Section 5.3. It first starts with initialising the model parameters. The matrices $P$, $Q$, $y$, and $\omega$ and user and item bias vectors ($bu$ and $bi$) are initialised using a Gaussian distribution with a mean of zero and a standard deviation of 1. However, the new matrices $W$, $Z$, and $Y$ are initialised to 1, 0, and 0 values respectively. As mentioned before, when all values in matrix $W$ are fixed at 1 and all the values at matrices $Z$ and $Y$ are fixed at zero, CTFVSVD simplifies to TrustSVD. By initialising the matrix $W$ entries with 1 and matrix $Z$ and $Y$ entries with zero, the algorithm starts the search process at the same starting point as TFVSVD. Therefore, the update rules in lines 17-22 enable the algorithm to reach more solutions the solution space, to find the possible conditional dependencies between the features and the differences between users in preferring item feature values.

The values of the learning rates $\mu_{bu}$, $\mu_{bi}$, $\mu_P$, $\mu_Q$, $\mu_W$, $\mu_Z$, $\mu_y$, $\mu_\omega$, $\mu_Y$ are adjusted depending on whether the error has increased or decreased. If the error value in the current iteration decreases, it means that the search is correctly directed towards a local/global minimum. Therefore, the learning rate is increased to fasten the movements towards the minimum. Otherwise if the error increases, it means that the search is not going on in the right direction, therefore the learning rate is decreased to enable more subtle movements. The model parameters are stochastically updated until a maximum number of iterations is reached (line 5). These details are omitted from Algorithm 3 for the sake of simplicity.

5.3.1.2 Computational complexity analysis

As can be seen in Algorithm 3, CTFVSVD is mainly comprised of two for loops. The first loop in line 9 iterates over the non-zero ratings in the ratings matrix $R$, and the second loop iterates over the non-zero trust relations in the adjacency matrix $T$. Let $|R|$ and $|T|$ denote the number of non-zero entries in the ratings matrix $R$ and adjacency matrix $T$ respectively. Then:

- The number of repetitions to calculate the estimated ratings ($\hat{R}$) in loop 9 is $D^2 \times |R| + D \times \sum_{u=1}^{N} |I_u|^2 + D \times \sum_{u=1}^{N} |I_u| \times |T_u|$.
- The number of repetitions to update user and item bias values ($bu$ and $bi$) in lines 13 and 14 is $2 \times |R|$.
- The number of repetitions needed to update the parameters $P$ in line 17 is $D \times |R|$.
- The number of repetitions needed to update the parameters $Q$, $W$, and $Z$ in line 17 is $3 \times D \times (\sum_{u=1}^{N} |I_u|^2 + \sum_{u=1}^{N} |I_u| \times |T_u| + D)$. 


Algorithm 3 CTFVSVD

1: void TrainModel($\mathbf{b}_u$, $\mathbf{b}_i$, $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$, $\lambda_y$, MaxIter, $\tau_b$, $\tau_i$, $\gamma_P$, $\gamma_Q$, $\gamma_W$, $\gamma_Z$, $\gamma_y$, $\gamma_y$)
2: {
3:  initModel();
4:  $l = 1$;
5:  for $l \leq$ MaxIter do
6:      Matrix $P^l$; Matrix $\omega^l_1$; Matrix $W^l$; Matrix $Z^l$;
7:      $u \leftarrow 1$;
8:      — Calculating intrinsic updates —
9:      for $u \leq N$ do
10:         $j \leftarrow l$;
11:         for $j \leq M$ do
12:             if $j \in I_u$ then
13:                 Update $\mathbf{b}_u$ according to Eq. 5.14
14:                 Update $\mathbf{b}_i$ according to Eq. 5.15
15:             $f \leftarrow 1$;
16:             for $f \leq D$ do
17:                 Update $P^l_{uf}$, $Q^l_{uf}$, $W^l_{uf}$, and $Z^l_{uf}$ according to Eqs. 5.16, 5.20 using $E^{l}_M$.
18:             $v_i \in I_u$ : Update $y^l_{uf}$ according to Eq. 5.21 using $E^{l}_R$.
19:             $f' \leftarrow f + 1$;
20:         end for
21:         for $f' \leq D$ do
22:             Update $Y^l_{f'u}$ and $Y^l_{f'u}$ according to Eqs. 5.22 and 5.30
23:             $f' \leftarrow f' + 1$;
24:         end for
25:         $f \leftarrow f + 1$;
26:     end for
27:     $j \leftarrow j + 1$;
28: end for
29: $u \leftarrow u + 1$;
30: end for
31: — Calculating social updates —
32: for $u \leq N$ do
33:     $j \leftarrow l$;
34:     for $j \leq N$ do
35:         if $v \in T_u$ then
36:             Update $P^l_{uv}$, $W^l_{uv}$, and $Z^l_{uv}$ according to Eqs. 5.16, 5.20 using $E^{l}_T$.
37:         $v \in T_u$ : Update $y^l_{uv}$ according to Eq. 5.21 using $E^{l}_R$.
38:         $f \leftarrow f + 1$;
39:     end for
40: $j \leftarrow j + 1$;
41: end for
42: $u \leftarrow u + 1$;
43: end for
44: — Updating model parameters —
45: $\forall u, f : P^l_{uf} \leftarrow -\gamma_P \times P^l_{uf}$
46: $\forall u, f : W^l_{uf} \leftarrow -\gamma_W \times W^l_{uf}$
47: $\forall u, f : Z^l_{uf} \leftarrow -\gamma_Z \times Z^l_{uf}$
48: $\forall u, f : \omega^l_{uf} \leftarrow -\gamma_y \times \omega^l_{uf}$
49: $l \leftarrow l + 1$;
50: end for
51: } void initModel()
52: {
53:  initMean $\leftarrow \text{initMean}(0);$. initStd $\leftarrow \text{initMean}(1)$;
54:  $\mathbf{b}_u$ $\leftarrow \text{initMean, initStd });$. $\mathbf{b}_i$ $\leftarrow \text{initMean, initStd });$
55:  $P$ $\leftarrow \text{initMean, initStd });$. $Q$ $\leftarrow \text{initMean, initStd });$
56:  $y$ $\leftarrow \text{initMean, initStd });$. $\omega$ $\leftarrow \text{initMean, initStd });$
57:  }
58: $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters. $\lambda_p$, $\lambda_Q$, $\lambda_W$, $\lambda_Z$, $\lambda_y$ are model hyper-parameters (Eq. 5.26). $\mathbf{b}_u$, $\mathbf{b}_i$, $P$, $Q$, $W$, $Z$, $\gamma$, $\gamma$ are the model parameters.
• The number of repetitions needed to update the parameters $\omega$ in line 18 are $D \times \sum_{u=1}^{N}(|I_u| \times |T_u|)$.

• The number of repetitions needed to update the parameters $y$ in line 19 is $D \times \sum_{u=1}^{N}|I_u|^2$.

• The number of repetitions needed to update the dependency matrix $Y$ in line 22 is $D^2 \times |R|$.

• The number of repetitions needed to update the parameters $P, W, Z,$ and $\omega$ in the second loop in line 34 is $4 \times D \times |T|$.

Assuming that on average, each user rates $c$ items, and trusts $k$ users, the computation time can be re-written as Eq. 5.31.

$$\text{Complexity} = O(D^2 \times |R|) + O(D \times |T|) + O(D \times c \times |R|) + O(D \times k \times |T|)$$  \hspace{1cm} (5.31)

Assuming that $c, k \ll N$, we will have:

$$\text{Complexity} = O(D^2 \times |R|) + O(D \times |R|) + O(D \times |T|)$$  \hspace{1cm} (5.32)

Therefore, the overall computation time is linear with respect to the number of observed ratings as well as observed trust statements. Since both ratings matrix and social network matrix are sparse, the algorithm is scalable to the problems with millions of users and items.

5.3.1.3 Illustrative Example

In order to demonstrate how CTFVSVD captures socially-influenced conditional preferences over feature values, in this section we give a simple example. We assume that five users rate a total of four movies on a scale between 1 and 5. The goal of preference modeling is to estimate the missing ratings denoted by zero in the ratings matrix $R$ in Figure 5.5. The red matrices denote the model inputs, and the blue matrices are constructed by the model. As shown in Figure 5.4, the users relate to each other in a social network. The edges in this figure represent the trust relations, and the values on the edges show the strength of the trust relationship. The social network data and the user ratings are the model inputs. Using CTFVSVD, the users are modelled using the user-specific matrices.
The matrices $P$, $W$, $Z$, $\omega$ the item-specific matrix $Q$ and $\gamma$, and the feature-specific interaction matrix $Y$.

As shown in the matrices $P$, $Q$, $W$, $Z$, $\gamma$, $\omega$ and $Y$ in Figure 5.5, we assume that the user preferences can be modelled by only 6 features. The values in matrix $P$ show the importance of each feature for each user. The fifth feature is clearly the most important for the first user in matrix $P$, as it has the highest value for this user. The values in matrix $Q$ show how much of a particular feature each item has.

Assuming that a large part of the first factor (represented by first column in matrix $Q$) is comprised of the amount of action in the movies domain, these values in matrix $Q$ mean that the second item is generally perceived to have a higher level of action than the others. The two matrices $W$, $Z$, along with matrix $Q$ capture the users' preferences over
Chapter 5. Modelling Socially-Influenced Preferences in Recommender Systems

**Figure 5.5**: The matrices $y$, $\omega$, $P$, $Q$, $W$, $Z$, $Y$, $R$, $\hat{R}$, and $T$ and the vectors $bu$ and $bi$ in the illustrative example. Red matrices are model inputs, and blue matrices are model outputs. The arrows show how the values in one matrix affect the values in another matrix.

The "gradient" matrix $W$ and "intercept" matrix $Z$ model individual preference functions of the users over item feature values. Strictly speaking, matrix $Z$ means how favourable an item feature with the value of zero would be. Matrix $W$ shows how much the user preference over an item feature value increases/decreases as its value increases. If feature $f$ represents the amount of action, $W_{uf} = -2$ means that user $u$’s preference for an action movie decreases as the amount of action increases. The values in the matrix $Y$ show the dependencies between item features. For example, the small value in the first row and third column, in matrix $Y$ in the feature dependency matrix in Figure 5.5 shows that there is a weak dependency between the first feature and the third feature, while the large value in the first row and fourth column shows a strong dependency between the first feature and the fourth feature.

The estimated bias values for users ($bu$) and items ($bi$) show that the fourth and fifth users are very biased in their ratings, meaning that they give higher and lower than average ratings respectively. Similarly, we can see that the first and third items have very biased ratings. The first item tends to receive higher ratings than the average, while the third item tends to receive lower ratings. The values for user and item bias vectors $bu$ and $bi$ are updated according to the lines 13 and 14 respectively.
As we can observe in Figure 5.5 using CTFVSVD in real world preferences, different functions are trained to model the user preferences for each feature. For example, user 3’s preference over the values for the third feature increases with a sharp slope, while the preference of user 5 increases slowly.

Assuming that this feature mostly represents the amount of action in the movies domain, this would mean that user 3 has a higher tendency to watch action movies. The same amount of increase in the level of movie action results in a stronger increase in user 3’s preference value than user 5’s, which reveals the differences between these two users in preferring the values for the amount of action feature. The values in the third column of the intercept matrix show that user 4 has a negative preference for a movie with zero amount of action. However, for the user 5, this amount is very small, which means that the user’s preference for a non-action movie is almost zero.

We argue that such differences between different users can reasonably exist (see Chapter 4), and this example shows how CTFVSVD can detect such differences and model them using the rating patterns. For example, the value of the entry at the first column and fourth row in the feature dependency matrix shows that there is a relatively strong dependency between the first and fourth feature.

MAE and RMSE are two standard and popular measures that are used to measure and compare the performance of preference modelling methods in recommender systems.

5.4 Experiments

5.4.1 Comparisons

In order to show the effectiveness of CTFVSVD and TFVSVD, we compared the results with the recommendation quality of some of the classic models, traditional models, popular latent models, and social recommendation models. We included the traditional and classic models in order to investigate how much improvements the additional aspects make compared with the old (traditional) and worst (classic) algorithms that do not capture any aspect at all. The social recommenders on the other hand capture social influence, but not the other aspects. We would also like to investigate the benefits of capturing non-social aspects that are not captured by the current social recommenders. Classic models include User Average which uses the mean value of every user to predict the missing rating values, Item Average which uses the mean value of every item to predict the missing rating values, and Global Average which uses the mean value of all the ratings in training set to predict
the missing ratings. Traditional models involve User KNN and Item KNN, Latent factor models without social aspect are BiasedMF, SVD++, BPMF, NMF, and PMF, and social recommendation models used in the experiments encompass TrustSVD, SoReg, SocialMF, and TrustMF. All the models were compared on the Epinions, Filmtrust, and Movielens datasets.

5.4.2 Setup

In this chapter, 80% of the ratings are randomly chosen for training and the remaining 20% are used for validation. The training and testing of each model is repeated 30 times for statistical significance. The optimal experimental settings for each method are determined either by our experiments or suggested by previous works [31]. For iterative recommendation methods we used 5, 10, 15, and 20 features, because no clear ideal value could be established.

Due to the over-fitting problem, the accuracy of iterative models such as PMF, BiasedMF, and SVD++ improves for a number of iterations, after which it starts to degrade. To compare the models, we recorded the best accuracy values achieved by each model during the iterations, and compared the models based on the recorded values. We believe that this approach results in a fairer comparison of the models than setting the number of iterations to a fixed value, because the models over-fit at different iterations, and using a fixed number of iterations actually prevents us from fairly comparing the models based on their real capacity in uncovering hidden patterns from data. Therefore, the reported results for iterative models here are the best results that they could achieve using the aforementioned parameters. MAE and RMSE measures are used to evaluate and compare the accuracy of the models.

5.5 Results

For better visualisation, Figures 5.7 through 5.9 present a comparison of the most successful methods using the best-performing factor setting. To investigate whether the outperformances are significant, Table 5.6 also includes the results of t-test.

On the Epinions and Filmtrust datasets, SocialMF and TrustMF show similar performance, with one outperforming the other depending on the dataset. However on the Ciao dataset, SocialMF obtains much better results and is competitive with TrustSVD. The methods CTFVSVD and TFVSVD proposed in this chapter provided the best results, with CTFVSVD outperforming TrustSVD on all but Filmtrust’s cold-start users for both
measures and Ciao’s cold-start users for RMSE.

**Table 5.1: Performance comparison of TFVSVD on Epinions dataset for a) all users, and b) cold-start users**

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>Stdev</th>
<th>MAE</th>
<th>Stdev</th>
<th>Ave</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CondTrustFVSVD</td>
<td>0.656015</td>
<td>0.005442</td>
<td>0.731271</td>
<td>0.007785</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustFVSVD</td>
<td>0.597935</td>
<td>0.004532</td>
<td>0.679335</td>
<td>0.007149</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustSV</td>
<td>0.667522</td>
<td>0.006943</td>
<td>0.726577</td>
<td>0.008030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SocialMF</td>
<td>0.631190</td>
<td>0.006187</td>
<td>0.813534</td>
<td>0.007212</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SoReg</td>
<td>0.631088</td>
<td>0.006033</td>
<td>0.813368</td>
<td>0.007256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF</td>
<td>0.636567</td>
<td>0.005919</td>
<td>0.841164</td>
<td>0.007299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.608543</td>
<td>0.004823</td>
<td>0.794821</td>
<td>0.006307</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.608148</td>
<td>0.005628</td>
<td>0.792986</td>
<td>0.006861</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.605267</td>
<td>0.005598</td>
<td>0.805641</td>
<td>0.007584</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.605043</td>
<td>0.007312</td>
<td>0.879459</td>
<td>0.009123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMF</td>
<td>0.623789</td>
<td>0.006442</td>
<td>0.806765</td>
<td>0.008204</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Average</td>
<td>0.627500</td>
<td>0.006932</td>
<td>0.738353</td>
<td>0.008387</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ciao Average</td>
<td>0.619408</td>
<td>0.006102</td>
<td>0.812821</td>
<td>0.007673</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item Average</td>
<td>0.634591</td>
<td>0.004736</td>
<td>0.821630</td>
<td>0.008287</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item Average</td>
<td>0.634591</td>
<td>0.004736</td>
<td>0.821630</td>
<td>0.008287</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Average</td>
<td>0.681525</td>
<td>0.006563</td>
<td>0.917974</td>
<td>0.009064</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.2: Performance comparison of TFVSVD on Filmtrust dataset for c) all users, and d) cold-start users**

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>Stdev</th>
<th>MAE</th>
<th>Stdev</th>
<th>Ave</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CondTrustFVSVD</td>
<td>0.593434</td>
<td>0.001258</td>
<td>0.930314</td>
<td>0.010924</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustFVSVD</td>
<td>0.596195</td>
<td>0.001388</td>
<td>0.954479</td>
<td>0.013492</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustSV</td>
<td>0.674967</td>
<td>0.001151</td>
<td>0.992650</td>
<td>0.012207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustMF</td>
<td>1.850858</td>
<td>0.014210</td>
<td>2.162287</td>
<td>0.013349</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SocialMF</td>
<td>0.647197</td>
<td>0.012346</td>
<td>0.985132</td>
<td>0.013990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SoReg</td>
<td>0.707133</td>
<td>0.010486</td>
<td>0.960702</td>
<td>0.014840</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF</td>
<td>0.596491</td>
<td>0.008124</td>
<td>0.946851</td>
<td>0.008938</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.512413</td>
<td>0.007684</td>
<td>0.928516</td>
<td>0.009388</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.509007</td>
<td>0.010349</td>
<td>1.326536</td>
<td>0.014965</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.659130</td>
<td>0.010328</td>
<td>1.072494</td>
<td>0.014663</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMF</td>
<td>0.445797</td>
<td>0.007919</td>
<td>0.702540</td>
<td>0.011114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Average</td>
<td>1.597528</td>
<td>0.008428</td>
<td>1.809724</td>
<td>0.006448</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Average</td>
<td>1.928401</td>
<td>0.009602</td>
<td>1.586357</td>
<td>0.011564</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item Average</td>
<td>0.525766</td>
<td>0.009698</td>
<td>0.991284</td>
<td>0.010428</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Average</td>
<td>1.822040</td>
<td>0.007672</td>
<td>1.569666</td>
<td>0.009040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.3: Performance comparison of TFVSVD on Ciao dataset for c) all users, and d) cold-start users**

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>Stdev</th>
<th>MAE</th>
<th>Stdev</th>
<th>Ave</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CondTrustFVSVD</td>
<td>0.599358</td>
<td>0.002055</td>
<td>0.887283</td>
<td>0.002043</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustFVSVD</td>
<td>0.595936</td>
<td>0.016416</td>
<td>0.888812</td>
<td>0.017318</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustSV</td>
<td>0.641974</td>
<td>0.011347</td>
<td>0.935374</td>
<td>0.014914</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustMF</td>
<td>1.916313</td>
<td>0.018867</td>
<td>2.283342</td>
<td>0.016855</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SocialMF</td>
<td>0.929455</td>
<td>0.015795</td>
<td>0.000320</td>
<td>0.020762</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SoReg</td>
<td>0.670988</td>
<td>0.011561</td>
<td>1.030547</td>
<td>0.018723</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF</td>
<td>0.544491</td>
<td>0.012438</td>
<td>0.891525</td>
<td>0.016863</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.562631</td>
<td>0.010867</td>
<td>0.891603</td>
<td>0.016292</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.802281</td>
<td>0.016816</td>
<td>1.256372</td>
<td>0.023519</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.623133</td>
<td>0.015900</td>
<td>1.020658</td>
<td>0.026816</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiasedMF++</td>
<td>0.581112</td>
<td>0.013702</td>
<td>0.977797</td>
<td>0.017705</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User KNN</td>
<td>1.685483</td>
<td>0.012793</td>
<td>1.892362</td>
<td>0.019008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item KNN</td>
<td>1.801235</td>
<td>0.011677</td>
<td>1.970393</td>
<td>0.018039</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Average</td>
<td>1.276167</td>
<td>0.020651</td>
<td>1.692460</td>
<td>0.023960</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item Average</td>
<td>0.424652</td>
<td>0.015487</td>
<td>0.896360</td>
<td>0.019009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Average</td>
<td>1.795715</td>
<td>0.014981</td>
<td>1.965200</td>
<td>0.019323</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The performance of CTFVSVD and TFVSVD are closest to those of TrustSV, SVD++.
and BiasedMF on all three datasets, and SocialMF on the Ciao dataset. The results of the t-test in Table 5.6 confirm the significance of the differences for almost all comparisons of the Epinions and Ciao datasets but is more mixed in the case of Filmtrust. Therefore CT-FVSVD and TFVSVD are more helpful on the Epinions and Ciao datasets than Filmtrust dataset. This means that these models benefit more from socially-influenced feature value differences on the Epinions and Ciao datasets than Filmtrust dataset.

Looking into the percentages of rating density and trust density in Table 5.5, we can see that users in the Filmtrust are more active in rating items than engaging in social interactions. However, in the Epinions and Ciao datasets, the users tend to establish social relationships rather than exploring the available items to buy. Therefore, the advantage of using CTFVSVD and TFVSVD becomes more obvious in the Epinions and Ciao datasets which include more socially active users. On the Epinions and Ciao datasets, CTFVSVD and TFVSVD achieve more accurate results by modelling the socially-influenced feature value preferences that are not modelled by the state of the art social recommendation techniques.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Users</th>
<th># of Items</th>
<th># of Ratings</th>
<th># of Trust Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epinions</td>
<td>40,163</td>
<td>139,738</td>
<td>664,824</td>
<td>487,183</td>
</tr>
<tr>
<td>Filmtrust</td>
<td>1,508</td>
<td>2,071</td>
<td>35,497</td>
<td>1,853</td>
</tr>
<tr>
<td>Ciao</td>
<td>2,248</td>
<td>16,861</td>
<td>35,835</td>
<td>57,544</td>
</tr>
</tbody>
</table>

Table 5.4: Number of users, items and trust relationships on the Epinions, Filmtrust, and Ciao datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Rating Density</th>
<th>Trust Density</th>
<th>Avg. Ratings Per User</th>
<th>Avg. Trust Relations Per User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>All (%)</td>
<td>CS (%)</td>
</tr>
<tr>
<td>Epinions</td>
<td>0.011%</td>
<td>0.03%</td>
<td>2.73</td>
<td>16.5531 (0.002%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CS (13.87%)</td>
<td>CS All (%)</td>
</tr>
<tr>
<td>Filmtrust</td>
<td>1.13%</td>
<td>0.08%</td>
<td>0.07</td>
<td>23.5371 (0.006%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CS (11.22%)</td>
<td>CS All (%)</td>
</tr>
<tr>
<td>Ciao</td>
<td>0.09%</td>
<td>1.14%</td>
<td>12.67</td>
<td>15.9408 (0.044%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CS (18.50%)</td>
<td>CS All (%)</td>
</tr>
</tbody>
</table>

Table 5.5: General statistics of the Epinions, Filmtrust, and Ciao datasets

The differences for all users are more significant when the accuracies are measured in terms of the RMSE. This can be explained by the formulation of CTFVSVD and TFVSVD as an optimisation problem. As explained before, these models focus on minimising accuracy using RMSE. However, achieving better MAE values is a secondary goal that is only pursued through minimising RMSE. All the latent factor models and social recommendation models in the experiments are explicitly defined to minimise the RMSE.

The error values in Tables 5.1 through 5.3 show that the models are more accurate on the Filmtrust and Ciao datasets than the Epinions dataset. From Table 5.5 we can clearly
see that the Filmtrust and Ciao datasets feature higher data densities in both the user-item ratings matrix and the trust matrix, as well as greater percentages of average ratings and average trust relations per user. Therefore, the models have access to more information about the users’ preferences and their social ties. Consequently, they generally result in smaller prediction errors in terms of both MAE and RMSE measures on the Filmtrust and Ciao datasets.

The error values in Table 5.1 and Figure 5.7 show that the models usually produce better results on all users than cold-start users on the Epinions dataset. However, as Tables 5.2 and 5.3 and Figures 5.8 and 5.9 show, this is not the case with Filmtrust and Ciao datasets, in which the models generally perform better on cold-start users. This can be particularly attributed to the difference between these datasets in the percentage of trust relations as well as ratings per user for cold-start users. Another remarkable difference between these datasets is that a cold-start user in the Filmtrust and Ciao datasets establishes roughly the same number of relations that any user establishes on average. Moreover, the density of ratings and trust relations for all users are also greater for the Filmtrust and Ciao users than Epinions users. Therefore, cold-start users can even be modelled more accurately, because the correlations of cold-start users with the other users can better be exploited in these datasets.

As Tables 5.1a and 5.1b show, CTFVSVD and TFVSVD achieve better performances than state of the art social recommenders (TrustSVD, TrustMF, SocialMF, and SoReg) on both measures for all users and cold-start users on the Epinions dataset. Table 5.2a shows that these models achieve higher accuracy in terms of RMSE for all users on the Filmtrust dataset. However, in terms of MAE, TrustSVD yields a better accuracy. According to Table 5.2b on the Filmtrust dataset, these models achieve a better performance than all state of the art models in terms of both measures for cold-start users. As Tables 5.3a and 5.3b demonstrate, these models outperform all state of the art models in terms of RMSE measure in the Ciao dataset. These models also outperform all state of the art except Item Average in terms of MAE. These results show that these models are more helpful in the Epinions and Ciao datasets than the Filmtrust dataset. According to Table 5.5, Epinions and Ciao users are more active in social relationships than ratings. Therefore, these models benefit more from modelling the socially-influenced feature value differences.

From Figures 5.7 through 5.9 we can see that all state of the art social recommendation models except TrustSVD perform worse than the latent factor models SVD++ and BiasedMF on the Epinions and Filmtrust datasets for both measures. On the Ciao
dataset, SocialMF and SoReg achieve accuracies close to those of TrustSVD, SVD++, and BiasedMF. However, TrustMF still performs much worse. This is probably because none of these social recommendation methods model the user and item bias values. However, as [51] emphasise, user and item biases can explain a large proportion of user preference patterns in data. SVD++ seems to be competitive with BiasedMF. The main difference between these two methods is the addition of implicit feedback for user ratings in SVD++. This observation suggests that implicit feedback is not essentially helpful all the time. It seems that in some cases implicit feedback adds more noise to the model than actual preference patterns.

Consistent with the previous studies, analysing the model accuracies with respect to the number of latent factors in Figures 5.6 to 5.8 shows that increasing the number of factors usually has a deteriorating effect on the accuracy of the models. The deteriorating effect of using too many factors in latent factor models has been frequently emphasised by researchers in the literature, e.g. [118, 43, 69, 109].

The p-value results in Table 5.6a show that TFVSVD achieves superior accuracies over TrustSVD that are statistically significant. The values of MAE for all users, RMSE for all users, MAE for cold-start users, in the Epinions dataset are better than those of TrustSVD.
Figure 5.7: Comparison of the performance of latent factor models in terms of different number of factors on Filmtrust dataset, for a) MAE and b) RMSE for all users, and c) MAE and d) RMSE for cold-start users.

Figure 5.8: Comparison of the performance of latent factor models in terms of different number of factors on Ciao dataset, for a) MAE and b) RMSE for all users, and c) MAE and d) RMSE for cold-start users.
at the confidence level of above 99.99% which means that with over 99.99% probability, TFVSVD is better than TrustSVD in these cases. However, the p-value for RMSE for cold-start users in the Epinions dataset (0.363) shows that the proposed model does not outperform TrustSVD on a statistically significant level. This means that all users benefit more from socially-influenced conditional feature value differences than cold-start users alone. All users provide more ratings than cold-start users, and conditional feature value differences are trained using the user ratings. This explains why the difference between CTFVSVD and TFVSVD is more significant for all users than cold-start users. Therefore, TFVSVD achieves better accuracies that are statistically significant for both measures in the Epinions dataset for all users and for MAE for cold-start users, and better accuracies over TrustSVD in the Filmtrust dataset with 93.5% probability for RMSE for all users, worse accuracy over TrustSVD with 18.5% probability for MAE for all users, and better accuracies with TrustSVD for MAE and RMSE measures for cold-start users with 61.3% and 81.1% probabilities. The benefits of using socially-influenced conditional feature value are more tangible in the Epinions dataset.

<table>
<thead>
<tr>
<th>Table 5.6: t-test results for a) TFVSVD, and b) CTFVSVD against the best performing state of the art models.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Epinions</strong></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Filmtrust</strong></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Ciao</strong></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cold-Start Users</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

(a) TFVSVD  
(b) CTFVSVD

Table 5.6: t-test results for a) TFVSVD, and b) CTFVSVD against the best performing state of the art models.

*Underlined values show a positive difference and bold values show a significant differences between the proposed methods and the state of the art method.

Similarly, the p-value results in Table 5.6b show that CTFVSVD achieves better accuracies that are statistically significant. Specifically, in the Epinions dataset, the p-values against all three state of the art models for all users for both measures and for MAE of cold-start users are 0.000. This means that with almost 100% confidence, we can say that CTFVSVD is superior to TrustSVD, SVD++, and BiasedMF. In the Epinions dataset,
for RMSE of cold-start users, the p-value for CTFVSVD against TrustSVD is 0.287. This means that in this state, with a 71.3% confidence level, CTFVSVD is superior to TrustSVD. However, the difference is not statistically significant. As Table 5.6 shows, in both datasets CTFVSVD gets more significant results than TFVSVD. Therefore, addition of conditional dependencies between features in both datasets improves the accuracy, which means that conditional dependencies exist in the Epinions and Filmtrust datasets. In the Filmtrust dataset, for RMSE, CTFVSVD achieves better accuracies with 99.994% confidence for all users.

As Table 5.6 shows, the best performances of CTFVSVD and TFVSVD are obtained from the Ciao dataset. CTFVSVD and TFVSVD achieve accuracies that are significantly better than those of SocialMF, TrustSVD, SVD++, and BiasedMF for both measures for all users and for MAE for cold-start users. For cold-start users, these models achieve significantly better accuracies than TrustSVD, and comparable accuracies with SVD++ and BiasedMF. Although the improvements obtained by CTFVSVD and TFVSVD are rather small, according to Koren [50], there is evidence that even small improvements in prediction accuracy can lead to a significant impact on the quality of presented recommendations in practice.

Figure 5.7: Performance of CTFVSVD and TFVSVD compared with the state of the art on Epinions dataset, in terms of a) MAE, b) RMSE
Figure 5.8: Performance of CTFVSVD and TFVSVD compared with the state of the art on Filmtrust dataset in terms of a) MAE, b) RMSE

Figure 5.9: Performance of CTFVSVD and TFVSVD compared with the state of the art on Ciao dataset in terms of a) MAE, b) RMSE
Figure 5.10: Box plots of CTFVSVD and TFVSVD compared with the state of the art on Epinions dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users.
Figure 5.11: Box plots of CTFVSVD and TFVSVD compared with the state of the art on Filmtrust dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users.
Figure 5.12: Box plots of CTFVSVD and TFVSVSVD compared with the state of the art on Ciao dataset in terms of a) MAE for cold start users, b) RMSE for cold start users, c) MAE for all users, and d) RMSE for all users.
5.6 Summary of key points

1. Peer selection suggests that users tend to be connected with other users with similar preferences, and their preferences change over time due to social influence.

2. Although some models address the problem of modelling the social influence in preferences, social influence on conditional preferences over feature values is ignored.

3. This limitation prevents the models from learning some social preference patterns and hence degrades the accuracy.

4. The social influence in latent factor models can be modelled by introducing a new matrix to capture the social influence, and training the model considering the interactions of that matrix with other matrices.

5. Experiments confirm that such a method can successfully capture the social influence on user preference aspects.

6. According to the experiments, the efficiency of modelling conditional dependencies between preferences that are subject to social influence can depend on the dataset, the accuracy measure used, and the subset of users for whom the accuracy is measured. This necessitates designing a component-based model which can switch off aspects arbitrarily.

5.7 Conclusion

In Chapter 4, we addressed the problem of incorporating conditional preferences over feature values into the latent factor models and proposed CFVSD. In this chapter, we addressed the problem of learning socially-influenced conditional feature value preferences in latent factor models, in order to improve the recommendation accuracy of the state of the art models. In order to tackle this problem, we proposed TFVSD and CTFVSD. In CTFVSD, five additional matrices were incorporated into the latent factor models, to capture the conditional preferences over feature values that are subject to social influence. Similar to PMF, the prediction error was minimised by using stochastic gradient descent. TFVSD was a simpler version of CTFVSD, in which the conditional dependencies between features were not considered.

The advantage of TFVSD compared with the existing methods is that unlike the state of the art methods, it considers the differences between users in preferring item feature
values, which were subject to change by the influence of friends in social network. Furthermore, unlike most of the state of the art models, CTFVSVSVD also modelled the conditional dependencies between features, which enabled it to model non-linear interactions between features. As the experimental results in Section 5.4 revealed, in all the three datasets, modelling socially-influenced differences in feature value preferences and the conditional dependencies resulted in accuracy improvements, compared with the state of the art models that did not consider such feature value preferences and feature dependencies. The new extensions slightly increased the computational time of the proposed methods. However, as analysed in Section 5.3.1.2, the overall computation time still remained linear with respect to the number of observed ratings as well as observed trust statements. The applications of the proposed methods ranged from movie, music, books, to food, jokes, and dating.

The experiments showed that in the Epinions, Filmtrust, and Ciao datasets, incorporating socially-influenced feature value preferences makes improvements in model accuracy, which meant that the differences between users in preferring item feature values as well as feature dependencies exist, and these differences are subject to social influence.

The experimental results revealed that the CTFVSVSVD achieves significantly better results than all the other latent factor models based on Matrix Factorisation (PMF, BPMF, NMF, BiasedMF, and SVD++) and memory-based model, classic models (Global Average, Item Average, and User Average), traditional collaborative filtering models (ItemKNN and UserKNN), and the state of the art social recommendation models (TrustSVD, TrustMF, SocialMF, and SoReg). This confirmed the significance of incorporating socially-influenced conditional preferences over feature values in latent factor models.

We concluded that the characteristics of the datasets play an important role in success of the preference models. All models perform more successful in datasets with denser information. We also found that CTFVSVSVD specifically benefits from modelling the socially-influenced conditional feature value differences, in cases users are more socially active. An interesting observation is that the prediction accuracy for cold-start users benefits from a model that exploits social information in a dataset that includes denser information for both cold-start users and all users.
Chapter 5. Modelling Socially-Influenced Preferences in Recommender Systems
Chapter 6

Modelling Temporal Dynamics of Preferences

6.1 Introduction

Since data usually changes over time, the models should continuously update to reflect the present state of data. A problem with the most of the recent recommender systems is that they mostly ignore the drifting nature of preferences. Modelling the time drifting data is a central problem in data mining. Drifting preferences can be considered a particular type of concept drift, which has received much attention from researchers in recent years. However, very few recommendation models have considered the drifting nature of preferences. Changes in user preferences can originate from substantial reasons, or transient and circumstantial ones. For example, the items can undergo seasonal changes or some items may experience periodic changes, for instance, become popular in specific holidays. Apart from the short-term changes, user preferences are also subject to long-term drifts. For example, a user may be a fan of romantic or action movies at a younger age, while his/her preference may shift more towards drama movies as gets older. Also, users may change their rating scale over time. For example, a user may be very strict and give 3 out of 5 for the best movie. However, might become less strict with age and be more willing to elect the full rate when fully satisfied. A similar situation may apply for movies. A movie may receive a generally high/low rate at some time period, and lower/higher rates at some other period. Therefore, a preference model should be able to distinguish between different types of preference drifting, and model them individually in order to achieve the highest accuracy.

In Chapter 2 we identified six major aspects to the preferences. These aspects include
Chapter 6. Modelling Drifting Preferences in Recommender Systems

feature preferences \[115, 69\], feature value preferences \[113, 114, 118\], socially-influenced preferences \[111, 121, 65, 66, 42\], temporal dynamics \[50\], conditional preferences \[61\], and user and item biases \[51\]. In chapter 5, we integrated all these aspects into a unified model and proposed CTFVSVD.

In this chapter, we extend CTFVSVD, by considering the drifting nature of preferences and their constituting aspects. We assume that the socially-influenced preferences over features and conditional preferences over feature values, as well as user and item rating scales can be subject to temporal drift. Therefore, the two major research questions addressed in this chapter are:

- How can we efficiently model the drifting behaviour preferences, and how much improvement would incorporating such information make?
- Which aspects are more subject to temporal changes, and how is this related to the domain on which the model is trained?

The current work proposes a novel latent factor model based on matrix factorisation to address these two questions. This chapter has two major contributions for the field. In this chapter, we make further improvements on the accuracy of CTFVSVD, which proved to be the the most accurate model among a large set of state of the art models. The additional improvements were achieved by incorporating the temporal dynamics of preference aspects. We also draw conclusions about the dynamicity of preference aspects, by analysing the temporal aspects of the these aspects using a component-based approach, and show which aspects are more subject to drift over time. This research provides useful insights into the accurate modelling of preferences and their temporal properties, and helps pave the way for boosting the performance of recommender systems.

The rest of the chapter is organised as follows: In Section 6.2 we introduce the proposed model, called TCTFVSVD to overcome the challenge of learning drifting conditional socially-influenced preferences over feature values. In Section 6.3 we first explain the experimental setup, and then report on the results of the proposed model using two popular recommendation datasets. Finally we conclude the chapter in Section 6.5 by summarising the main findings of this work.

6.2 Modelling time-aware preference aspects in CTFVSVD

In this section, we explain how to integrate the time-awareness on different aspects of preferences into CTFVSVD \[111\], and propose time-aware CTFVSVD abbreviated to TCT-
Chapter 6. Modelling Drifting Preferences in Recommender Systems

Figure 6.1: The preference aspects and their interplay in TCTFVSVD

FVSVD [110]. In the following, we first provide a high-level view of TCTFVSVD by explaining the interactions between aspects that are captured by the model, and then elaborate how the aspects are trained from the users’ ratings and social relationships.

6.2.1 Aspect interactions and high-level view of the model

To address the problem of capturing drifting socially-influenced conditional preferences over feature values, we extend the method CTFVSVD proposed in Chapter 5 by adding the dynamicity of each one of the preference aspects that are assumed to be subject to concept drift. A high-level overview of the preference aspects and their interplay in TCTFVSVD are presented in Figure 6.1. Feature preferences and conditional feature value preferences as well as user and item bias were captured by CFVSVD in Chapter 4 and social influence was captured by CTFVSVD in Chapter 5.

In Figure 6.2a, FP represents preferences over features, which is captured by matrix $P$ in CTFVSVD. F represents item features captured by matrix $Q$ in CTFVSVD. CP represents conditional dependencies, FVP represents preferences over feature values, SI stands for social influence, and finally T is an abbreviation for time. TCTFVSVD incorporates additional matrices and vectors into matrix factorisation to capture as many aspects present in the data as possible. As Figure 6.2 shows, the model starts by loading the time-stamped user ratings as well as the social network data into the memory. The main loop accounts for the learning iterations over the model. The first loop within the main loop iterates over the time-stamped user-item ratings matrix, while the second loop iterates over the social network adjacency matrix, to train the socially-influenced parts of the model. In each loop, one entry of the input matrix is read and used to update the ma-
traces/vectors related to that input data. As can be seen, the user and item bias values are only updated in loop 1, since they are only related to the user-item ratings. Both user-item ratings and the users’ social relationships include information about the users’ preferences over features. Therefore, the new values for FP are calculated in both loops and updated in the main loop, when all new values have been calculated. Similarly, the values for SI and FVP depend on both user-item ratings and social relationships. Consequently, their new values are calculated inside both loops 1 and 2, and are updated in the main loop. In contrast, the values of F as well as CP only need the user-item ratings to be updated. Therefore, they are immediately updated inside loop 1. The time aspect includes parameters that account for the dynamics of user and item biases, feature value preferences, and preferences over features. Since bias values do not depend on the user-item ratings matrix, they are updated immediately in loop 1. However, the new values for the dynamics of feature value preferences, and preferences over features are updated in the main loop. In TCTFVSVD, every one of the preference aspects can be arbitrarily switched off and on by
setting their respective learning rates and regularisation parameters (hyper-parameters) to zero or a non-zero value respectively.

Although social relationships are likely to be time-dependent, most datasets do not contain information about the time-related properties of social connections (e.g. the time when the users are connected). Conditional preferences are related to the feature value preferences, since they model the dependencies between the features and their values, and therefore, are applied to the matrices that account for the users’ preferences over feature values. Social influence is applied to the aspects of preferences over features and preferences over feature values. However, applying social influence to the user and item biases showed no observable benefits and user or item biases do not seem to be influenced by social interactions. Therefore, we concluded that user and item biases are not much influenced by the social interactions. Therefore, in the most abstract view of the model as depicted in the high-level representation in Figure 6.2a, the model is comprised of four main modules. Initialising the model parameters (Model Initialiser), learning the intrinsic constituting aspects of preferences (i.e. preferences over features, preferences over feature values, conditional dependencies, and user and item bias values) and the drifting properties of preferences (Intrinsic Trainer), learning the social influence of the friends over the drifting intrinsic preference aspects (Social Trainer), and finally updating the model to reflect the new information extracted from the data about user ratings, time, and social connections (Model Updater). These modules will be discussed in more details later, when we introduce the algorithm in Section 6.2.4.

6.2.2 TCTFVSVD model formulation

In this section, we provide the mathematical formulation of the preferences captured in TCTFVSVD. In TCTFVSVD, the user preferences are modelled as a Bayesian network. Figure 6.3 shows the topology or the structure of the Bayesian network for user preferences that are modelled by TCTFVSVD.

TCTFVSVD extends CTFVSVD, by adding the time factor to the aspects of preferences as depicted in Fig 6.1. In CTFVSVD, the user preferences were captured using the matrices $P, Q, W, Z, Y, \omega, y$, with the hyper-parameters $\sigma_P, \sigma_Q, \sigma_W, \sigma_Z, \sigma_\omega, \sigma_y, \sigma_Y, \sigma_{bu}$ and $\sigma_{bi}$.

In TCTFVSVD, the drifting social influence of friends in the user’s social network are captured through Eqs. 6.1 to 6.3.
where $\hat{T}^t_{uv}$, $\hat{S}^t_{uv}$, $\hat{G}^t_{uv}$ model the time-dependent influence of user $v$ on the preferences of user $u$ for the preferences over features (captured by $P_{uf}(t)$) and preferences over feature values (captured by $W_{uf}(t)$ and $Z_{uf}(t)$), and similar to CTFVSVD, $\omega$ captures the implicit influence of user $u$ on other users and is obtained using the matrix factorisation process. As can be seen in Figure 6.3, the user preferences over features and feature values in TCTFVSVD are subject to social influence, and they also drift over time. In Eqs. 6.1 to 6.3 $I_u^t$ is the set of time-stamps for all the ratings given by user $u$. Therefore, using
these equations, the influence of the user \( v \) on the preferences of user \( u \) is calculated for all the time points, and then it is averaged. Intuitively, these equations are telling us that the trust of user \( u \) in user \( v \) can be estimated by calculating the average of the weighted averages of user \( v \)’s influence on user \( u \)’s preferences for different features, in different times. Intuitively, if user \( u \) strongly trusts user \( v \), his preferences would be more strongly influenced by user \( v \). Furthermore, depending on the trust strength of user \( u \) in user \( v \) and the influence he gets from user \( v \) and its direction (positive or negative), the user’s preference can be positively or negatively affected. Therefore in TCTFVSVD, the user preferences are subject to social influence, and the social influence depends on the strength of their trust in the friends. According to these equations, if there is no relationship between user \( u \) and user \( v \), user \( u \)’s preferences will not be directly affected by the social influence of user \( v \). In TCTFVSVD, the drifting preference value of the user \( u \) over an item \( j \) at time \( t \) is obtained according to Eq. 6.4.

\[
\hat{R}_{uj}(t_{uj}) = \mu + bu_{u}(t_{uj}) + bi_{j}(t_{uj})
\]

\[
+ \sum_{f=1}^{D} (P_{uf}(t_{uj}) + |I_u|^{-\frac{1}{2}} \sum_{v \in I_u} y_{if} + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf})(W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj}))
\]

\[
+ \sum_{f' = 1}^{D} (\sum_{f = 1}^{D} (W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj}))Y_{ff'})(W_{uf}(t_{uj})Q_{jf'} + Z_{uf}(t_{uj}))
\]

\[\text{(6.4)}\]

According to Eq. 6.4 in TCTFVSVD, different aspects of preferences as well as user and item biases are subject to temporal drift. Notice that this is dynamic version of the Eq. 5.23 in Chapter 5. As can be seen in Eqs. 6.1 to 6.4, the user bias, item bias, preferences over features captured by the matrix \( P \), and preferences over feature values captured by the matrices \( W \) and \( Z \) are subject to temporal drift. In order to model the drifting properties of these aspects, we use Eqs. 6.5 to 6.9.

\[
b_{u_{u}}(t_{uj}) = bu_{u} + \alpha_{u}dev_{u}(t_{uj}) + but_{ut_{uj}} \tag{6.5}\]

\[
b_{i_{j}}(t_{uj}) = (b_{i_{j}} + bi_{j}Bin(t_{uj}))(C_{u} + Ct_{ut_{uj}}) \tag{6.6}\]

\[
P_{uf}(t_{uj}) = P_{uf} + \alpha_{u}P_{dev_{u}(t_{uj})} + Pt_{ut_{uj}} \tag{6.7}\]
Z_{uf}(t_{uj}) = Z_{uf} + \alpha_u^Z \text{dev}_u(t_{uj}) + Z_{uft_{uj}} \tag{6.8}

W_{uf}(t_{uj}) = W_{uf} + \alpha_u^W \text{dev}_u(t_{uj}) + W_{uft_{uj}} \tag{6.9}

where \( P_{uf} \), \( W_{uf} \), and \( Z_{uf} \) capture the static preferences of the user \( u \), while the variables \( P_{uft_{uj}} \), \( W_{uft_{uj}} \), \( Z_{uft_{uj}} \) capture the short-term variations in the user preferences for feature \( f \) (e.g. due to the mood of the users in a particular day), and \( \alpha_u^P \), \( \alpha_u^W \), and \( \alpha_u^Z \) model the users’ long-term preference shifts, and \( \text{dev}_u(t_{uj}) \) is obtained according to Eq. 6.10. \[ \text{dev}_u(t_{uj}) = \text{sign}(t_{uj} - t_u)|t_{uj} - t_u|^\beta \tag{6.10} \]

where \( t_u \) is the mean of the dates for the ratings given by the user \( u \), and \( \beta \) is a constant value. In Eq. 6.6 all the dates are placed in a fixed number of bins, and the function \( \text{Bin}(.) \) returns the bin number for a particular date. For example, if the maximum period of the ratings is 30 years and 30 bins are used, all the ratings given in a particular year are placed in a bin, and the function \( \text{Bin}(.) \) returns the year number for that particular year. The reason why this function is only used for items is that items (and not for users) are not expected to change on a daily basis, and as opposed to users’ biases, longer time periods are expected to pass, before we see any changes in the items’ popularity. In simple words, \( \text{dev}_u(t_{uj}) \) shows how much the time of the rating given by user \( u \) to item \( j \) deviates from the average time of the ratings given by that user. Therefore, if a rating is given at the same time as the average time of the ratings, then according to these equations, there will be no long-term preference shift for that aspect. However, for instance, if the average time of the ratings given by user \( u \) is 11/04/2006, the rating of the same item by that user on 11/04/2016 would be different, and this shift is captured by the coefficients of the function \( \text{dev}_u(t_{uj}) \) in Eq. 6.5 and Eqs. 6.7 to 6.9.

The drifting preferences captured using Eq. 6.5 and Eqs. 6.7 to 6.9 are depicted in Figure 6.4. In these figures, the mean of the dates on which the user has given the ratings are assumed to be 50 (the fiftieth day in a year), and the variations of the user preferences over a period of one year are captured for different values of \( \alpha \) in Eq. 6.5 and Eqs. 6.7 to 6.9. The red lines in these figures represent the case in which the day-specific variations in the user preferences are not captured, while the blue lines also include the day-specific variations. Therefore, as can be seen, in these figures there are two types of preference shifts, long-term drifts (captured by the values of \( \alpha, \alpha^P, \alpha^W \), and \( \alpha^Z \)), and short-term or day-specific drifts (captured by the values of \( P_u, P_t, W_t, \) and \( Z_t \)). Therefore, the
preference drifts are comprised of small variations from one day to the other, mainly because of temporary factors such as the mood of the user, and the large variations which happen in the long-term, as the user changes preferences because of the shift in the his/her tastes. The blue lines show the preference shift patterns that can be learnt by TCTFVSVD. Furthermore, the first three terms in Eq. 6.14 model the social influence of the feature preferences and feature value preferences captured by \( P, \alpha^P, Pt, W, \alpha^W, Wt, Z, \alpha^Z, Zt \).

Therefore, assuming that two users have established the social relationship from the very beginning (which is not essentially true, but usually social relationships do not contain time-stamps), using Eqs. 6.1 to 6.3, the social influence is applied to the preferences of the user over the entire period for which the rating data is recorded. Therefore, the formulation of the estimated ratings in TCTFVSVD (Eq. 6.4) allows it to learn the drifting conditional feature value preferences, and the formulation of the optimisation problem in TCTFVSVD (Eq. 6.14) enables it to learn the influence of social friends on the drifting preferences of a user.

Eqs. 6.5 to 6.9 show how TCTFVSVD can capture long-term and short-term drifts in each of the preference aspects (user bias, item bias, feature preferences, and feature value preferences). The advantage of formulating the problem using Eq. 6.4 is that each of these preference aspects can be arbitrarily switched on/off. This results in a component-based approach, in which the model aspects interact with each other, with the purpose of extracting as much preference patterns from the raw data as possible.

### 6.2.3 TCTFVSVD model training

According to the Bayesian network of TCTFVSVD in Figure 6.3, this model minimises the log-posterior probability of matrices that define the user preferences, given the model hyper-parameters and the training matrix. Formally,

\[
\begin{align*}
\arg\min_{P, Pt, \alpha^P, Q, W, Wt, \alpha^W, Z, Zt, \alpha^Z, Y, \omega, y, bu, a, bu_t, c_t, \alpha, \alpha^P, \omega, P, Z_t, W_t} & \\
\text{lnp}(P, Q, W, Z, \omega, y, bu, a, bu_t, c_t, \alpha, \alpha^P, \omega, P, Z_t, W_t) | R, T_t, S_t, G_t, \sigma_N} & \\
\end{align*}
\]

\( \sigma_N = \{\sigma, \sigma_T, \sigma_P, \sigma_{Pt}, \sigma_{\alpha^P}, \sigma_Q, \sigma_{Wt}, \sigma_{\alpha^W}, \sigma_Z, \sigma_{Zt}, \sigma_{\alpha^Z}, \sigma_{\omega}, \sigma_y, \sigma_{bu}, \sigma_a, \sigma_{bu_t}, \sigma_c, \sigma_{Ct}, \sigma_{bt}, \sigma_{bt}, \sigma_y \} \) denotes the set of all the hyper-parameters. \( T_t, S_t, G_t \) respectively denote the real values for the estimated matrices \( \hat{T}_t, \hat{S}_t, \) and \( \hat{G}_t \) in Eqs. 6.1 to 6.3. According to the Bayesian network in Figure 6.3 and by decomposing the full joint distribution using chain rule of probability theory \cite{158} according to the conditional dependencies between the variables...
Figure 6.4: An example of drifting preferences in Eq. 6.5 and Eqs. 6.7 to 6.9 for a) positive $\alpha$ values and b) negative $\alpha$ values defined in this figure, minimising the probability above is equal to minimising the value given in Eq. 6.12.

\[
\begin{align*}
\text{argmin}_{P,P_{t},Q,W,W_{t},Z,Z_{t},\sigma_{y},Y,Y_{t},\omega,\omega_{t},\alpha,\alpha_{t},\nu,\nu_{t},\lambda,\lambda_{t},\beta,\beta_{t},\gamma,\gamma_{t},\delta,\delta_{t},\epsilon,\epsilon_{t}} & \{ \ln p(R|P(t), Q, W(t), Z(t), \mu(t), \nu(t), Y, \sigma) + \ln p(Q|\sigma_Q) + \\
& + \ln p(P(t)|\sigma_P) + \ln p(W(t)|\sigma_W) + \ln p(Z(t)|\sigma_Z) + \\
& + \ln p(\mu(t)|\sigma_\mu) + \ln p(\nu(t)|\sigma_\nu) + \ln p(\omega(t)|\sigma_\omega) + \ln p(\omega_{t}|\sigma_\omega) + \\
& + \ln p(T^t\omega_{uv}\omega, P(t), \sigma_T) + \ln p(S^t\omega_{uv}\omega, W(t), \sigma_T) + \ln p(G^t\omega_{uv}\omega, Z(t), \sigma_T) + \\
& + \ln p(P(t)|\sigma_T) + \ln p(W(t)|\sigma_T) + \ln p(Z(t)|\sigma_T) + \ln p(\omega|\sigma_T) \} \\
\end{align*}
\]
Chapter 6. Modelling Drifting Preferences in Recommender Systems

123

Provided that all the probabilities above follow a normal distribution, it can be shown that minimising the function in Eq. 6.12 is equivalent to minimising the error value using the function in Eq. 6.15.

\[
E_R = \frac{1}{2} \sum_{u=1}^{N} \sum_{j=1}^{M} (R_{uj} - \hat{R}_{uj})^2 + \frac{\lambda_P}{2} \sum_{j=1}^{M} \|Q_j\|_{Frob}^2 + \frac{\lambda_U}{2} \sum_{i=1}^{M} \|U_i\|_{Frob}^2 \\
+ \sum_{u=1}^{N} \frac{\lambda_P}{2} |T_u|^{-\frac{1}{2}} (\|P_u\|_{Frob}^2 + \|P_{tuv}\|_{Frob}^2 + \|\alpha^P\|_{Frob}^2) \\
+ \sum_{u=1}^{N} \frac{\lambda_W}{2} |T_u|^{-\frac{1}{2}} (\|W_u\|_{Frob}^2 + \|W_{tuv}\|_{Frob}^2 + \|\alpha^W\|_{Frob}^2) \\
+ \sum_{u=1}^{N} \frac{\lambda_Z}{2} |T_u|^{-\frac{1}{2}} (\|Z_u\|_{Frob}^2 + \|Z_{tuv}\|_{Frob}^2 + \|\alpha^Z\|_{Frob}^2) \\
+ \sum_{u=1}^{N} \lambda_{B,i} \sum_{j=1}^{M} |U_j|^{-\frac{1}{2}} b_{ij}^2 + \frac{\lambda_{B,i}}{2} \sum_{j=1}^{M} \sum_{v \in I_j} |U_j|^{-\frac{1}{2}} b_{ij}^2 \text{Bin}(t) + \frac{\lambda_V}{2} \sum_{f=1}^{D} \sum_{j'=1}^{D} V_{jj'}^2
\]

\[
E_T = \frac{\lambda_P}{2} \sum_{u=1}^{N} \sum_{v \in T_u} (T_{uv} - \hat{T}_{uv})^2 + \frac{\lambda_W}{2} \sum_{u=1}^{N} \sum_{v \in T_u} (T_{uv} - \hat{T}_{uv})^2 + \frac{\lambda_Z}{2} \sum_{u=1}^{N} \sum_{v \in T_u} (T_{uv} - \hat{T}_{uv})^2 \\
+ \sum_{u=1}^{N} \frac{\lambda_P}{2} |T_u|^{-\frac{1}{2}} (\|P_u\|_{Frob}^2 + \|P_{tuv}\|_{Frob}^2 + \|\alpha^P\|_{Frob}^2) \\
+ \sum_{u=1}^{N} \frac{\lambda_W}{2} |T_u|^{-\frac{1}{2}} (\|W_u\|_{Frob}^2 + \|W_{tuv}\|_{Frob}^2 + \|\alpha^W\|_{Frob}^2) \\
+ \sum_{u=1}^{N} \frac{\lambda_Z}{2} |T_u|^{-\frac{1}{2}} (\|Z_u\|_{Frob}^2 + \|Z_{tuv}\|_{Frob}^2 + \|\alpha^Z\|_{Frob}^2) \\
+ \frac{\lambda}{2} \sum_{u=1}^{N} |T_u|^{-\frac{1}{2}} \|\omega_u\|_{Frob}^2
\]

\[
\arg\min_{P,P_t,\alpha^P,Q,W,W_t,\alpha^W,Z,Z_t,\alpha^Z,Y,\omega,y,b,u,\alpha,b,t,C,Y,B,t} \left[ E = E_R + E_T \right]
\]

(6.15)

where \( I_j \) is the set of time-stamps, for all the ratings given to item \( j \), and \( \eta_P, \eta_W \), and \( \eta_Z \) are constants added to control the weights of the components related to the social
aspect in this equation. In TCTFVSVD, we use gradient descent to get the optimal values formulated using the function in Eq. 6.15. The gradients for the model parameters are obtained as Eqs. 6.16 to 6.24.

\[
\frac{\partial E}{\partial b_{tu}} = \frac{\partial E_R}{\partial b_{tu}} = e_{uj} + \lambda_{bui} |I_u|^{-\frac{1}{2}} b_{tu}
\]

(6.16)

\[
\forall t_{uj} \in I'_u: \frac{\partial E}{\partial b_{utu}} = \frac{\partial E_R}{\partial b_{utu}} = e_{uj} + \lambda_{bui} |I_u|^{-\frac{1}{2}} b_{atu}
\]

(6.17)

\[
\frac{\partial E}{\partial \alpha_u} = \frac{\partial E_R}{\partial \alpha_u} = e_{uj} \text{dev}_u(t_{uj}) + \lambda_{bui} |I_u|^{-\frac{1}{2}} \alpha_u
\]

(6.18)

\[
\frac{\partial E}{\partial b_{ij}} = \frac{\partial E_R}{\partial b_{ij}} = e_{uj}(C_u + C_{ui}) + \lambda_{bij} |J_j|^{-\frac{1}{2}} b_{ij}
\]

(6.19)

\[
\frac{\partial E}{\partial b_{iBin(t_{uj})}} = \frac{\partial E_R}{\partial b_{iBin(t_{uj})}} = e_{uj}(C_u + C_{ut}) + \lambda_{bij} |J_j|^{-\frac{1}{2}} b_{iBin(t_{uj})}
\]

(6.20)

\[
\frac{\partial E}{\partial C_u} = \frac{\partial E_R}{\partial C_u} = e_{uj}(b_{ij} + b_{iBin(t_{uj})}) + \lambda_{bij} |J_j|^{-\frac{1}{2}} C_u
\]

(6.21)

\[
\frac{\partial E}{\partial C_tu} = \frac{\partial E_R}{\partial C_tu} = e_{uj}(b_{ij} + b_{iBin(t_{uj})}) + \lambda_{bij} |J_j|^{-\frac{1}{2}} C_{tu}
\]

(6.22)

\[
\frac{\partial E}{\partial P_{uf}} = \frac{\partial E_R}{\partial P_{uf}} + \frac{\partial E_T}{\partial P_{uf}}
\]

(6.23)

\[
\frac{\partial E_R}{\partial P_{uf}} = e_{uj}(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + \lambda_P |I_u|^{-\frac{1}{2}} P_{uf}
\]

(6.24)
\[
\frac{\partial E_T}{\partial P_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}} P_{uf} + \lambda_T \eta_P \sum_{v \in T_u} e^{(1)}_{uv} \omega_{uf} \tag{6.25}
\]

\[
\forall t_{u_j} \in I_u^t: \frac{\partial E}{\partial P_{uf}} = \frac{\partial E_R}{\partial P_{uf}} + \frac{\partial E_T}{\partial P_{uf}} \tag{6.26}
\]

\[
\frac{\partial E_R}{\partial P_{uf}} = e_{u_j}(W_{uf}(t_{u_j})Q_{jf}(t_{u_j}) + Z_{uf}(t_{u_j})) + \lambda_P|I_u|^{-\frac{1}{2}} P_{uf} \tag{6.27}
\]

\[
\frac{\partial E_T}{\partial P_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}} P_{uf} + \frac{\lambda_T \eta_P}{|I_u|} \sum_{v \in T_u} e^{(1)}_{uv} \omega_{uf} \tag{6.28}
\]

\[
\frac{\partial E}{\partial \alpha_{uf}} = \frac{\partial E_R}{\partial \alpha_{uf}} + \frac{\partial E_T}{\partial \alpha_{uf}} \tag{6.29}
\]

\[
\frac{\partial E_R}{\partial \alpha_{uf}} = e_{u_j} \text{dev}_u(t_{u_j})(W_{uf}(t_{u_j})Q_{jf}(t_{u_j}) + Z_{uf}(t_{u_j})) + \lambda_P|I_u|^{-\frac{1}{2}} \alpha_{uf}^p \tag{6.30}
\]

\[
\frac{\partial E_T}{\partial \alpha_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}} \alpha_{uf}^p + \frac{\lambda_T \eta_P}{|I_u|} \sum_{v \in T_u} \sum_{t_{u_j} \in I_u^t} e^{(1)}_{uv} \omega_{uf} \text{dev}_u(t_{u_j}) \tag{6.31}
\]

\[
\frac{\partial E}{\partial W_{uf}} = \frac{\partial E_R}{\partial W_{uf}} + \frac{\partial E_T}{\partial W_{uf}} \tag{6.32}
\]

\[
\frac{\partial E_R}{\partial W_{uf}} = e_{u_j}Q_{jf}(t_{u_j})(W_{uf}(t_{u_j})Q_{jf}(t_{u_j}) + Z_{uf}(t_{u_j})) + 2Q_{jf}(t_{u_j}) \sum_{f' = 1}^{D} (W_{uf}(t_{u_j})Q_{jf'}(t_{u_j}) + Z_{uf'}(t_{u_j})) + \lambda_W|I_u|^{-\frac{1}{2}} W_{uf} \tag{6.33}
\]

\[
\frac{\partial E_T}{\partial W_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}} W_{uf} + \lambda_T \eta_W \sum_{v \in T_u} e^{(2)}_{uv} \omega_{uf} \tag{6.34}
\]
\[
\forall t_{uj} \in I_t^u : \frac{\partial E}{\partial W_{uft}} = \frac{\partial E_R}{\partial W_{uft}} + \frac{\partial E_T}{\partial W_{uft}} \tag{6.35}
\]

\[
\frac{\partial E_R}{\partial W_{uft}} = e_{uj}Q_{jf}(t_{uj})(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + 2Q_{jf}(t_{uj}) \sum_{f'=1}^{D} (W_{uf}(t_{uj})Q_{jf'}(t_{uj}) + Z_{uf'}(t_{uj})) + \lambda_W |I_u|^{-\frac{1}{2}} W_{uf} \tag{6.36}
\]

\[
\frac{\partial E_T}{\partial W_{uft}} = \lambda_T |T_u|^{-\frac{1}{2}} W_{uf} + \frac{\lambda_T W}{|I_t^u|} \sum_{v \in T_u} e_{uv}^{(2)} \omega_{vf} \tag{6.37}
\]

\[
\frac{\partial E}{\partial \alpha_{uf}^W} = \frac{\partial E_R}{\partial \alpha_{uf}^W} + \frac{\partial E_T}{\partial \alpha_{uf}^W} \tag{6.38}
\]

\[
\frac{\partial E_R}{\partial \alpha_{uf}^W} = e_{uj}d e_{uv}(t_{uj})(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + 2Q_{jf}(t_{uj}) \sum_{f'=1}^{D} (W_{uf}(t_{uj})Q_{jf'}(t_{uj}) + Z_{uf'}(t_{uj})) + \lambda_{\alpha} |I_u|^{-\frac{1}{2}} \alpha_{uf}^W \tag{6.39}
\]

\[
\frac{\partial E_T}{\partial \alpha_{uf}^W} = \lambda_T |T_u|^{-\frac{1}{2}} \alpha_{uf}^W + \frac{\lambda_T W}{|I_t^u|} \sum_{v \in T_u} \sum_{v' \in I_u^v} e_{uv}^{(2)} \omega_{vf} d e_{uv}(t_{uj}) \tag{6.40}
\]

\[
\frac{\partial E}{\partial Z_{uf}} = \frac{\partial E_R}{\partial Z_{uf}} + \frac{\partial E_T}{\partial Z_{uf}} \tag{6.41}
\]

\[
\frac{\partial E_R}{\partial Z_{uf}} = e_{uj}(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + 2 \sum_{f'=1}^{D} (W_{uf}(t_{uj})Q_{jf'}(t_{uj}) + Z_{uf'}(t_{uj})) + \lambda_Z |I_u|^{-\frac{1}{2}} Z_{uf} \tag{6.42}
\]

\[
\frac{\partial E_T}{\partial Z_{uf}} = \lambda_T |T_u|^{-\frac{1}{2}} Z_{uf} + \lambda_{\omega} \sum_{v \in T_u} e_{uv}^{(3)} \omega_{vf} \tag{6.43}
\]
\[ \forall t_{uj} \in I_u^t : \frac{\partial E}{\partial Z_{uf}} = \frac{\partial E_R}{\partial Z_{uf}} + \frac{\partial E_T}{\partial Z_{uf}} \] (6.44)

\[ \frac{\partial E_R}{\partial Z_{uf}} = e_{uj}(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + 2 \sum_{j' = 1}^{D} (W_{uf}(t_{uj})Q_{j'f}(t_{uj}) + Z_{uf}(t_{uj}))+ \lambda_Z|I_u|^{-\frac{1}{2}}Z_{uf} \] (6.45)

\[ \frac{\partial E_T}{\partial Z_{uf}} = \lambda_T|I_u|^{-\frac{1}{2}}Z_{uf} + \frac{\lambda_T \eta_Z}{|I_u|} \sum_{u \in I_u} e_{uj}^{(3)} \omega_{uf} \] (6.46)

\[ \frac{\partial E}{\partial \alpha_{uf}^2} = \frac{\partial E_R}{\partial \alpha_{uf}^2} + \frac{\partial E_T}{\partial \alpha_{uf}^2} \] (6.47)

\[ \frac{\partial E_R}{\partial \alpha_{uf}^2} = e_{uj}dev_u(t_{uj})(W_{uf}(t_{uj})Q_{jf}(t_{uj}) + Z_{uf}(t_{uj})) + 2 \sum_{j' = 1}^{D} (W_{uf}(t_{uj})Q_{j'f}(t_{uj}) + Z_{uf}(t_{uj}))+ \lambda_a|I_u|^{-\frac{1}{2}} \alpha_{uf}^2 \] (6.48)

\[ \frac{\partial E_T}{\partial \alpha_{uf}^2} = \lambda_T|I_u|^{-\frac{1}{2}} \alpha_{uf}^2 + \frac{\lambda_T \eta_Z}{|I_u|} \sum_{u \in I_u} \sum_{t_{uj} \in I_u^t} e_{uj}^{(3)} \omega_{uf} dev_u(t_{uj}) \] (6.49)

\[ \forall i \in I_u : \frac{\partial E}{\partial y_{if}} = \frac{\partial E_R}{\partial y_{if}} = e_{uj}|I_u|^{-\frac{1}{2}}(W_{uf}V_{jf} + Z_{uf}) + (\lambda_y|J_f|^{-\frac{1}{2}}y_{if}) \] (6.50)

\[ \forall v \in I_u : \frac{\partial E}{\partial \omega_{vf}} = \frac{\partial E_R}{\partial \omega_{vf}} + \frac{\partial E_T}{\partial \omega_{vf}} \] (6.51)

\[ \frac{\partial E_R}{\partial \omega_{vf}} = e_{uj}|I_u|^{-\frac{1}{2}}(W(t)_{uf}Q_{jf} + Z(t)_{uf}) \] (6.52)
\[ \frac{\partial E_T}{\partial \omega_{uf}} = (\lambda_T |T_u^+|^{-\frac{1}{2}}) \omega_{uf} + \lambda_1 \eta_P \sum_{v \in I_u} e_{uv}^{(1)} P(t_{uj})_{uf} + \lambda_2 \eta_W \sum_{v \in I_u} e_{uv}^{(2)} (1 - W(t_{uj})_{uf}) + \lambda_3 \eta_Z \sum_{v \in I_u} e_{uv}^{(3)} Z(t_{uj})_{uf} \]  

(6.53)

\[ \frac{\partial E}{\partial Y_{ff'}} = \frac{\partial E_R}{\partial Y_{ff'}} = e_{uf}(W_{if}V_{jf} + Z_{if})(W_{if}V_{jf'} + Z_{if'}) - \lambda Y_{ff'} \]  

(6.54)

\[ \frac{\partial E}{\partial Q_{jf}} = \frac{\partial E_R}{\partial Q_{jf}} = e_{uf}[W_{uf}(P_{uf} + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} y_i + |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_v) + 2W_{uf} \sum_{f'}^{D} (W_{f'f}V_{jf'} + Z_{if'})Y_{ff'}] + \lambda Q|U_j|^{-\frac{1}{2}} Q_{jf} \]  

(6.55)

where:

\[ e_{uf} = R_{uf} - \hat{R}_{uf} \]  

(6.56)

Therefore, the gradients in Eqs. 6.16 to 6.35 are used to update the values of the matrices and the vectors used to capture socially-influenced drifting conditional feature value preferences using an incremental gradient descent method.

### 6.2.4 TCTFVSVD algorithm

Algorithm 1 describes the details of the gradient descent method TCTFVSVD uses to train the model parameters \( (P, Pt, \alpha^P, Q, W, Wt, \alpha^W, Z, Zt, \alpha^Z, Y, \omega, y, bu, \alpha, but, C, Ct, bi, bit) \) as expressed in Eq. 6.15.

The algorithm receives the set of model hyper-parameters \( \lambda \) and the set of learning rates \( \gamma \) as input, and trains the model parameters according to the Bayesian approach described in Section 6.2.2. As we showed in the high-level representation of the algorithm in Figure 6.2a, the model is comprised of four basic components. A model initialiser, which initialises the model parameters after the input data is loaded into memory, an intrinsic trainer, which trains the model parameters using the user-item ratings, a social trainer which trains the model parameters using the social relationship data, and finally, a model updater, which updates the model based on the trained parameters for a particular iteration.
As can be seen in line 11 in Algorithm 1, the training starts with initialising the model parameters. The matrices \( P, Q, y, \) and \( \omega \) and user and item bias vectors \( (bu \text{ and } bi) \) are randomly initialised using a Gaussian distribution with a mean of zero and a standard deviation of one. The new matrices \( Pt, W, Wt, Z, Zt, Ct, \text{ but, bit, and } Y \) and the vectors \( \alpha, \alpha^P, \alpha^W, \alpha^Z, C \) are initialised with constant values. By using constant values to initialise the matrices and vectors, the algorithm starts the search process at the same starting point as CTFVSVD, and explores the modified search space to find more promising solutions, by considering the possible conditional dependencies between the features and the differences between users in preferring item feature values, as well as dynamic properties of the preferences, and the influence of social friends in the preferences of a user.

The main algorithm consists of a main loop, which implements the learning iterations of the model. Each iteration is comprised of one model intrinsic training operation (Algorithm 2), one model social training operation (Algorithm 3), and one model updating operation (Algorithm 4). In the model intrinsic trainer, the model parameters are updated using the gradient values in Eqs. 6.16 to 6.35 using a rating value that is read from the user-item ratings matrix. First in line 8, the estimated rating is calculated according to Eq. 6.4. Then the basic parameters of the model, \( P, Q, W, Z, Y, bu, \) and \( bi, \) and the temporal parameters \( but, bit, \alpha, C, Ct, \alpha^P, \alpha^W, \alpha^Z, Pt, Wt, \) and \( Zt \) are updated using the rating-related gradient values \( \left( \frac{\partial E_R}{\partial \theta_k} \right) \) in Eqs. 6.16 to 6.35. Since this trainer only learns the intrinsic user preferences, only the error value in Eq. 6.13 is used to update the model parameters. After learning the intrinsic preferences, the function in Algorithm 3 is invoked to train the social aspects of the preferences. Similar to IntrinsicTrainer, SocialTrainer is also comprised of a main loop, which iterates over the social relationship data in the social matrix. In each iteration, one entry from the social matrix is read, and the socially-influenced parameters of the model are updated though the gradient values that are obtained using the error in Eq. 6.14. Finally, the ModelUpdater in Algorithm 4 is invoked, and the calculated model updates are applied to the model parameters. This process is repeated for a fixed number of iterations, or until a specific condition is met. At the end of this process, the model parameters \( (P, Pt, \alpha^P, Q, W, Wt, \alpha^W, Z, Zt, \alpha^Z, Y, \omega, y, bu, \alpha, but, C, Ct, bi, bit) \) are trained using the input data, and can be used to estimate the rating value given by a user \( u \) to an item \( j \) according to Eq. 6.4.
expressed in Eq. 6.57. Therefore, the computation time of the model trainer is comprised of one main loop that iterates for a fixed number of iterations (maxIter). Therefore, the computation time of the model trainer is expressed in Eq. 6.57

\[ C(\text{ModelTrainer}) = C(\text{IntrinsicTrainer}) + C(\text{SocialTrainer}) + C(\text{ModelUpdater}) \] (6.57)

6.2.5 Computational complexity analysis

The model training in Algorithm 1 is comprised of one main loop that iterates for a fixed number of iterations (maxIter). Therefore, the computation time of the model trainer is expressed in Eq. 6.57.

First, we examine the computational complexity of Intrinsic Training in Algorithm 2. On the highest level, this algorithm is comprised of two loops that iterate over the non-zero ratings in the rating matrix \( R \). In the following, \(|R|\) and \(|T|\) denote the number of non-zero entries in the rating matrix \( R \) and adjacency matrix \( T \) respectively. In Intrinsic Trainer:

- The number of repetitions to calculate the estimated ratings \( \hat{R} \) in line 8 is \( (D^2 \times |R|) + (D \times \sum_{u=1}^{N} |I_u|^2) + (D \times \sum_{u=1}^{N} |I_u| \times |T_u|) \).
- The number of repetitions to update the parameters related to the user and item biases in lines 10, 11 and 12 is \( 7 \times |R| \).
- The number of repetitions needed to update the parameters \( P, Q, W, \) and \( Z \) in lines 15, 16, 17 and 18 is \( 10 \times D \times |R| \).
- The number of repetitions needed to update the parameters \( \omega \) in line 19 is \( D \times \sum_{u=1}^{N} (|I_u| \times |T_u|) \).

\[ \lambda \] is the set of the model hyper-parameters as specified in Eqs. 6.13 and 6.14 and Figure 6.1. \( N, M, \) and \( D \) respectively denote number of users, number of items, and number of features. \( \gamma \) denotes the set of learning rates, \( \text{maxIter} \) denotes the maximum number of learning iterations.
Algorithm 2 Intrinsic Training

1: void IntrinsicTrainer($\lambda$, $\gamma$)
2: {
3:     $u \leftarrow 1$
4:     for $u \leq N$ do
5:         $j \leftarrow 1$
6:         for $j \leq M$ do
7:             if $R_{uij} \neq 0$ then
8:                 Calculate $R_{uij}$ according to Eq. 6.14
9:             end if
10:            Get the time $t$ that the rating $R_{uij}$ has been given.
11:            Update $bu_i$, $bt_{uj}$, and $c_u$ according to Eqs. 6.16, 6.18, using $\gamma_u$, $\gamma_{bu_i}$, $\gamma_{bt_{uj}}$
12:            Update $c_u$ and $CI_t$ according to Eqs. 6.20, 6.22 using $\gamma_C$ and $\gamma_{CI_t}$
13:            $j \leftarrow j + 1$
14:        end for
15:    end for
16:    for $f \leq D$ do
17:        Update $P_{uf}^S$, $P_{uf}^{CS}$, and $\beta_{uf}^S$ according to Eqs. 6.24, 6.27 and 6.30 using $\gamma_P$, $\gamma_{Pf}$, and $\gamma_{uf}$
18:        Update $W_{uf}^S$, $W_{uf}^{CS}$, and $\beta_{uf}^S$ according to Eqs. 6.26, 6.29, and 6.32 using $\gamma_{Wf}$, $\gamma_{Wt}$, and $\gamma_{uf}$
19:        Update $Z_{uf}^S$, $Z_{uf}^{CS}$, and $\beta_{uf}^S$ according to Eqs. 6.28, 6.31, and 6.34 using $\gamma_{Zf}$, $\gamma_{Zt}$, and $\gamma_{uf}$
20:        if $v \in T_u$ then
21:            $f \leftarrow f + 1$
22:        end for
23:        for $f \leq D$ do
24:            Update $Y_{uf}^f$, and $Y_{uf}^{fS}$ according to Eq. 6.53 using $\gamma_Y$
25:            $f \leftarrow f + 1$
26:        end for
27:    end for
28:    end if
29:    $v \leftarrow v + 1$
30: end for
31: $u \leftarrow u + 1$
32: end for
33: }

Algorithm 3 Social Training

1: void SocialTrainer($\lambda$, $\gamma$)
2: {
3:     $u \leftarrow 1$
4:     for $u \leq N$ do
5:         $v \leftarrow 1$
6:         for $v \leq N$ do
7:             if $v \in T_u$ then
8:                 for $f \leq D$ do
9:                     Update $P_{uf}^S$, $W_{uf}^S$, and $Z_{uf}^S$ according to Eqs. 6.25, 6.28, 6.43 using $\gamma_P$, $\gamma_{Wf}$, and $\gamma_{Zf}$
10:                    Update $P_{uf}^{CS}$, $W_{uf}^{CS}$, and $Z_{uf}^{CS}$ according to Eqs. 6.26, 6.29, 6.40 using $\gamma_{Pf}$, $\gamma_{Wt}$, and $\gamma_{Zt}$
11:                    Update $\beta_{uf}^S$, $\beta_{uf}^{CS}$, according to Eqs. 6.31, 6.34, and 6.41 using $\gamma_{uf}$, $\gamma_{Wt}$, and $\gamma_{Zt}$
12:                    Update $\omega_{uf}$ according to Eq. 6.53 using $\gamma_w$
13:                    $f \leftarrow f + 1$
14:                end for
15:            end if
16:            $v \leftarrow v + 1$
17:        end for
18:     $u \leftarrow u + 1$
19:    end for
20: }

Algorithm 4 Model Updating

1: void ModelUpdater($\lambda$, $\gamma$)
2: {
3:     $v \leftarrow 1$
4:     for $v \leq N$ do
5:         $u \leftarrow 1$
6:         for $u \leq N$ do
7:             $f \leftarrow 1$
8:             for $f \leq D$ do
9:                 $P_{uf} \leftarrow -\gamma_U \times P_{uf}^S$
10:                $W_{uf} \leftarrow -\gamma_{uw} \times W_{uf}^S$
11:                $Z_{uf} \leftarrow -\gamma_{uw} \times Z_{uf}^S$
12:                $\beta_{uf} \leftarrow -\gamma_{uv} \times \beta_{uf}^S$
13:                $\omega_{uf} \leftarrow -\gamma_{uv} \times \omega_{uf}^S$
14:             end for
15:         end for
16:     end for
17: }

Chapter 6. Modelling Drifting Preferences in Recommender Systems 131
• The number of repetitions needed to update the parameters $y$ in line 20 is $D \times \sum_{u=1}^{N} |I_u|^2$.

• The number of repetitions needed to update the dependency matrix $Y$ in line 23 is $D^2 \times |R|$.

Therefore, the overall number of repetitions for the Intrinsic Trainer is obtained according to Eq. 6.58.

$$N(IntrinsicTrainer) = D^2 \times |R| + D \times \sum_{u=1}^{N} |I_u| \times |T_u| + 7 \times |R| + 10 \times D \times |R|$$

$$+ D \times \sum_{u=1}^{N} (|I_u| \times |T_u|) + D \times \sum_{u=1}^{N} |I_u|^2 + D^2 \times |R|$$

(6.58)

Assuming that on average, each user rates $c$ items, and trusts $k$ users, the computation time can be obtained as Eq. 6.59.

$$C(IntrinsicTrainer) = O(D^2 \times |R|) + O(D \times c \times |R|) + O(D \times k \times |T|)$$

(6.59)

Assuming that $c, k \ll N$, we can ignore the values of $c$ and $k$. Therefore, the computational time of the Intrinsic Trainer would be obtained according to Eq. 6.60.

$$C(IntrinsicTrainer) = O(D^2 \times |R|) + O(D \times |R|) + O(D \times |T|) = O(D^2 \times |R|) + O(D \times |T|)$$

(6.60)

Consequently, the overall computation time is linear with respect to the number of observed ratings as well as observed trust statements. Social Trainer consists of two loops that iterate over the non-zero trust relations in the adjacency matrix $T$. The number of repetitions needed to update the parameters $P$, $W$, $Z$, and $\beta^P$, $\beta^W$, and $\beta^2$ is $6 \times D \times |T|$. The number of repetitions to update the values of $Pt$, $Wt$, $Zt$, and $\omega$ is equal to $4 \times (\sum_{u=1}^{N} |I_u| \times |T_u| \times D)$. Therefore, the computation time of Social Trainer is equal to:

$$C(SocialTrainer) = O(D \times |R|) + O(D \times |T|)$$

(6.61)

In the Model Updater, the values of matrices $P$, $W$, $Z$, and vectors $\omega$, $\alpha^P$, $\alpha^W$, and $\alpha^2$ need to be updated. The computation time needed to update these parameters is
\( O(N \times D) \). Assuming that each user has rated at least one item, it is safe to say that \(|R|\) is greater than the number of users \(N\). Therefore, the computation time of Model Updater does not exceed the maximum computation time of Intrinsic Trainer and Social Trainer. Finally, the computation time of the Model trainer is obtained as Eq. (6.62).

\[
C(\text{ModelTrainer}) = O(D^2 \times |R|) + O(D \times |T|)
\]  

(6.62)

The number of latent factors \(D\) is fixed, hence the computation time is only a function of \(|R|\) and \(|T|\). Since both ratings matrix and social network matrix are sparse, the algorithm is scalable to the problems with millions of users and items.

### 6.3 Experiments

#### 6.3.1 Setup

We tested the proposed method on three popular datasets, Ciao, Epinions, and Flixster. In all the experiments in sections 6.3.3, 6.3.4, and 6.3.5, 80% of the datasets are used for training and the remaining 20% are used for evaluation. In order to achieve statistical significance, each model training is repeated 30 times and the average values are used. In Section 6.3.6, we analyse the behaviour of the models in other cases, where 60% and 40% of the ratings are used for training.

#### 6.3.2 Comparisons

In order to show the effectiveness of the proposed model, we compared the results against the recommendation quality of some of the most popular state of the art models that have reported the highest accuracies in the literature. These models include TrustSVD [31], CTFVSVD [111] proposed in Chapter 5, and TCTFVSVD, which is the model proposed in this chapter. Guo, Zhang and Yorke-Smith [32] carried out comprehensive experiments, and showed that their model, TrustSVD outperformed all the state of the art models. Recently, Zafari and Moser [111] showed that their model, CTFVSVD significantly outperforms TrustSVD. Therefore, in this section, we limited our comparisons to these two models from the state of the art, since they outperform a comprehensive set of state of the art recommendation models [32, 111].

The component-based approach that we took in designing this model enabled us to
arbitrarily switch on/off the dynamicity over different preference aspects (FV, F, and B). Therefore, in the experiments we try all the combinations of dynamic preference aspects. This results in 7 combinations denoted by B, B.F, B.F.FV, B.FV, F, F.FV, and FV.

The optimal experimental settings for each method are determined either by our experiments or suggested in previous studies [31, 32, 112]. Due to the over-fitting problem, the accuracy of iterative models improves for a number of iterations, after which it starts to degrade. Therefore, we recorded the best accuracy values achieved by each model during the iterations, and compared the models based on the recorded values. We believe that this approach results in a fairer comparison of the models than setting the number of iterations to a fixed value, because the models over-fit at different iterations, and using a fixed number of iterations actually prevents us from fairly comparing the models based on their real capacity in uncovering hidden patterns from data. Therefore, the reported results for iterative models here are the best results that they could achieve. MAE and RMSE measures are used to evaluate and compare the accuracy of the models. MAE and RMSE are used to measure and compare the performance of preference modelling methods in recommender systems. In the following sections, we consider the performances separately for All Users and Cold-start Users. Cold-start Users are users who have rated less than 5 items, and All Users include all users regardless of the number of items they have rated.

6.3.3 Discussion

All latent factor approaches have been evaluated with 5 factors, because no clear ideal value could be established. In Section 6.3.3.1 first we analyse the performance of the models from different perspectives. Since the results are subject to randomness, we also performed a t-test to guarantee that the out-performances achieved do not happen by chance. The results are discussed in Section 6.3.4. As we mentioned in Section 6.1 one of the research questions we are interested in, in this chapter is related to the interplay between the dynamicity of preference aspects and the preference domain. In Section 6.3.5 we consider the performance of combinations of TCTFVSVD, in order to pinpoint the aspects that are more subject to temporal drift in each dataset. In Section 6.3.6 we also consider the effect of the amount of training data that is fed to the model as input, and analyse the robustness of the models to the shortage of training data.
6.3.3.1 Model performances

We can consider the performance of the models from different perspectives. A preference model’s performance can be considered with respect to the dataset on which it is trained, the accuracy measure that is used to evaluate the model’s performance, and the performance of the model on cold-start users vs the performance on all users.

Datasets

The error values in Figures 6.5 through 6.7 show that TCTFVSVD results in substantial improvements over TrustSVD in all three datasets for both measures and for all users and cold-start users. As we can see in this figure, the box plots of TCTFVSVD’s combinations do not have much overlap with the box plot of TrustSVD, which means that the differences are definitely statistically significant. In this figure, we can also see that the box plot widths for TCTFVSVD’s combinations are usually much smaller than that for TrustSVD. This suggests that TCTFVSVD’s combinations are more stable than TrustSVD, meaning that they find roughly the same solutions across different model executions. This is a favourable property of the model, since it makes the model performance less subject to randomness. Clearly, a model that performs well sometimes and worse at other times is less reliable. The model’s superior performance is likely due to its taking multiple preference aspects into account, therefore, it has more clues as to where the optimal solutions might reside in the solution space.

In particular, we can see that the model is more stable in the case of the Ciao and Epinions datasets than the Flixster dataset. On the Epinions dataset, each typical user and cold-start user rates 41.61 items and 4.08 items on average. These numbers respectively are 15.94 and 2.94 for the Ciao dataset, and 11.12 and 1.94 for the Flixster dataset. This could explain why the variations are larger on Flixster dataset than Epinions and Ciao datasets. Since more ratings per user are available in the Ciao and Epinions datasets, different executions lead the model to more similar solutions than the solutions that are found on the Flixster dataset across different model executions. We can also see from Table 6.1, that on the Ciao and Flixster datasets, the improvements are more significant for RMSE, while less significant improvements are achieved for MAE. We can also clearly observe that the model variations are smaller for all users in the Epinions dataset, and for cold-start users in the Flixster dataset.

\[B\] stands for the combination with only bias, \(B.F\) stands for the combination with bias and features, \(B.F.FV\) stands for the combination with bias, features and feature values, \(F\) stands for the combination with only features, \(F.FV\) stands for the combination with features and feature values, and \(FV\) stands for
Accuracy measures

As the statistical analysis of the models in Table 6.1 show, the differences are generally more significant when the accuracies are measured in terms of the RMSE. This can be explained by the formulation of these models as an optimisation problem. These models focus on maximising accuracy using RMSE and achieving better MAE values is a secondary goal that is only pursued through minimising RMSE.

Cold-start vs all users

By taking a close look at the statistical analysis results in Table 6.1 and also the box plots of CTFVSVD vs TCTFVSVD’s combinations in Figure 6.5, we can see that in all three datasets, the improvements of the TCTFVSVD are more significant over all users than cold-start users. This can be explained by the amount of dynamic information that the models receive for each one of these groups of users. For all users, the model is trained using all ratings and also all associated time stamps for those ratings. Therefore the model the combination with only feature values
can more successfully discern the temporal patterns in the preferences, and the accuracy improvements are larger. However, for the cold-start users, the model does not have access to much temporal information about these users, since they do not have many ratings. As a result, the model cannot identify the shift in the preferences of these users, and the improvements are smaller. From this, we conclude that temporal models are more successful on all users, because for them, more temporal information is available.

6.3.4 Statistical analysis

The statistical analysis of the performances provided in Table 6.1 shows that all TCTFVSVSD’s combinations achieve significantly better results than TrustSVD, which does not include the temporal information. The values in Table 6.2 also show that TCTFVSVSD’s combinations result in improvements over CTFVSVSD that are statistically significant, which means that in all three datasets, TCTFVSVSD has been successful in extracting the temporal patterns in the users’ preferences. We can also see that the all the p values in Table 6.1 are 0.0000, which means that with almost 100% probability, the two model
Figure 6.7: Box plots of the TCTFVSVd’s combinations and CTFVSVd versus TrustSVd on the Flixster dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL)

executions (TCTFVSVd and TrustSVd) do not come from distributions with equal mean performances. Therefore, we are almost 100% sure that the observed differences in performance are due to the superiority of TCTFVSVd over TrustSVd, and not the result of chance. Similarly, the p values in Table 6.2 are almost zero, which means that we are certain that TCTFVSVd is better than CTFVSVd, in cases where the t-test shows a statistically significant improvement.

6.3.5 Dynamic aspects

The comparison of the error values achieved by TCTFVSVd in Figure 6.8 show that in terms of MAE for all users, TCTFVSVd achieves the best performance on the Ciao and Epinions datasets, for the models including dynamic bias ($B$) and features ($F$) aspects. However, on the Flixster dataset, the model combination with dynamic $B$ and $FV$ aspects performs best. Interestingly, for cold-start users, the model that performs best varies. In particular, on the Ciao dataset, the model including dynamic $F$ performs best, whereas on the Epinions and Flixster datasets, the model including dynamic $B$, $F$, and feature
values \((FV)\) aspects, and the model with drifting \(F\) aspect achieve the best results respectively. Furthermore, the error values in Figure 6.9 show that different model combinations might achieve the best performances for RMSE. From these figures, we can make several conclusions.

The first conclusion is that the dynamic patterns are dataset-dependent. Therefore, users and the items in different dataset can have preferences with aspects with different levels of dynamicity. This finding supports the choice of a component-based approach in modelling the dynamic properties of the preference aspects.

The second conclusion is that the prediction of the ratings for the cold-start users is less dependent on the drifting bias than that of all users. As we see in these Figures 6.8 and 6.9 for all users, the combinations that include dynamic \(B\) aspects are strictly better than the other combinations, whilst this is less consistent for cold-start users, where sometimes the models with only dynamic \(F\) aspects perform best. This suggests that the preferences of cold-start users are not much affected by the shifts in the popularity of the items, while other users’ preferences are more influenced by such shifts. Therefore, the accurate modelling of such temporal effects is of greater importance in the case of all users than cold-start users. As previous studies have shown [47], bias is a very important aspect in human preferences. Since the cold-start users do not have enough ratings, there is also not enough temporal data to train the preferences for these users. Therefore, the temporal aspects trained for these users are probably not very accurate, and as a result, the combinations that include bias perform poorly on these users, due to imprecise predictions.

To summarise, it is very advantageous to have a component-based model in which the temporal aspects of preferences can be arbitrarily captured in different conditions. This enables us to capture the patterns only when they are actually helpful, and consequently, build the most accurate preference models, tailored to different datasets and domains with varying temporal patterns.

6.3.6 Effect of the size of the training dataset

The main purpose of this section is to evaluate the robustness of the models against shortage of training data. In the experiments in sections 6.3.3 through 6.3.5 80% of the ratings matrix was used for training the models and the remaining data was used for evaluation. The question that arises here is how the models would perform if less amount of data was fed to the models for training.

In order to analyse the behaviour of the models with respect to the amount of training
Table 6.1: The t values and p values for TCTFVSVSVD’s combinations vs TrustSVD in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS)

As can be seen in Figure 6.10 on the Flixster dataset, in the case of all users, for all combinations of TCTFVSVSVD, the increase in error is less when the data is decreased from 80% to 60% (denoted by 80-60 in these diagrams), compared to when it drops from 60% to 40% (denoted by 60-40 in these diagrams).

Furthermore, we can observe that in terms of MAE, the combination that includes \( F \) and \( FV \) resulted in the smallest error increase when the training data decreased from 80%...
Table 6.2: The t values and p values for TCTFVSV’s combinations vs CTFVSV in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS)

to 60%, and the model that included FV resulted in the smallest error increase when the training data decreased from 60% to 40%. This observation suggests that the dynamic model is more robust to the shortage of training data, when the error is measured in terms of MAE for all users. In terms of RMSE, the least accuracy deterioration happened for the model combination with the F aspect, both when the training data amount drops to 60%, and when it drops to 40%.

Cold-start users

For cold-start users however, a different pattern is evident. Interestingly, we can see that for cold-start users, the error increases more when the training data is decreased from 80% to 60%, compared to when it is decreased from 60% to 40%. This means that the accuracy degrades more when the training data drops to 60%. Judging by the higher error increase for cold-start users in comparison with all users, cold-start users seem to be more sensitive to the decrease in the amount of training data. This seems understandable, since the cold-start users do not have many ratings. Therefore, when evaluating the model accuracy for
cold-start users, less accurate predictions for each rating have a larger effect on the overall accuracy.

TrustSVD seems to be more robust to the shortage of training data for cold-start users, when the training data drops from 60% to 40%. This can be attributed to the fact that the dynamic model contains time information, and this information can be misleading if we
substantially decrease the amount of training data, and evaluate the accuracy for cold-start users who do not have much ratings. A similar observation was made in Figures 6.8 and 6.9, where the dynamic model including the B aspect performed poorly on the cold-start users.
Figure 6.10: Effect of the training amount on error on Flixster dataset for a) MAE for all users, b) MAE for cold-start users, c) RMSE for all users, d) RMSE for cold-start users (80-60 and 60-40 respectively show the percentages of error increase when the training data amount drops from 80% to 60%, and from 60% to 40%)

All users vs cold-start users

A similar trend to the one observed in Flixster dataset can also be seen in the Ciao dataset in Figure 6.11. As this figure shows, the accuracy deterioration for cold-start users is much larger compared with that for all users. Again, we attribute this to the high sensitivity of cold-start users to inaccurate predictions. For the case where the training data amount drops from 80% to 60%, the model combination with all the dynamic aspects (B.F.FV) results in the lowest increase in MAE for all users. For cold-start users, the model combination with $B$ and $F$ aspects achieve the smallest deterioration of accuracy.

However, in terms of RMSE for all users, TrustSVD incurs the lowest increase in the error, while for cold-start users, the model with the dynamic $FV$ aspect is the most robust. In the second case where the training data amount is decreased from 60% to 40%, at least one of the model combinations performs best (incurs the lowest accuracy deterioration) for each measure, among the models tested. We can also see that when the training data amount is decreased from 80% to 60%, the error increase is much lower than when the
training data amount drops from 60% to 40%. This means that the models are still quite robust with 60% of the ratings data as training data, but their accuracy considerably drops when the training data decreases to 40%.

**Flixster vs Ciao**

One of the key differences between the behaviour of the models on the Flixster and Ciao datasets, as can be seen in Figures 6.10 and 6.11, is the point at which the accuracy sharply drops for cold-start users (60% vs 40%). For the Flixster dataset, the accuracy of cold-start users sharply worsens when the training data amount is decreased from 80% to 60%, while for the Ciao dataset, the sharp decrease in accuracy happens when the training data amount decreases from 60% to 40%. This can be easily justified by looking at the statistics of these two datasets for cold-start users. On the Flixster dataset, each cold-start user rates 1.94 items on average, while this number is 2.94 in the Ciao dataset. Therefore, the accuracy of cold-start users on the Flixster dataset is more sensitive to inaccurate predictions than that on the Ciao dataset.
Considering all four measures on the two datasets, in general, we can observe that TCTFVSVD’s combinations are more robust to the decrease in the amount of training information than TrustSVD and CTFVSVD.

**Insights**

From the observations for cold-start users, we can conclude that in order for the time information to be helpful, we need to provide the model with enough time-related data as input, so that the accuracy can be improved. The importance of such data is more pronounced for cold-start users, whose predictions are more sensitive to the inaccuracies. Otherwise, if the amount of training data is insufficient, the model can learn misleading temporal patterns that directly result from a shortage of training information.

We also saw that the degree of deterioration of the accuracy when the training data amount is decreased is somewhat dependent on the dataset. On Flixster, the accuracy degrades by somewhere between just under 1% to just under 5%. On Ciao, however, the accuracy deteriorates much more (roughly between 6.5% and 19.5%). Therefore, it is up to the system users to decide whether they would like to use smaller datasets and sacrifice the accuracy, or spend more time on training more accurate models using more information.

We did not observe any tangible differences between the execution times of these cases (80%-60%-40%), and the computational complexity analysis of the model in Section 6.2.5 showed that the model time is of linear order. Therefore, it is probably advisable for the system owners to use as much data as available to achieve the highest accuracies, as long as their computational limitations allow.

### 6.3.7 Run time analysis

We also measured the run time of state of the art models and compared it with that of TCTFVSVD. For this end, we used a regular PC with 8GB of RAM, and 3.20GHz Core i5-4570 Intel CPU. First, we analysed how the run time of TCTFVSVD changes by increasing the number of factors in the Ciao dataset. The results in Figure 6.12 show that the run time on the ciao dataset does not change much as the number of factors increases from 5 to 35. However, when we increased the number of factors to 50, the run time significantly increased due to memory deficiency. Seems the model’s run time with respect to the number of factors is more restricted by memory than computation.

Then, we ran TCTFVSVD on the Ciao and Epinions datasets using 5 factors for 10 iterations. Each experiment was repeated for 5 times and average run times were
Chapter 6. Modelling Drifting Preferences in Recommender Systems

Figure 6.12: Run time of TCTFVSVD with respect to the number of factors on the Ciao dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>BiasedMF</th>
<th>PMF</th>
<th>SocialMF</th>
<th>SoReg</th>
<th>SVD++</th>
<th>TCTFVSVD</th>
<th>TrustMF</th>
<th>TrustSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ciao</strong></td>
<td>39</td>
<td>64</td>
<td>10368</td>
<td>516</td>
<td>653</td>
<td>73999</td>
<td>497</td>
<td>1379</td>
</tr>
<tr>
<td><strong>Epinions</strong></td>
<td>5663</td>
<td>5139</td>
<td>87100</td>
<td>9512</td>
<td>90110</td>
<td>3372502</td>
<td>18143</td>
<td>117243</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of average run times for different models on the Ciao and Epinions datasets in terms of milliseconds

calculated for each model. The results in Table 6.3 and Figures 6.13a and 6.13b show that TCTFSVD needs more time to model the preference aspects. This is expected, since this model captures 6 aspects, while the other models capture only 1 or 2 aspects. Training the model on the Epinions dataset with about 600K ratings takes less than 1 hour. Although this number is significantly higher than the training times for other models, it is still easily applicable in real-world applications. This model can catch up with the ratings traffic, as long as the user ratings given to the items do not exceed 10000 rates per minute, which is very unlikely to happen even in high traffic web sites. The users would need to rate about 14 million items in one day for this to happen. If case this happened, the model’s performance might degrade because it would miss some ratings as it would be busy with training.
6.4 Summary of key points

1. The drifting nature of preferences has been emphasized in the literature as a specific type of concept drift. The most obvious example in the movies domain is how our preference shifts from action movies to drama movies as we age.

2. Existing models with dynamic preferences do not model preference aspects such as conditional dependencies and preferences over feature values. As a result, the dynamic properties of these aspects are also overlooked.

3. The dynamic characteristics of these preferences can be efficiently modelled by assuming specific functions (e.g. linear, exponential, etc.) over these aspects, and adding parameters to latent factor models to train the parameters of those functions.

4. Analysing computational complexity of the algorithm shows that this approach can efficiently learn the dynamic preference patterns. Furthermore, run time analysis proves the practicality of this method in real-world applications.

6.5 Conclusion

In this chapter, we addressed the problem of modelling the temporal properties of human preferences in recommender systems. In order to tackle this problem, we proposed a novel latent factor model called TCTFVSVD. TCTFVSVD is built on the basis of CTFVSVD, a model that we proposed in Chapter 5 in order to capture socially-influenced conditional preferences over feature values. In TCTFVSVD, three major preference aspects were assumed to be subject to temporal drift. These aspects include user and item biases,
preferences over features, and preferences over feature values. Moreover, we also analysed
the temporal behaviour of each of these preference aspects and their combinations. We
also considered the robustness of TCTFVSVD’s combinations with respect to the shortage
of training data.

In order to evaluate the model, we carried out extensive experiments on three pop-
ular datasets in the area of recommender systems. We considered the model errors in
terms of MAE and RMSE measures on all users and cold-start users. We also performed
statistical analyses on the performances observed, to make sure that the differences in
accuracies are significant, and hence do not happen by chance. The experiments revealed
that in all three datasets, all combinations of TCTFVSVD for both measures on all users
and cold-start users significantly outperformed TrustSVD, which had proven to be the
most accurate static social recommendation model before CTFVSVD. The experiments
also proved that most of the TCTFVSVD’s combinations were significantly more accu-
rately than CTFVSVD. In particular, we found that TCTFVSVD with all dynamic aspects
outperformed CTFVSVD in all three datasets on all users.

The analysis of the temporal behaviour of preference aspects and their combinations on
the three datasets showed that different datasets included different temporal patterns, and
therefore, required models with different dynamic aspects. This supported our component-
limited approach in modelling the basic preference aspects and their temporal properties.
We also concluded that the dynamic models are more helpful in cases where there is
enough training data to discern the temporal properties. In particular, we concluded that
the models proposed in this chapter are more successful in modelling all users, because
more time-related data is available for all users than cold-start users, and therefore the
temporal characteristics were extracted more accurately. The analysis of the robustness of
the models with respect to the shortage of training data also revealed that TCTFVSVD
was in general more robust than CTFVSVD and TrustSVD. The models were also more
robust for all users than cold-start users, because cold-start users were more sensitive to
the inaccurate predictions.
Chapter 7

A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

7.1 Introduction

In Chapter 2 we identified five major aspects to the preferences besides feature preferences that are already modelled by the latent factor models. These aspects included feature value preferences (FVP), social influence (SI), temporal dynamics (T), conditional preferences (CP), and user and item biases (B). In Chapter 6, we showed how we integrated all these aspects into a unified model called TCTFVSVD.

In this chapter, we show that different interactions actually exist between preference aspects, which necessitates the designing of a unified component-based model integrating all the preference aspects in order to achieve the highest accuracy. Although these aspects have been extensively researched individually, to the best of our knowledge, there is no study that integrates all these aspects into a unified model. Moreover, the interaction effects of these aspects have been disregarded mostly due to the lack of a model that integrates all of these aspects. Therefore, in this chapter, we address the question, how do the preference aspects interact with each other, and in particular, are these interactions specific to the preference dataset or domain? Answering this question would enable the research community to design more accurate recommender systems and possibly tailor them to the specific needs of a domain.

The rest of this chapter is organised as follows: In Section 7.2.1, we briefly explain the evolutionary algorithm we used to optimise the hyper-parameters and learning rates.
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

in TCTFVSVD and then in Section 7.2.2 we introduce a hybrid method that achieves optimised preference models for every combination of preference aspects. In Section 7.3, we first explain the experimental setup in Section 7.3.1, and then report the accuracy results and analyse the interactions in Section 7.3.2. We conclude the chapter in Section 7.5 by summarising the main findings.

7.2 Adaptive training of preference aspects and interactions

One of the problems faced in probabilistic matrix factorisation is that if the regularisation parameters are not tuned carefully, the model is prone to over-fitting because it finds a single point estimate of the parameters. Therefore, in basic matrix factorisation, it is assumed that the regularisation parameters are known in advance, and they are fed into the model as model inputs. In order to address the problem of finding the optimal parameters, Salakhutdinov and Mnih extended PMF and proposed BPMF, by assuming Gaussian-Wishart priors on the user and item regularisation parameters. In BPMF these parameters are also learnt along with other model parameters. However, the proposed approach increases the computational complexity. Furthermore, our experiments show that this method provides less accurate recommendations than some other methods. We found that the performance of TCTFVSVD depends to a large extent on fine-tuning the hyper-parameters and learning rates for each of the preference aspects. To optimise the hyper-parameters and learning rates, we employ a hybrid method combining an Evolutionary Algorithm (EA) and Gradient Descent (GD). In the following sections, we first briefly introduce Evolutionary Algorithm (EA), and then explain the hybrid method to learn the preference aspects and interactions. This method is abbreviated to ATCTFVSVD (Adaptive TCTFVSVD).

7.2.1 EA

To optimise the learning rate and regularisation parameter of each model aspect, we applied an evolutionary algorithm (EA) with real encoding, Gaussian mutation and single-point crossover. The flow chart for this method is illustrated in Figure 7.1.

The parent population was chosen using binary tournament selection. The algorithm receives its feedback from the error value resulting from the designed matrix factorisation model. The algorithm begins with the uniformly random initialisation of the current population (P). Then three empty sets of solutions, Offspring Set (OP), Parents Set (PP), and Children Set (CP) are created. Then M solutions from the set P are selected according to
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

Figure 7.1: The schematic representation of ATCTFVSVD
the tournament selection method, and added to PP. The parent solutions in PP are evolved and added to CP, and then all the solutions in CP are added to OP, and this process is repeated until the size of this set reaches the maximum number of offspring. Then the solutions in OP are evaluated and all solutions in P are also added to OP. The solutions in OP are then truncated (to the maximum size of P) according to a truncation strategy, and all solutions in P are replaced by the truncated solutions in OP. Now P contains the next generation of solutions. This process is repeated until a stopping criteria is met, and finally the best solution in P is chosen as the optimal solution and returned by the algorithm. This best solution includes the optimal hyper-parameters and learning rates for the TCTFVSVD. Optimising the matrices repeatedly with hyper-parameters (regularisation factors) and learning rates produced by the EA is computationally expensive. For the hyper-parameter optimisation, a representative subset of 10%, 3%, and 1% of the matrix entries depending on the dataset were used, sampled uniformly randomly. The interactions between the EA and the matrix optimisation are illustrated in Figure 7.2. In this figure, HP is acronym for hyper-parameters, and LR for learning rates. As we explained in Chapter 6, parameters are the matrices and vectors that are trained using TCTFVSVD. The training of these matrices and vectors are done with different learning rates.

![Figure 7.2: The schematic representation of ATCTFVSVD](image-url)

Figure 7.2: The schematic representation of ATCTFVSVD
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

7.2.2 ATCTFVSVD

Having obtained the near-optimal values for the learning rates and regularisation parameters of the model using a subset of the matrix entries, we can optimise the matrices by applying gradient descent repeatedly on the complete datasets using the pre-optimised hyper-parameters. In recommender systems, there are usually thousands of users who show their preferences over thousands of items. Therefore, the constructed model includes millions of parameters to optimise. Most of the latent factor models employ gradient descent to optimise the model parameters, because of its low computational complexity and scalability. This method is applicable to the problems in which the solution space is differentiable, and particularly suitable to the problems that include many parameters to optimise. The popularity of this method stems from its efficiency and the high quality solutions that it can find at a reasonable time. In chapters 5 and 6, we observed that recommender systems achieve their best accuracies when different aspects, represented by matrices, capture different aspects of a dataset. It is intuitively clear that the aspects of the model interact; for example, we can assume that users’ preferences change over time.

7.2.3 Analysing the interaction effects using ATCTFVSVD

In order to evaluate ACTCFVSVD and determine the interactions of the preference aspects, we train separate models for different aspect combinations. As explained in Section 6.2, preferences are comprised of 5 major aspects besides feature preferences captured by basic latent factor models. Considering different combinations of the aspects, 32 preference models are possible. In order to analyse the interactions of the aspects, we first need to obtain the optimal hyper-parameters and learning rates using ATCTFVSVD as in Figure 7.2. This gives us 32 sets of hyper-parameters and learning rates, each for one combination of aspects. Then we run TCTFVSVD using the optimal hyper-parameters and learning rates obtained using ATCTFVSVD and evaluate the error of each combination of aspects.

Optimising the model hyper-parameters and learning rates is an important step in analysing the interactions and deciding whether an aspect or an interaction is helpful. As an aspect is switched off, the optimal hyper-parameters and learning rates for each aspect changes. Therefore, to be able to analyse the interactions, we first need to optimise the hyper-parameters and learning rates, and make sure that all aspects are trained in the best way possible when some aspects are switched off. Then if an aspect shows not to be helpful in the experiments, we have already ruled out the possibility that maybe the aspect was not properly trained using the default hyper-parameters and learning rates.
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

7.3 Experiments

7.3.1 Experimental setting

We tested ATCTFVSVFD on three popular datasets, Ciao, Epinions, and Flixster. In order to reduce the computational costs, we applied the algorithm to a uniformly randomly selected subset of the users and items. These reduced matrices retain the original density and are a representative sample. In all the experiments, 80% of the datasets are used for training and the remaining 20% are used for evaluation. Each model training is repeated for 30 times and the average values are used.

7.3.2 Results

The MAE values and RMSE values obtained on the Ciao, Epinions, and Flixster datasets are shown in the Table 7.1. We also executed TrustSVD and applied t-test to find out whether the differences between the model combinations and TrustSVD are significant. TrustSVD has yielded the highest accuracy among a large set of state of the art models. The MAE and RMSE values on Ciao dataset for TrustSVD are respectively obtained as 1.1508 and 1.3969. These values are 0.9242 and 1.1912 for the Epinions, and 0.7910 and 0.9932 for Flixster.
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

As Table 7.1 shows, the helpfulness of the aspects seems to be dependent on the dataset. To be able to advise the practitioners on how to choose the aspects, an investigation into the practical quality of the evaluation metrics in recommender systems is needed, to find out which measure better translates into more revenue for the business. This is highly dependent on how the measures are correlated with business metrics, as well as the business model of the company. For example, if the company has a subscription-based business model, where the users need to pay a monthly subscription fee to use the services, the

<table>
<thead>
<tr>
<th>Model</th>
<th>MAE Ciao</th>
<th>MAE Epinions</th>
<th>MAE Flixter</th>
<th>RMSE Ciao</th>
<th>RMSE Epinions</th>
<th>RMSE Flixter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature preferences only</td>
<td>1.4275</td>
<td>0.8919</td>
<td>0.8425</td>
<td>1.6772</td>
<td>1.1912</td>
<td>1.0687</td>
</tr>
<tr>
<td>B</td>
<td>1.1032</td>
<td>0.9213</td>
<td>0.7802</td>
<td>1.3725</td>
<td>1.1924</td>
<td>0.9947</td>
</tr>
<tr>
<td>T</td>
<td>1.4252</td>
<td>0.8903</td>
<td>0.8426</td>
<td>1.6774</td>
<td>1.1918</td>
<td>1.0729</td>
</tr>
<tr>
<td>T B</td>
<td>1.1086</td>
<td>0.9219</td>
<td>0.7812</td>
<td>1.3733</td>
<td>1.1929</td>
<td>0.9933</td>
</tr>
<tr>
<td>SI</td>
<td>1.4152</td>
<td>0.8923</td>
<td>0.8441</td>
<td>1.6698</td>
<td>1.1921</td>
<td>1.0722</td>
</tr>
<tr>
<td>SI B</td>
<td>1.1116</td>
<td>0.9219</td>
<td>0.7774</td>
<td>1.3731</td>
<td>1.1924</td>
<td>0.9926</td>
</tr>
<tr>
<td>SI T</td>
<td>1.4225</td>
<td>0.8932</td>
<td>0.8413</td>
<td>1.67</td>
<td>1.193</td>
<td>1.0702</td>
</tr>
<tr>
<td>SI T B</td>
<td>1.1215</td>
<td>0.9231</td>
<td>0.7823</td>
<td>1.3815</td>
<td>1.1947</td>
<td>0.9931</td>
</tr>
<tr>
<td>CP</td>
<td>1.3965</td>
<td>0.8923</td>
<td>0.8457</td>
<td>1.6506</td>
<td>1.1897</td>
<td>1.0698</td>
</tr>
<tr>
<td>CP B</td>
<td>1.1362</td>
<td>0.9196</td>
<td>0.7807</td>
<td>1.4458</td>
<td>1.1898</td>
<td>0.9917</td>
</tr>
<tr>
<td>CP T</td>
<td>1.3753</td>
<td>0.8926</td>
<td>0.8448</td>
<td>1.6239</td>
<td>1.1921</td>
<td>1.0701</td>
</tr>
<tr>
<td>CP T B</td>
<td>1.1079</td>
<td>0.9193</td>
<td>0.7784</td>
<td>1.3714</td>
<td>1.19</td>
<td>0.9895</td>
</tr>
<tr>
<td>CP SI</td>
<td>1.3868</td>
<td>0.8938</td>
<td>0.8463</td>
<td>1.6279</td>
<td>1.1918</td>
<td>1.0705</td>
</tr>
<tr>
<td>CP SI B</td>
<td>1.1234</td>
<td>0.9226</td>
<td>0.7799</td>
<td>1.3843</td>
<td>1.1946</td>
<td>0.9922</td>
</tr>
<tr>
<td>CP SI T</td>
<td>1.356</td>
<td>0.8937</td>
<td>0.8504</td>
<td>1.6101</td>
<td>1.1918</td>
<td>1.0706</td>
</tr>
<tr>
<td>CP SI T B</td>
<td>1.2487</td>
<td>0.9224</td>
<td>0.7814</td>
<td>1.4996</td>
<td>1.1926</td>
<td>0.9931</td>
</tr>
<tr>
<td>FVP</td>
<td>1.3702</td>
<td>0.8895</td>
<td>0.8141</td>
<td>1.6204</td>
<td>1.1883</td>
<td>1.0501</td>
</tr>
<tr>
<td>FVP B</td>
<td>1.1154</td>
<td>0.9226</td>
<td>0.7774</td>
<td>1.3781</td>
<td>1.1925</td>
<td>0.9944</td>
</tr>
<tr>
<td>FVP T</td>
<td>1.3525</td>
<td>0.8905</td>
<td>0.8136</td>
<td>1.6006</td>
<td>1.1862</td>
<td>1.0438</td>
</tr>
<tr>
<td>FVP T B</td>
<td>1.1223</td>
<td>0.9182</td>
<td>0.7776</td>
<td>1.3833</td>
<td>1.1896</td>
<td>0.9948</td>
</tr>
<tr>
<td>FVP SI</td>
<td>1.363</td>
<td>0.8912</td>
<td>0.8143</td>
<td>1.6118</td>
<td>1.1856</td>
<td>1.0516</td>
</tr>
<tr>
<td>FVP SI B</td>
<td>1.1194</td>
<td>0.9214</td>
<td>0.7733</td>
<td>1.3826</td>
<td>1.1936</td>
<td>0.9945</td>
</tr>
<tr>
<td>FVP SI T</td>
<td>1.3368</td>
<td>0.8922</td>
<td>0.8201</td>
<td>1.6051</td>
<td>1.1897</td>
<td>1.0586</td>
</tr>
<tr>
<td>FVP SI T B</td>
<td>1.1118</td>
<td>0.9201</td>
<td>0.7754</td>
<td>1.3746</td>
<td>1.1918</td>
<td>0.9953</td>
</tr>
<tr>
<td>FVP CP</td>
<td>1.3475</td>
<td>0.8947</td>
<td>0.8125</td>
<td>1.5907</td>
<td>1.1927</td>
<td>1.0466</td>
</tr>
<tr>
<td>FVP CP B</td>
<td>1.1147</td>
<td>0.918</td>
<td>0.7803</td>
<td>1.3773</td>
<td>1.1896</td>
<td>0.9944</td>
</tr>
<tr>
<td>FVP CP T</td>
<td>1.3386</td>
<td>0.8904</td>
<td>0.8169</td>
<td>1.5803</td>
<td>1.1857</td>
<td>1.0514</td>
</tr>
<tr>
<td>FVP CP T B</td>
<td>1.1217</td>
<td>0.9207</td>
<td>0.7809</td>
<td>1.3816</td>
<td>1.1932</td>
<td>0.9936</td>
</tr>
<tr>
<td>FVP CP SI</td>
<td>1.3025</td>
<td>0.8933</td>
<td>0.8218</td>
<td>1.567</td>
<td>1.188</td>
<td>1.0557</td>
</tr>
<tr>
<td>FVP CP SI B</td>
<td>1.1086</td>
<td>0.919</td>
<td>0.7779</td>
<td>1.3768</td>
<td>1.1893</td>
<td>0.9932</td>
</tr>
<tr>
<td>FVP CP SI T</td>
<td>1.3081</td>
<td>0.8931</td>
<td>0.8183</td>
<td>1.579</td>
<td>1.1878</td>
<td>1.0553</td>
</tr>
<tr>
<td>FVP CP SI T B</td>
<td>1.1021</td>
<td>0.9192</td>
<td>0.7762</td>
<td>1.3738</td>
<td>1.1881</td>
<td>0.9964</td>
</tr>
</tbody>
</table>

Table 7.1: The results of model combinations in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures. The bold values represent significant differences between the model combination and TrustSVD using t-test ($p \leq 0.05$), and red and blue values respectively mean performing better or worse than TrustSVD. The best performances for the model combinations in each dataset are underlined. Black values mean there is no significant difference.

As Table 7.1 shows, the helpfulness of the aspects seems to be dependent on the dataset. To be able to advise the practitioners on how to choose the aspects, an investigation into the practical quality of the evaluation metrics in recommender systems is needed, to find out which measure better translates into more revenue for the business. This is highly dependent on how the measures are correlated with business metrics, as well as the business model of the company. For example, if the company has a subscription-based business model, where the users need to pay a monthly subscription fee to use the services, the
company can benefit indirectly from improving, say, RMSE of the recommendation model in the following way. Lower RMSE values could be correlated with more matches between users and company’s services, which could result in increasing the company’s popularity, and consequently more users purchasing subscription plans.

In terms of MAE, FVP seems to be mostly helpful in all three datasets, since it appears in most of the best combinations. This means that the users in these datasets have different preferences over values of the features, and hence, the model can benefit more by modeling FVP. B is not helpful in the Epinions while it is helpful on the other datasets. Therefore, training B in the Epinions only adds noise and reduces the accuracy. The system practitioners are advised not to train B, if their users are not much biased in ratings or the products they offer do not get biased ratings.

Time seems to be generally helpful in the Ciao dataset since the best accuracy is achieved with a dynamic model. This could imply that the users in these datasets have more dynamic preferences than Epinions and Flixster. If the users of a company tend to have more stable preferences, training T aspect is not recommended. Similarly, SI seems not to be helpful in the Epinions as the best combination does not model SI. This could be because Epinions users are probably more reliant on their own preferences than getting influenced by friends. Again, whether the system owners should train the SI aspect depends on whether the users are self-reliant or socially-suggestible. Intuitively, if the users are not much affected by social friends, the preference data would not include patterns about social influence. Therefore, training social influence would only add noise to the model and would deteriorate the accuracy, since the model would be trying to capture a pattern that does not exist in data. The system practitioners are advised to analyse the attributes of their customers before applying the model, and then decide which aspects they should model to attain the best accuracy. For example, if their customers seem to be more active in social media, training the social influence aspect can boost the performance of the model. Otherwise, it would just add noise to the model and degrade the accuracy. To analyse the customers, the system administrators could arbitrarily switch off one or more aspects as they periodically update the model, and monitor the accuracy. Based on the model’s response, they could get more understanding of their customers and draw conclusions as to what aspects probably exit in their users’ preference data.

The accuracies obtained by ACTCFVSVSD in the three datasets presented in Table 7.1 are also depicted in Figure 7.3. Several interesting observations and conclusions can be made from these two figures. We observe that interactions on the Epinions dataset are
totally different from the interactions on the Ciao and Flixster. For example, addition of B on the Ciao and Flixster improves the accuracy, while it worsens the accuracy on the Epinions. From this observation, we can conclude that interactions are dataset-dependent. We also observe that overall, adding more aspects on Ciao and Flixster improves the accuracy, while on the Epinions, the error remains almost the same. This means that capturing all aspects is not necessarily a good idea for all datasets. Another observation is that similar patterns are evident for MAE and RMSE measures. For example, the effect of adding T to the model that only includes FVP is similar for both measures. Therefore, the interactions and the helpfulness of aspects does not seem to be much dependent on the measures used. However, as we mentioned before, the practical quality of these measures for businesses could be different.

In general, trying to capture an aspect that does not exist degrades the accuracy of TCTFVSVD. The reason is that the model works based on the assumption that those patterns exist and some dimensions in the solution space are defined for optimising that aspect. For example, having the prior knowledge that the users are not biased in a dataset would simplify the solution space, and would let the model carry out the search in areas of solution space where the promising solutions actually reside. The model’s component-based design enables us to incorporate such prior knowledge, by switching off that aspect and removing the associated dimensions from the solution space. Similarly, interactions between the aspects can also exist depending on dataset. For example, the users in a dataset might have different preferences about feature values, but have preferences. In this case, switching on T while FVP is present might actually worsen the accuracy. Similarly, conditional dependencies might exist between features. However, the users might not have different preferences over feature values. Therefore, addition of FVP to CP might make the accuracy worse instead of improving it. Therefore, understanding the helpfulness of aspects and their interactions is practically valuable for system owners in achieving the best performances given the data in hand.
Figure 7.3: The results of model combinations for a) MAE and b) RMSE
Chapter 7. A Hybrid Methodology for Analysing the Interaction Effects of Preference Aspects in Recommender Systems

7.4 Summary of key points

1. The existing interactions between preference aspects require a component-based model design that can arbitrarily switch off preference aspects in situations they are not helpful.

2. We achieved such design by taking a component-based approach and incorporating different sets of latent parameters that were responsible for different aspects.

3. Latent factor model hyper-parameters for each combination of aspects can be adjusted using a revolutionary method such as genetics algorithm.

4. The experiments on two dataset confirm the hypothesis that interactions between preference aspects exist, and that these interactions depend on the preference dataset.

7.5 Conclusion

In this chapter, we proposed a novel hybrid latent factor model, ATCTFVSVD, that incorporated different aspects of preferences, i.e. user and item biases, feature preferences, feature value preferences, conditional dependencies, social influence, and preference dynamicty. ATCTFVSVD extended TCTFVSVD that was proposed in Chapter 6 and employed an Evolutionary Algorithm to optimise the model hyper-parameters and the learning rates to be used in Gradient Descent. Then using three popular datasets, we showed that the proposed method achieves significantly better results than TrustSVD, which has shown superior accuracy over a large set of the state of the art recommender system.

We further analysed the interaction effects between the preference aspects in those three datasets. We concluded that different types of interactions may exist between different aspects in different datasets. This finding emphasised the importance of designing a component-based approach in preference modelling, which enables understanding of these interaction effects in different datasets and domains, in order to achieve the highest accuracy. In particular, we showed that the time aspect was only helpful in Ciao dataset, but not in Epinions and Flixster datasets. Moreover, bias and social influence aspects were not helpful in Epinions dataset, and they were only useful in Ciao. We speculated that this originated from the differences between the users of these websites. The system practitioners are advised to analyse the attributes of their customers before applying the model, and then decide which aspects they better model to attain the best accuracy. For example, if their customers seem to be more active in social media, training the social influence aspect
can boost the performance of the model. Otherwise, it might just add noise to the model and degrade the accuracy. The system administrators could arbitrarily switch off one or more aspects as they periodically update the model, and monitor the accuracy. Based on the model’s response, they could get better understanding of their customers and find out what aspects probably exit in their users’ preference data.
Chapter 8

An Integrated Architecture for Preference Model Representation and Explanation

8.1 Introduction

Recommender systems usually perform as black boxes which present a user with a preference ranking of choices, which may not be perceived as trustworthy by a user. To help the user understand the recommendations, researchers have proposed adding explanation facilities to recommender systems [19]. A number of benefits of recommendation explanations have been suggested by researchers in the literature, such as efficiency and effectiveness by enabling better and faster decisions, trust, user satisfaction, persuasiveness, transparency, and scrutability [19] [118] [24] [89]. Explaining the recommendations is particularly important in high investment product domains (e.g. digital cameras, laptops), where consumers try to avoid financial risk and appreciate help with making good decisions [17].

Nilashi et al. [73] proposed a new trust model and assessed the relative importance of the model factors for trust-building. Using empirical studies on two popular electronic commerce websites, they showed that transparency is equally important for trust building as recommendation quality and conclude that focusing merely on recommendation quality may be insufficient. Similarly, Quijano-Sanchez [77] proposed a personalised social individual explanation approach for group recommendations, which were found to significantly improve the user’s satisfaction and likelihood of following the recommendations. Other similar studies [98] [97] emphasise the importance of recommendation explanation.

Most of the popular models that are very accurate rely on complex models from which it
is difficult to extract reasoning. Practitioners often have a choice between an understandable simple algorithm with low accuracy, or an accurate latent factor approach lacking explainability [118, 53]. Therefore, designing a model that is both accurate and explainable is a great challenge.

In Chapter 6, we proposed TCTFVSVD, a novel latent factor model that integrated additional contextual information into a single model. The model captures more aspects of the rating data than other collaborative filtering models and is therefore both more complex and more accurate. Modelling the different aspects separately in the framework enables us to determine the contributions each aspect makes to the recommendation. Specifically, this chapter makes the following contributions to the field:

- We present a technique to quantify the explainable preference aspects captured by TCTFVSVD that was proposed in Chapter 6.
- We propose a novel explicit preference model by discovering important explicit features from user reviews using semantic clustering.
- We integrate the explicit model with the latent factor model and match the feature preferences in both models to extract more reasons that explain why an item is being recommended.
- We show how the reasons for recommending items from the integrated model can be visualised in an intuitive and understandable manner.

This study presents the most comprehensive illustrations of reasons for recommendations. It is the first to provide illustrations that can help both customers and businesses. The explanations provided to users improve system related metrics such as trust, persuasiveness, efficiency, effectiveness, and transparency. The explanations for businesses can help improve marketing strategies and long-term planning. The architecture proposed in this chapter fits into the review-based subcategory of latent models explanations in collaborative filtering systems in Table 2.2. As far as we know, it is the first system that explains all the preference aspects alongside making the recommendations. It is also the first system that provides explanations from both customer and business perspectives.

### 8.2 Recommendation and explanation framework

The explanations of the recommendations are extracted from a latent factor model on the one hand and an analysis of explicit feedback given in the user reviews on the other.
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

The latent factor model has been described in Chapter 6; the analysis of the reviews is a new contribution. Another contribution is the consolidation of latent and explicit (review-based) features to establish whether the two parts of the model agree before adding review-based sentiments to the explanations why an item was recommended.

As we explained in Chapter 6, TCTFVSVD is a matrix factorisation method that captures aspects of preferences in ratings. The aspect $\text{FP}$ represents preferences over features, $\text{FVP}$ represents preferences over feature values, $\text{B}$ represents user and item bias, $\text{CP}$ represents conditional dependencies, $\text{SI}$ stands for social influence, and finally $\text{T}$ is an abbreviation for time (see Figures 1.1 and 6.2b). Modelling these implicit components found in rating data opens the possibility of quantifying the contribution of each aspect, which can be used to illustrate the reasons for a recommendation. TCTFVSVD incorporates matrices and vectors for each of these aspects, which we were able to identify in typical rating data and the social connections of users. As Figure 6.2b shows, the model starts by loading the time-stamped user ratings as well as the social network data into memory. On the highest level, the model includes two loops enclosed in one main loop. The first loop trains the model parameters responsible for the intrinsic preferences, while the second loop mainly trains the components that capture social influence.

8.2.1 Explicit model

The reviews allow the identification of features of recommended items. If a user mentions a topic frequently in his reviews of items, the respective property of the item is likely important to him. An item is often described in terms of a few predominant topics—e.g. good service in a restaurant, where service is a feature and good service is a feature value. By identifying predominant topics for users and items, we can illustrate to the user that a preference of his for a characteristic of an item has led to recommending this item to him.

The process of extracting features from reviews is illustrated in Figure 8.1. Initially, both user and item reviews are tokenised and filtered to remove expletives. To extract the features and their predominance, we use the tools for speech and sentiments from the Stanford CoreNLP [67].

Using Parts of Speech (POS) tags, which specify word classes, we extract the nouns for users and items separately. For each user and item, the tool yields a word vector of the features that have appeared in the reviews submitted by a particular user or for a particular item. In order to prevent the extraction of the same features multiple times, WordNet [71, 72] is employed to group the synonyms. For example, in the movies domain,
one feature might be the actors who star in the movie. Both actor and player can be used interchangeably and have to be grouped.

For each feature extracted from the user reviews, we calculate the average sentiment value of a user and for an item towards feature word. The sentiment analysis tool in Stanford CoreNLP generates the sentence-based sentiment of the word with 5 possible values, which we quantify by assigning -2 to very negative, -1 to negative, 0 to neutral, +1 to positive, and +2 to very positive. For every sentence in which a word has appeared, we extract the sentiment score and take the average. For each user or item, we also calculate the relative frequency of each word in the reviews of that user or item, with respect to the sum of the frequencies of all words. For example, if a specific word has been repeated 10 times, and all the other words together have been repeated 190 times, the relative frequency of that word is obtained as 0.05.

Multiplying the average sentiments by the frequencies yields an estimate of the attractiveness of each item feature or feature value. The reason for multiplying the relative frequencies by the average sentiments is that people usually tend to talk more about issues for which they have strong sentiments.

Extracting these attractiveness scores for users and items yields two matrices that form the equivalents of the user and item matrices in the latent model of TCTFV SVD, which
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

is based on Probabilistic Matrix Factorisation (PMF) [79]. As we explained in Chapter 2, in PMF, the user’s preferences for features are factorised into the user-feature matrix $P$ and the preferences for items into the item-feature matrix $Q$. Review-based matrix $P'$ shows the users’ sentiments towards each feature. Matrix $Q'$ on the other hand shows the sentiments expressed towards the features of the items they have had experience of.

$$\forall w \in \Psi_u : P'_{uw} = \frac{a_u(w)b_u(w)}{\sum_{w \in \Psi_u} a_u(w)} \quad (8.1)$$

Eq. 8.1 shows how an entry for user $u$ and feature $w$ in user-feature matrix $P'$ is calculated based on all words $w$ in the user-specific word collection $\Psi_u$. $a_u(w)$ denotes the total number of times the feature word $w$ has appeared in the user $u$’s reviews, and $b_u(w)$ denotes the average sentence-based sentiment score of the feature word $w$ in user $u$’s reviews.

$$\forall w \in \Psi_j : Q'_{jw} = \frac{a'_j(w)b'_j(w)}{\sum_{w \in \Psi_j} a'_j(w)} \quad (8.2)$$

Analogously, Eq. 8.2 uses the words in the item-specific word collection $\Psi_j$ whose occurrences $a'_j(w)$ and sentiment scores $b'_j(w)$ are used to calculate the entries $Q'_{jw}$ of the item-matrix $Q'$.

ConceptNet [58] is a semantic network which consists of semantic vectors as representations of the meaning of words. The k-means clustering module in ConceptNet was used to cluster the words according to semantic similarity.

$$\forall c \in C(\Psi) : P''_{uc} = \frac{\sum_{w \in \Psi_c} P'_{uw}}{|c|} \quad (8.3)$$

The matrix entries $P''_{uc}$ for each cluster are calculated using the entries in the matrix $P'_{uw}$ and normalising by the number of words in each cluster $|c|$ (Eq. 8.3).

$$\forall c \in C(\Psi) : Q''_{jc} = \frac{\sum_{w \in \Psi_c} Q'_{jw}}{|c|} \quad (8.4)$$

Eq. 8.4 creates the item-based word cluster matrix $Q''$ based on the clusters and matrix entries $Q'_{jw}$.

Given matrices $P'$ and $Q'$ contain the user- and item-specific sentiments of a feature, the matrices $P''$ and $Q''$ denote the average sentiments of users and items regarding each feature cluster (topic), and this completes the model that can now be used for reasoning with the recommendations.
8.2.2 Latent and explicit models combined

Figure 8.2 illustrates how the models introduced in sections 6.2.2 and 8.2.1 are combined. The latent model provides both ratings and explanations, the explicit model supplies additional explanations which include explicit features users appreciate in items. The recommendation module receives the time-stamped user ratings as well as social network data as input and trains the TCTFVSVD model on the training data. Given the equivalent matrices $P$ and $P''$ as well as $Q$ and $Q''$, it is possible to correlate both models and establish whether the recommendations are significantly similar (rectangle marked 'Integration' in Figure 8.2). If this is the case, review-based explanations from the explicit model can be added to the recommendations. If the correlation is not significant, the explanations rely on three types of reasoning that are based on the latent model, which take input from ratings and social connections data only:

- Item-specific explanations encompass the item-rating-based aspects that contribute to an estimated rating (which the recommendation is based on) their relative contribution, the temporal change of the item rating for the user’s community, as well as the dynamics of an item rating for the user (rating-based explanations). These explanations are mainly of interest to the users themselves.

- User-specific explanations are based on the preference model extracted for a particular user. This includes the relative influence of the user’s friends on her, the relative contribution of each of the preference aspects in the user’s preference model, the comparison of a user’s friends’ influence compared to the influence of people outside the user’s social sphere, as well as the temporal properties of each of the preference aspects and the temporal behaviour of the overall model, which shows how the preferences of the user are drifting over time. These explanations can be useful both for users and businesses.

- Community-specific explanations include the representation of the preferences of the community the user belongs to. We define the user community or social circle as the user’s friends as well as friends of friends. A community is connected to people outside the community by social relationships. We refer to the people connected to the community as community friends. These explanations help the user see how similar her preferences are to those of her social circle. Since these explanations provide insights at a community level, they might be more interesting for businesses than individual users.
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

ReEx Integration

Reommendation & Explanation

Natural Language Processing

Training Data

Calculation the ratings and correlate

Top Items

Yes

Sig. & Strong Correlation?

No

Review-based Explanation

Recommendation & Explanation Presentation Interface

Figure 8.2: The ReEx Framework
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

The scopes of item-specific, user-specific, and community-specific explanations are shown with an example in Figure 8.3. Item-specific explanations explain the model of a specific movie for a user (item’s estimated rating), user-specific explanations explain the model of the user (estimated ratings given by the user to all movies), and community-specific explanations explain the model of the user’s community (estimated ratings given by the user, his friends, and friends of friends to all movies).

<table>
<thead>
<tr>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie 1</td>
<td>4.23</td>
<td>3.74</td>
<td>3.6</td>
<td>5</td>
</tr>
<tr>
<td>Movie 2</td>
<td>3.97</td>
<td>3.48</td>
<td>3.33</td>
<td>4.8</td>
</tr>
<tr>
<td>Movie 3</td>
<td>2.9</td>
<td>2.42</td>
<td>2.27</td>
<td>3.73</td>
</tr>
<tr>
<td>Movie 4</td>
<td>3.83</td>
<td>3.35</td>
<td>3.2</td>
<td>4.66</td>
</tr>
</tbody>
</table>

(a) Estimated ratings

(b) Social network

Figure 8.3: An example of the constructed preference model for 5 sample users over 4 movies. Green, red, and violet rectangles show the scopes of item-specific, user-specific, and community-specific explanations, respectively

8.2.3 Correlations between latent and explicit features

As discussed, the use of review-based information in ReEx depends on:

1. the existence of correlations between the information held by the ratings and the reviews pertaining to each rating, and

2. the capability of the models to extract this information in an accurate way so that correlations between ratings and reviews – when present – are preserved.

In most cases in rating datasets with reviews, the sentiment of the user review associated with a rating can be expected to be commensurate with the rating value. When the user is happy about a product, she gives a high rating to that product and writes a positive feedback note. If she is dissatisfied with the product, she writes a negative review and gives a low rating.

We investigated the correlation between rating-based and review-based preferences using sentiment analysis on 500 reviews from the Yelp restaurants (Yelp hotels and Epinions were also analysed with similar results) dataset using IBM’s tool AlchemyAPI which generates sentiment values between 0 and 1, 0 for the weakest sentiment and 1 for the strongest. We divided the reviews into two classes, positive and negative, based on the sentiment.
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

scores. Similarly, we divided the user ratings into negative (values below 3) and positive (ratings above 3). The rating-based classification is regarded as the true classification.

<table>
<thead>
<tr>
<th>Actual Positive</th>
<th>Actual Negative</th>
<th>Class Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td>217</td>
<td>35</td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>33</td>
<td>215</td>
</tr>
<tr>
<td>Class Recall</td>
<td>86.80%</td>
<td>86.00%</td>
</tr>
</tbody>
</table>

Table 8.1: Confusion matrix for the explicit rating prediction using sentiment analysis with IBM’s AlchemyAPI tool

Table 8.1 shows that the class precision (the fraction of correctly predicted positive/negative values over the total of the predicted positive/negative values) and recall (the fraction of correctly predicted positive/negative values over the total of true positive/negative values) are greater than 86%, which means that in over 86% of the cases, the sentiment analysis of the reviews supports the rating of the item.

Figure 8.4: The distribution of the correlations between the four top rated restaurants using the latent model, and the explicit model for users in the Yelp restaurants dataset

Given the evidence of strong correlations, we have to investigate whether the ReEx framework preserves them. The Pearson correlation value between the predictions of the latent factor model and the review-based predicted ratings of the top four restaurants for each user show significant correlations in the case of 1452 users among a total of 2496 (58.17%). This result corroborates that for more than half of all users, the top four restaurants are ranked almost identically. This means that we can extract personalised explanations from reviews for most of the users.

The distribution of significant correlations between top rated restaurants using the latent model and the explicit model is shown in Figure 8.4. Half of the users have a correlation value of about 80%, and the minimum correlation is above 60%. Therefore, all the significant correlations are strong, suggesting that for all users, the latent factor model
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

results in strongly correlated outcomes with those of the explicit model.

Therefore, for these users, the top restaurants predicted by the latent factor model are complemented with the explanations from the explicit model, since the explicit model results in similar predictions as the latent factor model for the top restaurants.

8.3 Extracting explanations

The explanations we extract from the latent and explicit models using the ReEx framework are given in the form of illustrations of recommendations rather than verbal explanations. In addition to explaining the extraction of the information for the graphics, we use an illustrative example to demonstrate the explanatory value of the information.

Illustrative example

The example demonstrates the explanatory value of the information graphics created by the architecture for the top four recommendations made for user no. 775 from the restaurants category of the Yelp dataset. The model recommends four restaurants, no. 94 with a projected rating of 4.49, no. 738 with 4.23, no. 462 with an estimated rating of 4.26 and no. 235 with a rating of 4.22. Due to limitations in space, some details are shown for the top recommendation only.

8.3.1 Rating-based explanation model

Item-specific explanations express why an item receives a particular rating and why specific items are recommended, based on the ratings of other items. Once the TCTFV-SVD model has been built from user ratings and social network information for each user, we obtain the estimated ratings $\hat{R}_{uj}$ of items $j$ for users $u$ according to Eq. 6.4 and use this value to rank the items in order of preference predicted for this user, and select the top 4 items to recommend. We rewrite Eq. 6.4 by separating the contribution of the aspects to the estimated rating. The results in Eqs. 8.5 to 8.9

$$\hat{R}_{uj}(t_{uj}) = \mu + bu_{u}(t_{uj}) + bi_{j}(t_{uj}) + \xi_{uj}(t_{uj}) + \theta_{uj}(t_{uj}) + \kappa_{uj}(t_{uj})$$ (8.5)

The expected rating $\hat{R}$ for user $u$ and item $j$ at time $t_{uj}$ consists of the users’ general taste $\mu$ calculated as the average rating value for all items given by all users, the user bias $bu_{u}$, item bias $bi_{j}$, the intrinsic preferences of the user $u$ denoted by $\xi_{uj}$, the social
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

influence \( \kappa_{uj} \) and other factors affecting user \( u \)'s preferences for item \( j \) denoted by \( \kappa_{uj} \). Eqs. 8.6, 8.7 and 8.8 show how these aspects are calculated.

\[
\xi_{uj}(t_{uj}) = \sum_{f=1}^{D} (P_{uf}(t_{uj}) + |I_u|^{-\frac{1}{2}} \sum_{v \in I_u} y_{if}) (W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj})) \tag{8.6}
\]

Eq. 8.6 calculates the intrinsic preferences of a user, \( \xi_{uj} \), based on the dynamic (\( t_{uj} \) expresses time) latent user matrix \( P(\cdot) \in \mathbb{R}^{N \times D} \) (for \( N \) users and \( D \) features), the number of ratings given by user \( |I_u| \), implicit feedback \( y_{if} \) of all users on feature \( f \) of item \( i \), the dynamic linear preferences of user \( u \) for feature \( f \), represented by the dynamic gradient \( W_{uf} \) and intercept values \( Z_{uf} \), which adjust the values of the feature matrix \( Q_{jf} \) to personalise them to a specific user's tastes. If the intrinsic preferences value \( \xi_{uj}(t_{uj}) \) for item \( i_1 \) is larger than the item \( i_2 \), it means that regardless of other people’s judgement, the user prefers item \( i_1 \) over item \( i_2 \).

\[
\theta_{uj}(t_{uj}) = \sum_{f=1}^{D} |T_u|^{-\frac{1}{2}} \sum_{v \in T_u} \omega_{vf}(W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj})) \tag{8.7}
\]

In Eq. 8.7, the social influence \( \theta_{uj} \) is calculated based on the number of friends of user \( u \), \( |T_u| \), the social interactions matrix \( T \in \mathbb{R}^{N \times N} \) (where \( N \) is the number of users), and \( \omega_{vf} \) denoting the influence of friends \( v \), which moderates the user’s linear dynamic preferences over features in the item feature matrix \( Q \in \mathbb{R}^{M \times D} \) (\( M \) items and \( D \) features) with \( W_{uf} \) and \( Z_{uf} \) gradient and intercept values.

\[
\kappa_{uj}(t_{uj}) = \sum_{f' = 1}^{D} (W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj}))(W_{uf}(t_{uj})Q_{jf'} + Z_{uf}(t_{uj})) \tag{8.8}
\]

Eq. 8.8 defines \( \kappa_{uj} \) as a function of the matrices \( W(\cdot) \in \mathbb{R}^{N \times D} \) and \( Z(\cdot) \in \mathbb{R}^{N \times D} \) which model the linear function of dynamic feature value preferences in combination with the the item feature matrix \( Q \in \mathbb{R}^{M \times D} \). Matrix \( Y \in \mathbb{R}^{D \times D} \) is the interaction matrix that captures the dependencies between features.

Other factors \( \kappa_{uj} \) account for the conditional preferences. For instance, if the value of other factors for item \( i_1 \) is greater than its value for item \( i_2 \), this means that item \( i_1 \) has feature values that are more compatible with better suit its other feature values. For example, if item \( i_1 \) is a travel plan to Switzerland in a hotel in mountain in winter, it would get a higher preference value than item \( i_2 \) which includes travel to Switzerland in summer and staying in the same hotel, although summer and winter alone might be equally
preferable for the user. This is because of the dependencies between hotel location and season in these two travel packages.

\[ E_u = \sum_{v \in I_u} [E_R(\pi_{bu}, \pi_{bi}, \pi_{\xi_{uj}}, \pi_{\kappa_{uj}}) + E_T(\pi_{\theta_{uj}})] \]  

(8.9)

The model error \( E_u \) for user \( u \) in TCTFVSVD in Eq. 8.9 \[110\] consists of the model error for the social aspect \( E_T \) and the error for non-social aspects \( E_R \). The model parameters and hyper-parameters for user bias, item bias, intrinsic preferences, other factors, and social influence are respectively denoted by \( \pi_{bu}, \pi_{bi}, \pi_{\xi}, \pi_D, \) and \( \pi_S \).

### 8.3.1.1 Item-specific: Aspect contributions

In order to obtain the aspect contributions for user \( u \), we first train the model using all aspects in Eq. 8.5 and record the user model error \( E_u \). To obtain the contribution of each aspect, we first set the value of that aspect in Eq. 8.5 and the respective parameters and hyper-parameters in Eq. 8.9 to zero to exclude the aspect’s influence on the outcome and recalculate the error. Then we calculate the absolute difference in these two error values.

![Figure 8.5: Aspect contributions for this user, Yelp restaurants dataset](image-url)
Example

The pie charts in Figure 8.5 show that for the first and third top-ranked restaurants, intrinsic (personal) preferences of the user are the major contributor to the estimated ratings, while for the other two restaurants, social influence contributes over 50% of the estimated rating that accounts for these three aspects. Apparently the second and fourth restaurants have unidentified (and unidentifiable) features for which the user’s preferences are influenced by her friends, whereas the first and third restaurants are preferred mainly because they have features that are to the user’s personal liking. In none of these cases do other factors account for any observable influence.

8.3.1.2 Item-specific: Item popularity shifts

It may be interesting for users to see how attractive they found the recommended item over time, which, in combination with the contribution of intrinsic preferences from Figure 8.5, shows how likely the user would have been to have this item recommended to him at other points in time. Similarly, in combination with the social influence illustrated in Figure 8.5, it may be interesting to see how likely the user would have been to have this item recommended to him based on the popularity of this item with his friends. This information may also be interesting for the company recommending the item.

To obtain the popularity shift of item \( j \) for user \( u \), we simply plot the estimated rating value of the item for the user \( \hat{R}_{uj}(t) \) with respect to time using Eq. 8.5. To obtain the popularity shift of an item for the community, at each time point, we calculate the average estimated rating of that item by the community members using Eq. 8.5 and then plot the average estimated ratings with respect to time.

Example

Figure 8.6 shows a user’s preferences for a restaurant with respect to time. The x-axis labels denote ‘bins’ of time used in TCTFVSD. The model is built using data from the period from 2000 to 2017, and this period is divided into 30 bins with equal durations. Therefore, each bin represents roughly a quarter of a year, and the graph shows how the user preferences have been developing from the first rating entry until the present. The user’s preferences for the items have experienced only minor fluctuations towards the end of the period. Among the user’s community, all four top restaurants recommended have become slightly less attractive over time, although differences are very minor in both cases (smaller than a hundredth of a rating score).
Figure 8.6: Development of popularity of the four recommendations over time. On the left, the projected attractiveness of the items to the user who receives the recommendations, on the right, the development of their popularity among the user’s friends. The x axis spans the time covered by the Yelp restaurants dataset.
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

Since the estimated rating calculation function in Eq. 8.3 is a function of time, we can use this equation to predict the estimated rating of the restaurant at any point in the future. The accuracy of the prediction naturally declines with the distance from the time of the actual ratings.

Figure 8.7: The result of semantic clustering on the Yelp restaurants dataset

8.3.2 Review-specific: Word Clouds

The clusters obtained through the application of ConceptNet as described in Section 8.2.1 are shown in a visual interpretation of the features that are extracted by the explicit model from the discussions in the user reviews. Each word cloud illustrates one cluster $c \in C(\Psi)$ in Eqs. 8.3 and 8.4. For each word $w$ in cluster $c$, the word frequencies needed to determine the size of the word in the word cloud are obtained using Eq. 8.10:

$$q_w = \sum_{u=1}^{M} A_{uw} \times \sum_{j=1}^{N} B_{jw}$$

For each word, we also multiply the accumulated frequency of that word in all user reviews (matrix $A$) by the accumulated frequency of that word in all item reviews (matrix $B$). $\Psi$ denotes the set of all feature words extracted for all users and all items. Logically, the important features include feature words that are discussed more often in both the user
and item dimensions. Therefore, we expected that more frequently occurring words would include more relevant feature words. Calculating the percentiles for the distribution of the feature word frequencies for a representative subset of the Yelp dataset showed that the top 5% of the words had occurred at least 775 times, and the lowest 5% had a frequency of less than 2. By comparing the top 5% with the bottom 5% frequent words, as we expected, we observed that the top 5% included mostly relevant words, while the bottom 5% mostly included irrelevant words. Therefore, we decided to only feed the top 5% to the clustering module, in order to get more meaningful clusters.

Example

The topics extracted for this dataset form meaningful clusters, which represent the topics that seem most important for the users since they feature prominently in user reviews. The cluster in Figure 8.7a is mostly about drinks served in restaurants (drinks, beer, coffee, bar, wine) and the cluster in Figure 8.7b includes the words about food (food, pizza, chicken, pork, salad, burger, steak). The relative quality of the clusters extracted can be evaluated by the metric of average distance within clusters, which are -40.18, -58.37, -61.35, -37.72, -56.57, -42.75 for the clusters in Figures 8.7a to 8.7f respectively.

8.3.3 Review-specific: Topic contributions

Given the Word Cloud clusters in Figure 8.7, the pie charts in Figure 8.8 quantify the significance of each topical cluster for the user and item. The user graphic is based on values from the $P''$ matrix for each cluster (Eq. 8.3) and the values for each cluster in the item chart are extracted from the $Q''$ matrix (Eq. 8.4).

Example

In the left chart, the user can see which features are more important to him, whereas the right-hand graph shows which features users like about the top recommended restaurant. The features that are prominent in both graphs (time, people, place and food) are an indication of the reasons why the item was recommended to this user. Time appears the top priority of the user (left chart), and time is the third most important feature why this item is popular (right chart). The user prioritises service, but service is not a feature for which people generally choose this restaurant. Conversely, the restaurant is popular for its drinks, which are not very important for this user. It seems that the location, place and food features are sufficient reason for this restaurant to be the top recommendation for
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

Figure 8.8: The composition of the intrinsic feature preferences of the user (left) and the feature values of the top recommended item (item 94)

this user. This is corroborated by Figure S.5 where we learn that the major contributor to the ranking of the top restaurant is the user’s intrinsic preferences rather than social influence.

8.3.4 User-specific aspects

Unlike item-specific explanations which focus on explaining how a user rates a particular item, the user-specific explanations are based on the general aspects of user preferences over all items. This information could be of interest for both users and businesses. Understanding the characteristics of customers can help businesses in various ways, e.g. in choosing marketing strategies and product offerings.

8.3.4.1 Influential friends

In order to calculate the social influence of a particular friend $v$ on user $u$, denoted by $F_{uv}$, we calculate the user model error $E_{uv}$ without the friend’s influence and subtract that value from the user model error $E_u$ that we obtain when all friends are assumed to be influencing the user’s preferences. The difference shows how much the user’s preferences would be changed, if that friend’s influence was ignored. The influence of user $u$’s friend $v \in T_u$ denoted by $F_{uv}$ is calculated according to Eqs. S.11 – S.12

$$F_{uv} = |E_u - E_{uv}|$$

(8.11)
\[ E_{uv} = \sum_{\forall j \in I_u} \left[ E_R(\pi_{bu_i}, \pi_{bij}, \pi_{Xi}, \pi_{Kj}) + E_T(\pi_{\theta_{uj}}) | (\forall f, \forall v: \omega_{vf} = 0) \right] \] (8.12)

Eq. 8.12 defines the error contribution \( E_{uv} \) of friend \( v \) of user \( u \) in terms of the error contribution of the total model rating error \( E_R \) and the social circle \( E_T \) where \( \pi_x \) are hyper-parameters.

To calculate the influences by the social circle and people outside the social circle, we calculate the average influence of the community members denoted by \( F_{u,Comm_u} \) according to Eq. 8.13:

\[ F_{u,Comm_u} = \frac{|E_u - E_{u,Comm_u}|}{N_{Comm_u}} \] (8.13)

\[ E_{u,Comm_u} = \sum_{\forall j \in I_u} \left[ E_R(\pi_{bu_i}, \pi_{bij}, \pi_{Xi}, \pi_{Kj}) + E_T(\pi_{\theta_{uj}}) | (\forall f, \forall v \in \text{Comm}_u: \omega_{vf} = 0) \right] \] (8.14)

Similarly, we calculate the influence of the users who are not in the user’s community \( E_{u,Commu} \), in Eq. 8.14, then plot the average in the column chart.

![Figure 8.9: Influential friends (number of friends: 3)](image)

**Example**

User 775 for whom these recommendations are made has three friends who make contributions to his recommendations. The bar chart in Figure 8.9 shows that friend no. 506 is the most influential. User 405 has about two thirds of 506’s influence, user 1189 slightly
less. The user’s community includes friends of friends in addition to friends. The column labelled ‘Comm.’ is an average of these six users, and it includes the values of the first three columns in the average. The average influence of an outsider is so small as not to show on this graph, given the total number of outsiders is 2709.

Users may be interested in knowing who they are influenced by in their decisions, but this information is also interesting for businesses. When in doubt, a business can recommend items that user 506 has liked, on the basis that the user no. 775 tends to follow the decisions of this friend.

8.3.4.2 Aspect contributions

To calculate the contribution of each aspect to user $u$’s preferences, we set the hyper-parameters for that user and that aspect to zero, and calculate the difference in model error. For example, the contribution of intrinsic preferences of user $u$ denoted by $C_{\xi_u}$ is calculated according to Eqs. 8.15 and 8.16.

\[ C_{\xi_u} = |E_u - E_{\xi_u}| \]  
(8.15)

\[ E_{\xi_u} = \sum_{\forall j \in I_u} [E_R(\pi_{bu}, \pi_{bi}, \pi_{\xi_u}, \pi_{\xi_j}) | (\xi_{\xi_j} = 0) + E_T(\pi_{\theta_j})] \]  
(8.16)

Figure 8.10: Aspect contributions
Example

The user knows based on previous information that he relies on his own (intrinsic) preferences over the suggestions by his social circle (friends and friends of friends). The column chart in Figure 8.10 adds the information about the influence of the average taste of all users to the model (users’ general taste), as well as the bias items, and shows their relative significance. The recommendation is predominantly influenced by the overall tastes of the users in the system, but user and item bias also contribute to a significant extent. These two items express the fact that some users give above-average ratings regardless of item, and the fact that some items receive above-average ratings, for reasons other than user-item interactions (e.g. user mood).

Users may be interested to know these details, but businesses could also use it for marketing and customer relationship management. For example, a business can use this information to segment their customers into two groups of self-reliant, and socially-suggestible customers. Then the business can use different strategies to market to these two groups.

8.3.4.3 Preference similarity within and outside the community.

From a marketing perspective, it is interesting for the businesses to know if there is any difference between the distribution of model similarities within the user’s community compared to similarities with the preferences of outsiders. To calculate the preference similarity of user $u$’s preferences to the people within his community compared to outsiders, for each user $v$ in the user’s community $Comm_u$, we calculate the cosine similarity between the estimated ratings over all items given by user $v$ and the estimated ratings given by user $u$. Similarly, we calculate the cosine similarities between user $u$’s ratings and the ratings of each user $v'$ not in user $u$’s community, and then plot the obtained distributions using box plots.

Example

Figure 8.11 shows the similarities between the user (775) and her community is smaller on average than her similarity with non-community members, although we also observe that some community members have the same influence as non-community members. This may be interesting for the user herself, but businesses might conclude that it is not a promising approach to try to reach this person by influencing her social circle.
Chapter 8. An Integrated Architecture for Preference Model Representation and Explanation

8.3.5 Community-specific aspects

At the highest abstraction level, we provide explanations of the preferences of the whole community the user belongs to. The information of the community’s aspect contributions is arguably more interesting for businesses than individual users. For example, a business might use a different marketing strategy when a community relies on their intrinsic preferences than when it is more susceptible to social influence. When a community is found to be highly susceptible to the influence of its members, a corporation can identify the most influential users and spread marketing information through these individuals, ensuring that the product or service is tailored to the preferences of the community. The information about the temporal properties of aspects can help the businesses identify the reasons for changes in community preferences. For example, a marketing strategy based on viral marketing would probably be no longer effective for a community that is getting more dependent on their own preferences and less reliant on social connections.

8.3.5.1 Influential friends

In order to calculate the influence of a particular user on the community, we first calculate the model error value for that community with the assumption that the user has no influence on the community, then subtract the calculated value from the model error calculated when the influence of that user is also taken into consideration. In other words, we calculate the user’s influence in terms of the change that results in the overall model error of the community. The last two columns in Figure 8.12 represent the average influence of community members and outsiders.
Figure 8.12: The influential friends of this community, among a total of 20 friends, and the average influence of the community of this community, and the outsiders

Example

The community of user 775 which is described by the column chart in Figure 8.12 has 20 friends. Users 443, 1463, 506, and 778 among these have the greatest influence on this community, with the other users having small or negligible influences. An additional 29 friends of friends in addition to the 20 friends form the community of this community. The average influence of the community as well as the average influence of outsiders is too insignificant to be visible.

From the business perspective, this could help identify users that are influential on an entire community (the community of user 775), which helps reach not one but an entire group of users.

8.3.5.2 Aspect contributions

In order to calculate the aspect contributions, similar to Section 8.3.4 we switch off each aspect respectively, and then calculate the community model error and subtract the obtained values from the error of the community model that employs all the aspects. This value shows how much the model error would change if that aspect was switched off, and can be used as a measure of aspect contribution.

Example

Figure 8.13 reveals that general tastes account for the major part of the community’s preferences, with user bias, item bias, and intrinsic preferences being responsible for the remaining part. Social influence has almost a negligible effect suggesting that the community overall is not much influenced by its social circle. This has implications for businesses who want to design effective marketing strategies for targeting communities. Similar to the individual users, they could segment the communities to two groups of socially-suggestible
and self-reliant and market to them in different ways.

8.3.5.3 Model temporal drift

The temporal drift of the community model is obtained by calculating the average rating given by all community members to all items at different times.

Example

Figure 8.14 shows that the average rating given by community gradually decreases until time period 11, at which it sharply falls. This suggests that the community has become less positive about the items on offer in general. This could alert the business to a negative trend and invite it to investigate the causes of this general disenchantment of the community. Such research on the part of the business might, in turn, reveal reasons why members of the communities like or dislike items in general, helping a business change their offerings to different groups.
8.3.5.4 Aspect temporal drift

To visualise the temporal drift of each aspect for the community, for each community member, we plot the normalised average value of that aspect in Eqs. 8.5 to 8.8 over time for all items. Although this community has 6 users, for some of these users, the aspects do not change. The plots show only users for whom the aspect contributions change over time.

(a) Bias temporal drift  
(b) Intrinsic preferences temporal drift  
(c) Socially-influenced preferences temporal drift  
(d) Other factors temporal drift

Figure 8.15: Community-specific explanation diagrams for the user in Yelp restaurants dataset

Example

Figure 8.15 shows the temporal development of preference aspects of 4 of 6 community members who have dynamic preferences. The graphs reveal that user 405 has become less strict over time, less socially-suggestible and more self-reliant. Users 405 and 506 have become less biased over time, whereas users 762 and 933 have increased their bias, 933 most recently. Three of the four users have become more self-reliant, whereas user 506 has a decreased value of intrinsic preferences (For user 506, everything decreases, for user 762, everything increases. How is this possible? Have to explain.)
8.4 Summary of key points

1. Explaining the recommendations to the users has drawn great attention recently, since it can improve user trust particularly in domains with high investment risk.

2. Latent factor models as the most powerful approaches to model preferences in industry suffer from lack of explainability power. Other more recent models based on deep learning (e.g. deep collaborative filtering) have the same issue.

3. A major step towards achieving explainability in latent factor models is taken by separating the model parameters for each preference aspect, which enables us to switch off any aspect arbitrarily.

4. Such a model allows us to explain recommendations in different levels, and serve the benefits of both users and businesses.

5. The latency of the features in latent factor models is another obstacle against explainability. This obstacle can be removed by complementing a latent factor model with explicit models created by user reviews using natural language processing techniques.

8.5 Conclusion

This chapter concludes our study of designing a general architecture for preference modelling, explanation, and representation in recommender systems. In this chapter, we complement the recommendation module, TCTFVSVD, by adding the module for explanation based on an explicit model, the integration module, and the visualisation module and complete the design of the ReEx architecture. The ReEx architecture shows how highly accurate recommendations can be accompanied by comprehensive visual recommendation interpretations and intuitive representations of user preferences, in order to improve system usability and trust for users, and marketing and customer relationship management for businesses.

The explanation framework generates explanations and representations for item-specific, user-specific and community-specific aspects. Item-specific explanations provide rationales for the recommendations and the rating values for the top 4 items that are recommended based on the preference model. Such explanations help the users understand why an item is recommended, and improve the transparency of the system. Item-specific explanations include rating-based and review-based explanations. Rating-based explanations are extracted from the user ratings, while review-based explanations are extracted from the user
reviews. They are only available in cases where the latent and explicit models agree on the items to recommend.

User-specific explanations provide a high-level picture of a user’s preference model, including the influential friends, the contribution of different aspects to the user’s preference model and the temporal behavior of each of the preference aspects as well as the temporal dynamics of the preference model. This information tends to have more potential for businesses who like to profile their customers.

Finally, community-specific explanations explain the preference model of the user community as a whole. They help the user compare her preference model with that of the community she belongs to. They are also helpful for businesses who want to understand the preferences of groups and the demographic of their customers, as well as needs for changing product ranges and other long-term planning themes.
Chapter 9

Conclusions and Future Directions

In this thesis, we proposed a general architecture for extracting different preference aspects from the user ratings and social network data. This architecture has significance, since it targets three objectives of accuracy, efficiency, and explainability at the same time. It extracts user preferences from the user feedback data without the need for additional information, is able to tackle data sparsity problem using assumptions, and exploits the interactions between different aspects by integrating all of them. It also uses the user reviews to enhance the explainability. The following sections elaborate on the research contributions and answers to research questions.

9.1 Research contributions

The major goal of this thesis was to build an accurate and efficient model of user preferences using additional information provided by contextual factors in recommender systems, and providing intuitive explanations to the users regarding the modelled preferences and recommendations extracted from those preferences. Accuracy may come in the cost of sacrificing efficiency, or explainability. Therefore, the main challenge in this thesis has been how we should make a balance between these objectives and meet all of them at the same time. Several aspects for the preferences have been identified in the literature. These aspects include feature preferences, conditional feature value preferences, social influence, temporal shift, and bias (see Figure 1.1). The existing work mainly captures feature preferences and some of these aspects depending on the model. As Figure 1.1 shows, these aspects are intuitively justified and the experiments have also shown that modelling these aspects improves the recommendation quality. However, a system that efficiently models these aspects did not exist before, probably due to the complexities of efficiently capturing
all of them from user ratings and social network data. Besides, due to the lack of a system that modelled all the aspects together, their interactions with one another were also not considered. Some systems that modelled feature value preferences used additional data from user reviews, and did not model these preferences directly in the latent factor models. Since each aspect is shown to be responsible for part of the preference pattern in data, and due to the possible interactions between the aspects, capturing them in a model poses a great opportunity for practitioners to improve the recommendation quality. Capturing the aspects that explain user preferences also enables us to explain the reasons behind a recommendation. In order to capture these aspects, we formulated an optimisation problem that minimised the error using variables specific to each aspect. Each aspect was captured using components that could be arbitrarily switched off to stop training that aspect. The estimated ratings and social network connections were estimated by mathematically combining the aspects in a rating prediction function. In the following sections, we elaborate on the research contributions of this thesis.

9.1.1 Contribution 1: feature-awareness in recommendations

We addressed the problem of incorporating feature awareness into the recommender systems based on latent factor models. A feature aware latent factor model should be able to avoid training irrelevant features. In the state-of-the-art models, especially when a large number of features is trained, irrelevant features are also trained in the model, and this degrades the model accuracy. The degrading effect of using many features had been emphasised by some researchers in the literature. However, to the best of our knowledge, there had been no solutions to this problem only using user ratings. In Chapter 3, we proposed a method to counter the negative effect of irrelevant features by training them individually and incrementally. Through extensive experiments on some of the popular datasets, we showed that incorporating the feature-awareness into latent factor models actually improved the recommendation quality.

9.1.2 Contribution 2: conditional feature value preferences in recommendations

In earlier work [113, 114], we proposed a novel opponent model to extract the opponent preferences in automated negotiations. In this model, the users’ preferences were modelled with a linear function using multi-attribute theory. This function combined the user preferences over negotiation issues, and user preferences over issue values. In the area of
recommender systems, the latent factor models also model user preferences using a linear function. However, the differences between users in preferring feature values are ignored in all the state-of-the-art latent factor models. This potentially leads to drop in accuracy, since the users might have completely conflicting preferences over feature values. For example a user may prefer movies with higher levels of action while another user could prefer the opposite. Furthermore, in the practical domains, there usually are dependencies between item features and their values, and these dependencies cannot be modelled using the multi-attribute theory. To address these problems, in Chapter 4 we proposed a new latent factor model, CFVSVD, which incorporated the conditional preferences over feature values. To the best of our knowledge, this was the first attempt in modelling these preferences, without the use of additional feedback information. The existing work extracted these preferences using the sentiments underlying user reviews. We also provided two theorems in this chapter and proved the suitability of the proposed function to estimate the user ratings. The experiments showed that incorporating conditional preferences over feature values in CFVSVD significantly improved the accuracy of the modelled user preferences, and empirically highlighted the importance of capturing these aspects.

9.1.3 Contribution 3: social influence on the preference aspects

In Chapter 4 we proposed a novel latent factor model to extract the conditional preferences over feature values. However, studies have previously shown that due to the homophily and social influence, the user preferences are actually subject to change by social factors \([65, 87, 31, 43, 105]\). In this chapter, we extended the model proposed in Chapter 4 so that the influence of friends on the preferences of the user was also taken into consideration and proposed CTFVSVD. In CTFVSVD, all preference aspects (preferences over features, and preferences over feature values) were subject to social influence. We carried out extensive experiments to evaluate CTFVSVD in comparison with the state-of-the-art, and performed statistical analysis to ensure that the observed differences do not happen by chance. The results showed that CTFVSVD significantly improved the accuracy of the state-of-the-art models.

9.1.4 Contribution 4: temporal dynamics of preference aspects

Another aspect that affects the user preferences is time. Obviously people change over time, their preferences and interests evolve. Some of these changes happen temporarily, in response to transient environmental factors (e.g. change in user mood), while others are
more permanent and happen as a result of change in the user’s personality. The temporal
dynamics of preferences had already been investigated and modelled by some researchers
in the literature \cite{62, 50, 15}. However, to the best of our knowledge, there were currently
no attempts at modelling the temporal properties of different preference aspects and their
combinations, because there was no model that combined all aspects of preferences and
their social and temporal properties into a single model. We extended the model proposed
in chapter 5 and proposed TCTFVSVD. In TCTFVSVD the temporal properties of each
one of the preference aspects and the combinations were considered. In this model, each
one of preference aspects and their temporal properties could be arbitrarily switched on
and off. Therefore it enabled us to analyse the temporal effects for each individual aspect
as well. We found that incorporating the time aspect of the preferences made significant
accuracy improvements over the static model and state-of-the-art models. We also found
that preference aspects are subject to different levels of dynamicity in various datasets and
domains, which supported our component-based approach in modelling user preferences.

9.1.5 Contribution 5: interaction effects of preference aspects

Inspired by the insights into the preference aspects obtained in Chapter \textit{\cite{2}} we also analysed
the interaction effects of the preference aspects using a component-based approach. To do
so, we argued that the model hyper-parameters and learning rates should first be optimised
for different preference aspect combinations to ensure that each aspect is optimally trained.
Therefore, we first proposed ATCTFVSVD (Adaptive TCTFVSVD), a hybrid latent fac-
tor approach using an evolutionary algorithm and gradient descent. In ATCTFVSVD, the
parameters in the latent factor model were trained using gradient descent, because the so-
lution space with respect to these parameters was differentiable, and gradient decent could
provide fast and good solutions. However, the hyper-parameters and learning rates did
not form a differentiable space with the model error. Therefore the evolutionary algorithm
was used to optimise the hyper-parameters and learning rates. Integrating the aspects
introduced some interaction effects between these aspects. For example, modelling tempo-
ral properties of feature value preferences may only be helpful when these preferences are
dynamic.

We analysed the interaction effects of the preference aspects in different datasets using
interaction plots. The results revealed that interactions are highly domain-dependent. This
supported the component-based approach taken in modelling the user preferences, and
suggested the need for a comprehensive architecture, in which all the preference aspects
and the factors affecting those aspects are captured using different components. To the best of our knowledge, this approach was the first to combine all preference aspects into a unified model, and the first to shed light on the interaction effects of the preference aspects. The component-based design of the model enabled us to switch off aspects arbitrarily, and train different models with different combinations of aspects. Therefore, we were able to analyse the helpfulness of individual aspects and the interactions between them. As we showed through the experiments, taking the interactions into consideration is important for achieving the best accuracy for model combinations in different datasets and domains.

**9.1.6 Contribution 6: providing intuitive visual explanations for the users and businesses**

Finally, we proposed a comprehensive bilateral architecture, ReEx, for providing intuitive recommendation explanations and preference model representations to the user and business sides. The explanations for the users are believed to help improve transparency and instil trust in users by giving the reasons behind the recommendations and how the aspects are combined in the model. The user reviews usually include rich information about how the users assess the items, and the experiments also showed strong correlations between the explicit ratings and the ratings extracted from the sentiments that underlie user reviews. Therefore, the reviews were potentially a good source to extract the explanations from. We used natural language processing techniques and proposed an explicit model that was integrated into TCTFVSVD in ReEx for generating explanations accompanying the recommendations. ReEx isolated the contributions of different aspects from ratings and visualised them to the users, through item-specific, user-specific, and community-specific explanations. These explanations are of interest for both the user and business sides. Providing these explanations are made possible, due to TCTFVSVD capturing all aspects in an explainable way. This contribution is important, since to the best of our knowledge, there is no architecture that is able to recommend and clearly explain all the aspects of preferences that contribute to a model or recommendation for both users and businesses.

**9.2 Answers to research questions**

Overall, the contributions of this thesis answer the research questions that were posed in Section 1.1. In the following sections, we provide detailed answers to each one of these research questions. As mentioned in Chapter 1 in this thesis we are interested in the following general research questions:
How can we use ratings, reviews, and contextual information to build an accurate, efficient, and easily explainable model of user preferences over items, given the data sparsity problem? This question was answered by addressing the following sub-questions.

- **How can we accurately model the user preferences with different aspects, given the data sparsity problem and interactions between aspects?**

  In Chapter 6, we completed the design of TCTFVSVD that incorporated all six preference aspects. In TCTFVSVD differences between users in preferring item feature values as well as the dependencies between item features are taken into consideration. Additional matrices were fused into the latent factor models to capture these aspects, and accuracy was improved. Due to the sparsity of the ratings, not enough data was provided for the models to be able to discern these patterns. Therefore, the existing work used additional feedback data from user reviews to model feature value preferences. To tackle the sparsity problem of user ratings, we assumed that the user preferences over feature values can be modelled using a linear function. Therefore this assumption enabled us to learn the feature values and respective preferences in a latent manner from user ratings, and also overcome the problem of data sparsity. TCTFVSVD also incorporates the influence of friends in social networks on the user preferences. Unlike the existing work, in TCTFVSVD all aspects of preferences, i.e. feature preferences and feature value preferences were assumed to be subject to social influence. The component-based formulation of the optimisation problem and the definition of influence matrix enabled us to quantify the social influence for each one the preference aspects and individual users. Through TCTFVSVD, we also addressed the problem of modelling the temporal properties of all preference aspects to improve the recommendation accuracy. The existing work only modelled the temporal drift of feature preferences. However, we speculated that other aspects are also subject to temporal drift, and ignoring their temporal properties would potentially degrade the accuracy. In order to capture different preference aspects, TCTFVSVD incorporates additional matrices and vectors, and made assumptions regarding the temporal pattern of the preference aspects. The hybrid framework in Chapter 7 built on TCTFVSVD was complemented with another optimiser to train the model hyper-parameters and learning rates. TCTFVSVD is a component-based model in which aspects could be arbitrarily switched on/off. Using the optimiser for hyper-parameters and learning rates, the model was able to train the sub-optimal values for each aspect when different aspect combinations were present. Therefore, the model
enabled us to investigate the interaction effects of the preference aspects in different datasets. The experiments revealed that due to the aspect interactions, the helpfulness of the individual aspects depend on datasets, suggesting that it is important to understand these interaction effects in order to design an accurate model in every preference domain. For example, if an aspect was found to be unhelpful in a specific domain or dataset, it could be switched off to improve the accuracy. Therefore, through TCTFVSVD and ATCTFSVD, we were able to capture all 6 preference aspects, improve the accuracy with every aspect captured, exploit the interactions without using additional data and overcome the data sparsity problem, and this question was answered.

• **How can we extract the preference patterns from ratings and social network data in a way that the model is efficient and scalable?**

As we mentioned before, TCTFVSVD is a latent factor based model. The additional matrices and vectors are added to the model to capture the preference aspects, complicates the model and increases the computational complexity. However, as we analysed in Section 6.2.5, the complexity still remains linear with the number of ratings and the social network connections. The formulation of the error and the rating estimation function using the assumptions about the way the aspects combined made the solution space differentiable. Consequently, gradient descent algorithm which can find good solutions in a reasonable amount of time was made applicable. Moreover, the component-based design of the algorithm enabled us to switch off an aspect when it was found to be unhelpful, hence further reduce computational time and improve the accuracy. TCTFVSVD strikes a balance between accuracy and efficiency using the assumptions as to how the ratings are constituted. Therefore, the component-based design of TCTFVSVD using matrix factorisation enabled us to answer this question as well.

• **How can we model the preference patterns from rating and social network data and user reviews, in a way that the resulting model and recommendations can be easily explained and represented to the users?**

Throughout this thesis, we showed how the three objectives of efficiency, accuracy, and explainability can be pursued in designing a preference model for recommendation systems. First, we adopted a solution based on latent factor models, due to their efficiency and explainability advantage. In this model, every preference aspect was
defined using matrices and vectors that were responsible for capturing the pattern related to that aspect. Then the mathematical equations combined these variables together to form a rating prediction. Mathematical analysis was given to assure that the way the aspects were defined were justified. Therefore, the model design was perfectly in line with the explainability objective, since every variable was interpretable, hence could be easily explained. To quantify the contributions of the preference aspects, we decomposed the rating prediction function as well as the model error into the constituting aspects. Then we measured the difference in error when each one of these aspects were switched off as opposed to when all the aspects were present. Intuitively, the most influential aspects are the ones that make the greatest difference in the model error when switched off. When the model error changes a lot by switching off an aspect, it means that that aspect accounts for a large portion of the existing pattern in data. Therefore, by calculating these values for each one of the users as well as for the whole user community, we were able to measure the contribution of the preference aspects. The equations for explaining the aspects by isolating them from the constructed model were presented in Chapter 8. We also used natural language processing techniques, and complemented the recommendations extracted from the user reviews and social network data, with explanations that were extracted from the user reviews to enhance the explanations and proposed ReEx framework. Therefore, through ReEX we showed how we also met explainability, along with accuracy and efficiency objectives and answered this question too. Therefore, this model provides a fine balance between accuracy, efficiency, and understandability.

9.3 Future research avenues

The research carried out in this thesis can be extended and improved in a number of ways. Below, we elaborate on some of the interesting future directions.

**Incorporating feature-awareness into TCTFVSVD**: In Chapter 3 we addressed the problem of incorporating feature-awareness into the basic latent factor models, and showed that feature-awareness results in improvements in recommendation quality. An interesting future direction involves incorporating the feature-awareness into TCTFVSVD in order to achieve further improvements in recommendation quality.

**Incorporating aspect-awareness into TCTFVSVD**: In Chapter 7 we concluded that preference aspects interact differently depending on the dataset and domain. Interesting future work involves aspect-aware training of the model, and allowing the model to
learn an aspect only when its presence is helpful for recommendation quality.

**Non-linear modelling of conditional preferences:** In Chapter 4, we looked at how we can model the conditional feature value preferences. We did this by using a simple linear function, which captured the linear dependencies between the features and their respective values. Obviously, some of the dependencies cannot be captured by this linear function, and some information is lost when the dependencies are non-linear. An interesting avenue for extension in the future could be using more complicated non-linear functions to model the conditional preferences in order to obtain further accuracy improvements. Furthermore, we have not explored the dependencies in different preference domains. Some experimental studies along with natural language processing could be carried out to obtain domain-specific knowledge into the dependencies between the features in each domain.

**Non-linear modelling of feature value preferences:** In Chapter 4, we showed how we can capture the differences between users in preferring item feature values. To strike a balance between the computational expense and the accuracy improvements, we used a simple linear function to model the user-specific feature value preferences. Then we showed that this function can capture some of the differences, but not all of them. A future extension to the model presented in this chapter involves exploring the possibility of devising non-linear functions to improve the accuracy, while keeping the computational complexity tractable.

**Improving recommendation quality by refining the social network graph:** In Chapter 5, we used the social network graph as the input, and modelled the social influence of friends on the preference aspects. However, we did not perform any pre-processing on the social network graph. We speculate that additional improvements are achievable, by further analysing the social connections between users, and using the additional contextual information to establish more accurate trust values. For example, the users’ chat history could be possibly processed and used to obtain more accurate estimations of their trusts.

**Extending TCTFVSVD for top-N recommendation problem:** Throughout this thesis, we addressed the problem of rating prediction, in which the rating value given by a particular user to an unseen item is predicted. The main problem with the rating prediction systems is that the users need to express their explicit ratings over a number of items. However, in the real world scenarios, the users usually provide implicit feedback (e.g. browsing or clicking behaviour), which provides an ordered list of items according to the users’ preferences. Furthermore, the exact preference values of the items may not much important in practice (probably except for explainability). The system would perform
satisfactorily, as long as it can successfully identify the top preferred items among a list of potential items to suggest to the user. Extending the proposed model in this thesis for the problem of top-N item recommendation is an interesting research direction for future.

**Interpreting latent features:** TCTFVSVD exploits a latent approach to learn the user preferences and the constituting preference aspects. Although using the latent features enables the model to achieve the highest accuracy, the meaning of the extracted features are not known. Although this is considered a benefit for the purpose of accuracy, it creates obstacles for explainability, since the features’ meanings are important in the model interpretability. Intuitively, they represent the criteria that affect the user’s decision when buying a particular product or service. An intriguing avenue for future exploration involves investigating ways of interpreting the latent features using natural language processing techniques. Interpreting the latent features would enable us to provide more intuitive explanations to the users, and hence would improve the system confidence and usability.

**Using deep learning to boost the recommendation accuracy:** As we reviewed in Chapter 2, recently there has been a line of research on unifying deep learning approaches with collaborative filtering using the deep collaborating framework [116]. Throughout this thesis, we only used the users’ time-stamped ratings as well as social network data as inputs to the model. The accuracy of the model can be improved by using additional information such as user reviews, item descriptions, users demographics, etc. In the future we intend to extend TCTFVSVD using the deep collaborative filtering framework. The basic idea is using the power of deep learning in extracting patterns from raw textual data, by extracting information about different aspects from textual data, and diffusing them into TCTFVSVD using matrix factorisation. For example, a deep network can take the user’s public posts or tweets on social media, his demographics data, etc. as input and extract hidden representations of social influence. Such a representation then can be diffused into the matrix factorisation model to further improve the model accuracy. Another deep network can estimate the parameters of users’ dynamicity defined in the model by processing different sources of data about the user.
Bibliography


[106] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys ’13, pages 347–350, New York, NY, USA, 2013. ACM.


[19] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, and Shi Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In


