EXTENDING LOW-RANK MATRIX FACTORIZATIONS FOR EMERGING APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

by

Ke Zhou

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology
December 2013

Copyright © 2013 by Ke Zhou

EXTENDING LOW-RANK MATRIX FACTORIZATIONS FOR EMERGING APPLICATIONS

Approved by:

Dr. Hongyuan Zha, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Le Song
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Polo Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Xiaoming Huo
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology

Date Approved: Aug 2013

TABLE OF CONTENTS

LIST OF TABLES . . . . **vii**

LIST OF FIGURES . . . . **viii**

ACKNOWLEDGEMENTS . . . . **xi**

SUMMARY . . . . **xii**

I INTRODUCTION **1**

1.1 Thesis 1

1.2 Cold-start Recommendation 2

1.3 Binary Coding for Collaborative Filtering 4

1.4 Learning Social Infectivity from Temporal Events 6

1.5 Learning Nonparametric Kernels for Hawkes Process 8

1.6 Outline 10

II BACKGROUND AND RELATED WORK **11**

2.1 Low-Rank Matrix Factorization 11

2.1.1 Singular Value Decomposition 11

2.2 Collaborative Filtering 14

2.2.1 Low-Rank Matrix Factorizations with Incomplete Observations for Collaborative Filtering 15

2.2.2 Nuclear Norm Regularization: A Convex Formulation 17

2.2.3 Cold-Start Collaborative Filtering 17

2.3 Learning Binary Codes 19

2.4 Self-Exciting Point Process 21

2.5 Estimating the Triggering Kernels 22

III FUNCTIONAL MATRIX FACTORIZATIONS FOR COLD-START RECOMMENDATION **24**

3.1 Functional Matrix Factorization 24

3.1.1 Functional Matrix Factorization 25

3.1.2 Alternative Optimization 27

3.1.3 Decision Tree Construction 28

3.1.4 Hierarchical Regularization 30

3.2 Extensions 32

3.2.1 Multiple Questions 32

3.2.2 Optimization 34

3.2.3 POMDP formulation for cold start problems 35

3.3 Experiments 39

3.3.1 Data sets and Evaluation Metrics 39

3.3.2 Deriving Interview Responses from User Ratings 41

3.3.3 Experiment Design 42

3.3.4 Cold-Start Performance 42

3.3.5 The Impact of Non-responses 46

3.3.6 Warm-Start Performance 49

3.3.7 Impact of Model Parameters 53

3.4 Summary 57

IV LEARNING BINARY CODES FOR COLLABORATIVE FILTERING 58

4.1 Learning Binary Codes 59

4.1.1 Problem Formulation 60

4.1.2 Learning 63

4.1.3 Obtaining Binary Codes 65

4.2 Experiments 69

4.2.1 Evaluation Metrics 70

4.2.2 Data Sets 71

4.2.3 Comparison of Rounding Methods 71

4.2.4 General Performance Results 72

4.2.5 Recommendation Efficiency 76

4.3 Summary 77

V LEARNING SOCIAL INFECTIVITY IN SPARSE LOW-RANK NETWORKS USING MULTI-DIMENSIONAL HAWKES PROCESSES 83

5.1 Multi-dimensional Hawkes Processes with Low-rank and Sparse Structures 84

5.1.1 One-dimensional Hawkes Processes 84

5.1.2 Multi-dimensional Hawkes Processes 87

5.1.3 Sparse and Low-Rank Regularization 89

5.2 Efficient Optimization 89

5.2.1 Solving for $\mathbf{Z}_1$ and $\mathbf{Z}_2$. 92

5.2.2 Solving for $\mathbf{A}$ and $\boldsymbol{\mu}$ 93

5.3 Experiments 96

5.3.1 Synthetic Data 96

5.3.2 Real-world Data 100

5.4 Summary 101

VI LEARNING TRIGGERING KERNELS FOR MULTI-DIMENSIONAL HAWKES PROCESSES 107

6.1 Nonparametric Triggering Kernel Estimation using Euler-Lagrange Equations 108

6.1.1 Optimization Problem and Space 108

6.1.2 Iterative Algorithm 110

6.2 Extension to Spatial-Temporal Process 114

6.3 Experiments 118

6.3.1 Toy Data 118

6.3.2 Synthetic Data 121

6.3.3 Real World Data 126

6.4 Summary 128

VII CONCLUSIONS AND DISCUSSIONS 130

7.1 Summary 130

7.2 Discussions and Future Directions 131

REFERENCES **133**

VITA **147**

LIST OF TABLES

1 Statistics for Data Sets 41

2 Examples of interview process with 6 questions 49

3 Examples of interview process with 6 questions 50

4 RMSE on MovieLens data set in warm-start setting 54

5 RMSE on EachMovie data set in warm-start setting 54

6 RMSE on Netflix data set in warm-start setting 54

LIST OF FIGURES

1 An example for decision trees: Top three levels of a model of depth 6 for MovieLens data set 3

2 The idea of learning binary codes for collaborative filtering 5

3 An illustration of recurrent mutually-exciting process. 7

4 Evaluation process for cold-start users 44

5 RMSE on MovieLens data set for cold-start users with respect to the number of interview questions 46

6 RMSE on MovieLens data set for cold-start users with respect to the number of interview questions 47

7 RMSE on MovieLens data set for cold-start users with respect to the number of interview questions 48

8 Performance measured by RMSE on Netflix data set with respect to the fraction of users with the most known ratings. 51

9 Performance measured by RMSE on three data sets with respect to the fraction of users with the most known ratings. 51

10 RMSE of fMF, Tree and MF on MovieLens data set with respect to the number of interview questions 53

11 Performance measured by RMSE with respect to different values of K on MovieLens data set. The performance is reported by setting the depth of the decision tree to be 6. 55

12 The performance of hierarchical regularization. The performance is reported by setting the depth of the decision tree to be 6. 55

13 Performance measured by RMSE with respect to different values of λ on MovieLens data set. 56

14 Performance measured by RMSE with respect to the number of iterations on MovieLens data set. 56

15 Performance of two rounding methods on MovieLens, EachMovie and Netflix data sets 72

16 Performance with respect to the length of binary codes on Movielens data set 78

17 Performance with respect to the length of binary codes on EachMoive data set 79

18 Performance with respect to the length of binary codes on Netflix data set . . . 80

19 Variance of predicted similarity on unknown ratings with respect to the length of binary codes . . . 81

20 Performance measured by DCG with respect to the regularization parameter λ on MovieLens data set . . . 81

21 Performance measured by DCG with respect to the number of iterations on MovieLens data set . . . 82

22 Comparison of low-rank matrix factorization and CFCodeReg on MovieLens data set . . . 82

23 Assortative mixing networks: performance measured by RelErr, PredLik and RankCorr with respect to the number of training samples. . . . 102

24 Disassortative mixing networks: Performance measured by RelErr, PredLik and RankCorr with respect to the number of training samples. . . 103

25 Performance measured by PredLik with respect to the value of λ_1. . . 104

26 Performance measured by PredLik with respect to the value of λ_2. . . 104

27 Performance measured by PredLik with respect to the number of inner/outer iterations. . . . 105

28 Performance measured by PredLik on MemeTracker data set. . . . 105

29 Performance measured by RankCorr on MemeTracker data set. . . . 105

30 Influence structure estimated from the MemeTracker data set. . . . 106

31 Estimated vs. True Triggering Kernels on toy data DataExp with exponential kernels. The four sub-figures show both the true triggering kernels and the estimated ones from the data . . . 119

32 Estimated vs. True Triggering Kernels on toy data DataCos. The four sub-figures show both the true triggering kernels and the estimated ones from the data . . . 120

33 Performance measured by LogLik and Diff with respect to the number of training samples on the synthetic data. . . . 122

34 Performance measured by LogLik with respect to number of base kernels D. . . . 123

35 Performance measured by LogLik with respect to point used to discretize the ODE. . . . 123

36 Performance measured by LogLik with respect to number of iterations. 124

37 Performance measured by LogLik with respect to regularization parameter α. 124

38 Performance measured by negative log-likelihood on MemeTracker data set. 128

39 Q-Q plot for comparing the transformed events with respect to the exponential distribution. 129

40 The base kernel estimated from the MemeTracker data set. 129

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Hongyuan Zha, for his invaluable help and support during the process of my research. I would also like to thank my committee members, Professor Le Song, Professor Polo Chau, Professor Xiaoming Huo and Professor Jacob Eisenstein, for all the discussions and suggestions that have greatly improved this dissertation.

I would like to thank my former advisers, Professor Yong Yu and Professor Gui-Rong Xue in Shanghai Jiao-Tong University, for helping me make my first steps into research. I would like to take this opportunity to thank my teachers of computer programming class in middle schools, Mr. Wu and Mr. Sun, for teaching me to write my first computer programs and opening the window to the world of computer science.

I would also like to thanks all my friends for their their support and help during my study. It is my fortunate to know all of you.

Last but not least, I would like to thank my parents, for their love and great support throughout my life.

SUMMARY

Low-rank matrix factorizations have become increasingly popular to project high dimensional data into latent spaces with small dimensions in order to obtain better understandings of the data and thus more accurate predictions. In particular, they have been widely applied to important applications such as collaborative filtering and social network analysis. In this thesis, I investigate the applications and extensions of the ideas of the low-rank matrix factorization to solve several practically important problems arise from collaborative filtering and social network analysis.

A key challenge in recommendation system research is how to effectively profile new users, a problem generally known as *cold-start* recommendation. In the first part of this work, we extend the low-rank matrix factorization by allowing the latent factors to have more complex structures — decision trees to solve the problem of cold-start recommendations. In particular, we present *functional matrix factorization* (fMF), a novel cold-start recommendation method that solves the problem of adaptive interview construction based on low-rank matrix factorizations.

The second part of this work considers the efficiency problem of making recommendations in the context of large user and item spaces. Specifically, we address the problem through learning binary codes for collaborative filtering, which can be viewed as restricting the latent factors in low-rank matrix factorizations to be binary vectors that represent the binary codes for both users and items.

In the third part of this work, we investigate the applications of low-rank matrix factorizations in the context of social network analysis. Specifically, we propose a convex optimization approach to discover the hidden network of social influence with low-rank and sparse structure by modeling the recurrent events at different individuals as multi-dimensional Hawkes processes, emphasizing the mutual-excitation nature of the dynamics of event occurrences. The proposed framework combines the estimation of mutually exciting process and the low-rank matrix factorization in a principled manner.

In the fourth part of this work, we estimate the triggering kernels for the Hawkes process. In particular, we focus on estimating the triggering kernels from an infinite dimensional functional space with the Euler Lagrange equation, which can be viewed as applying the idea of low-rank factorizations in the functional space.

CHAPTER I

INTRODUCTION

Low-rank matrix factorizations have become increasingly popular to project high dimensional data into latent spaces with small dimensions in order to obtain better understandings of the data and thus more accurate predictions. In particular, they have been widely applied to important applications such as collaborative filtering and social network analysis. However, most current studies of low-rank matrix factorizations focus on latent factors that can be represented by vectors and relative simple objective functions such as the square loss, which considerably limits their applications. On the other hand, the problems in real-world are much more complex and diverse in nature and thus often involve more structures and constraints on not only the latent factors themselves but also the objective functions. In particular, we consider four different problems to illustrate the diversity and complexity arise from real-world problems: cold-start recommendation, binary coding for collaborative filtering, learning social infectivity from temporal events and learning nonparametric kernels for Hawkes process.

1.1 Thesis

In this thesis, we propose to investigate the applications and extensions of the ideas based on the low-rank matrix factorization to solve the problem mentioned above. The basic idea is to adapt and extend the matrix factorizations to incorporate the requirements from the application domains. In particular, we propose to introduce structured latent factors into traditional low-rank matrix factorizations: The structures can be restricted by decision trees to solve the cold-start recommendation problem and discretized binary structures can be adapted to solve the binary coding problem for collaborative filtering. Moreover, we also combine the ideas of low-rank matrix factorization with the self-exciting processes to model the temporal events in social networks. We also consider latent factors that comes from infinite dimensional functional space to estimate the triggering kernels for Hawkes process. In the following sections, we will briefly describe the backgrounds of the three problems.

1.2 Cold-start Recommendation

Recommendation systems have become a core component in today's online business world. A key challenge for building an effective recommender system is the well-known cold-start problem — *how to provide recommendations to new users?* While existing *collaborative filtering* (CF) approaches to recommendation perform quite satisfactorily for warm-start users (e.g. users purchasing a lot from an online retailer), it could fail completely for fresh users, simply because the system knows very little about those users in terms of their preferences.

Providing effective cold-start recommendations is of fundamental importance to a recommender system since it directly impacts to the attractiveness of the system to new users and thus vital for reputation and the market share of the system. A natural approach to solving the cold-start problem is to elicit new user preferences by query users' responses progressively through an initial interview process [123]. Specifically,

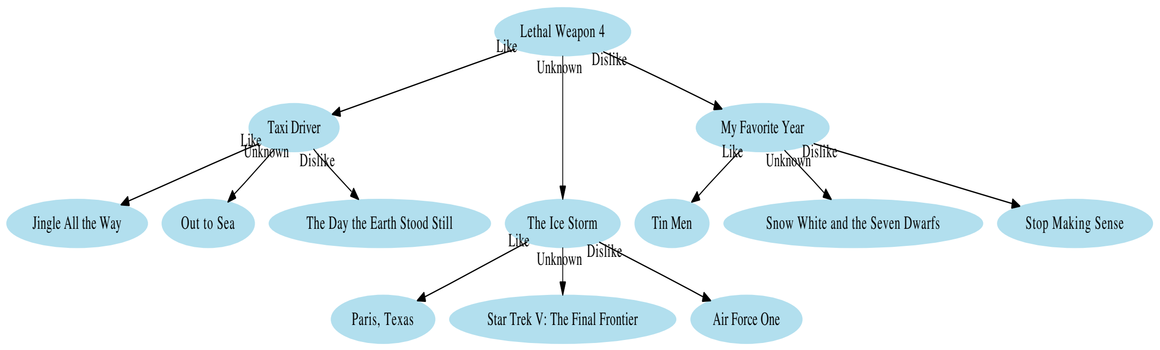


Figure 1: An example for decision trees: Top three levels of a model of depth 6 for MovieLens data set

at the visit of a new user, the recommender initiates several trials. At each trial, the recommender provides a seed item as a question and ask the user for her opinion; based on the user's responses to the questions, the recommender gradually refines its characterization (i.e. profile) of the user so that it can provide more satisfactory recommendations to the user in the future. A good initial interview process should not be time-consuming: the recommender can only ask a very limited number of questions or otherwise the user will become impatient and leave the system. Equally important, the process should also be effective, i.e. the answers collected from the user should be useful for constructing at least a rough profile for the user. As an illustration of movie recommendation, in an interview process depicted in Figure 1, the system asks a new user the question "Do you like the movie Lethal Weapon 4?". The user is allowed to answer with either "Like", "Dislike" or "Unknown". The recommender refines its impression about the user and then direct the user to one of the child nodes and then presents the next interview question according to her previous response. For example, if a user chooses "Like" for "Lethal Weapon 4?" at the root node of Figure 1, she will be directed to the left child node and will be asked the question "Do you like the movie Taxi Driver?"

In this work, we propose *functional matrix factorization* (fMF), a novel method for the construction of such interview decision trees. We argue that it is more effective to combine the matrix factorization model for collaborative filtering and the decision-tree based interview model into a single framework. Our proposed method is based on the low rank factorization of the incomplete user-item rating matrix with an important twist: it restricts the user profiles to be a *function* of the answers to the interview questions in the form of a decision tree, thus the name functional matrix factorization. The function — playing the role of the initial interview — is in the form of a decision tree with each node being an interview question [50, 122].

1.3 Binary Coding for Collaborative Filtering

With the rapid growth of E-commerce, hundreds of thousands of products, ranging from books, mp3s to automobiles, are sold through online marketplaces nowadays. In addition, millions of customers with diverse backgrounds and preferences make purchases online, generating great opportunities as well as challenges for E-commerce companies — How to match products to their potential buyers not only accurately but also efficiently. Due to the nature of their applications, collaborative filtering systems are usually required to learn and predict the preferences between a large number of users and items. Therefore, for a given user, it is important to retrieve products that satisfy her preferences efficiently, leading to fast response time and better user experience.

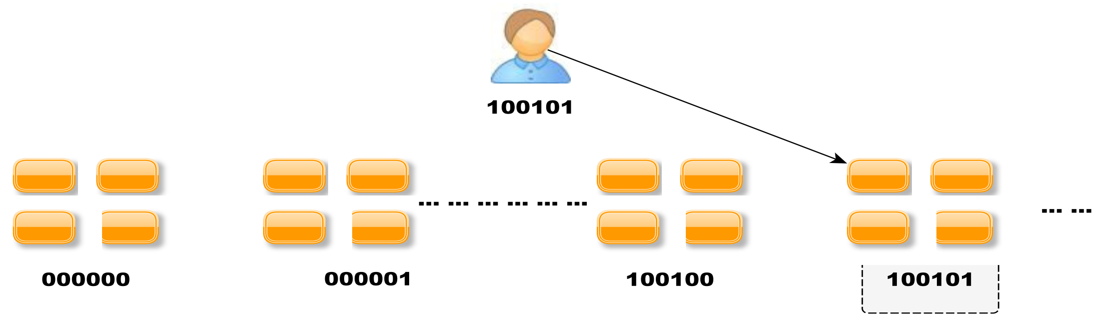


Figure 2: The idea of learning binary codes for collaborative filtering

In this part of the thesis, we address the problem of learning binary codes for collaborative filtering. Specifically, we propose to learn compact yet effective binary codes for both users and items from the training rating data. A great advantage is that given the binary code representations for users and items, the recommendation can be performed quickly through retrieving the items with similar binary codes as the user. (See Figure 2)

Unlike previous works on learning binary codes, we do not assume the similarity between users and items are known explicitly. Therefore, the binary codes we construct not only accurately preserve the observed preferences of users, but they also can be used to *predict* the unobserved preferences, making the proposed method conceptually unique compared with the existing methods for learning binary codes [65,80,161]. Our approach is based on the idea that the binary codes assigned to users and items should preserve the preferences of users over items. Two loss functions are applied to measure the divergence between the training data and the estimates based on the binary codes. Unfortunately, thus formulated, the resulting discrete optimization problem is difficult to solve in general. Through relaxing the binary constraints, it turns out the relaxed optimization problem can be solved effectively by existing solvers. The problem can be viewed as a matrix factorization problem with restricted binary latent factors. Moreover, we propose two effective methods for rounding the relaxed solutions to obtain binary codes.

1.4 Learning Social Infectivity from Temporal Events

In today's explosively growing social networks such as Facebook, millions of people interact with each other in a real-time fashion. The decisions made by individuals can be largely influenced by their neighbors, the authorities and various communities, which is known as the property of following the crowd in social behaviors [19, 101]. For example, a recommendation from a close friend can be very decisive for the purchasing of a product. Thus, the problem of modeling the influences between people is a vital task for studying social networks. Despite its importance, the network of social influence is usually hidden and not directly measurable. It is different from the "physical" connections in a social network which do not necessarily indicate direct influence. However, the network of social influence does manifest itself in the form of various time-stamped and recurrent events occurring at different individuals that are readily observable, and the dynamics of these historical events carry much information about how individuals interact and influence each other. For instance, one might decide to purchase a product the very next day when she saw many people around her bought it today while it may take the same person a prolonged period to try out the product if none from her community has adopted it. From a modeling perspective, we also want to take into account several key features of social influence.

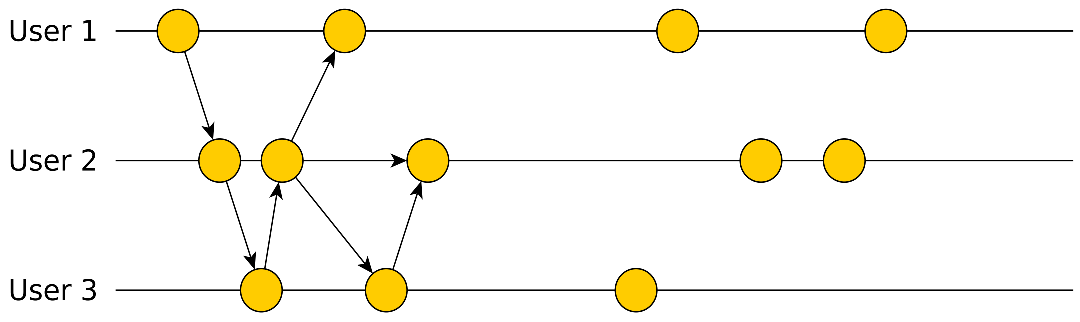


Figure 3: An illustration of recurrent mutually-exciting process.

First, many social actions are recurrent in nature. For instance, an individual can participate in a discussion forum and post her opinions multiple times. Second, actions between interacting people are often self- and mutually-exciting. The likelihood of an individual's future participation in an event tends to increase if she has participated in it before and more so if many of her neighbors also have participated in the event (See Figure 3). Third, the network of social influence have certain topological structures. It is usually sparse, *i.e.* most individuals only influence a small number of neighbors, while there are a small number of hubs with wide spread influence on many others. Moreover, people tend to form communities, with the likelihood of taking an action increased under the influence of other members of the same community (assortive, *e.g.,* peer-to-peer relation) or under the influence of members from another specific community (dissortative, *e.g.,* teacher-to-student relation) [46, 101]. These topological priors give rise to the structures of the adjacency matrices corresponding to the networks: they tend to have a small number of nonzero entries and they also have sophisticated low-rank structures.

In this part of the thesis, we propose a regularized convex optimization approach to discovering the sparse and low-rank hidden network of social influence based on a multi-dimensional Hawkes process. The multi-dimensional Hawkes process captures the mutually-exciting and recurrent nature of individual behaviors, while the regularization using nuclear norm and ℓ_1 norm simultaneously on the infectivity matrix allows us to impose priors on the network topology (sparsity and low-rank structure). The advantage of our formulation is that the corresponding network discovery problem can be solved efficiently by bringing a large collection of tools developed in the optimization communities. In particular, we developed an algorithm, called ADM4, to solve the problem efficiently by combining the idea of alternating direction method of multipliers [25] and majorization minimization [64]. In our experiments on both synthetic and real world datasets, the proposed method performs significantly better than alternatives in term of accurately discovering the hidden network and predicting the response time of an individual, especially when the underlying networks are sparse and low-rank.

1.5 Learning Nonparametric Kernels for Hawkes Process

Real world interactions between multiple entities, such as earthquake aftershocks [156], civilian death in conflicts [85] and user behaviors in social network [104], often exhibit the self-exciting and mutually-exciting property. For example, the time of aftershocks are usually close to the main shock and may trigged further aftershocks in the future. Multi-dimensional Hakwes processes, an important class of mutually exciting process, can be used to capture these interactions.

Despite the usefulness of the mutually exciting property in real world problems, the actual dynamics of *how* previous events trigger future events, which is modeled by the *triggering kernels*, can be quite complex and vary a lot across different applications. For example, the dynamics of user behaviors in social network can be very different from those in earthquake after shocks or disease contagion. Moreover, these dynamics can be inherently complex since the diverse nature of user behaviors. Unfortunately, most existing work based on Hawkes processes assumes that the triggering kernels are known or chosen manually in advance, which trends to be oversimplified or even infeasible for capturing the problem complexity in many applications. Therefore, it is highly desirable to estimate the temporal dynamics in a principled data-driven way rather than relying on ad-hoc manual selections.

In this part of the thesis, we propose a general framework to estimate the triggering kernels of Hawkes processes from the recurrent temporal events that can be viewed as samples from the Hawkes processes — without the knowledge of the actual triggering structure in the events. The challenge of the problem arises not only from the fact that the parameters is in an infinite dimensional space but also they are coupled with each other in the likelihood function. To address the problem, we propose MMEL which applies idea of majorization minimization [64] to construct an upper-bound of the objective function at each iteration that decouples the parameters so that they can be optimized independently. Another novelty of our method is that we used the Euler-Lagrange equation to derive an ordinary differential equation (ODE) that the optimal trigger kernel should satisfy. This connection allows us to exploit the fruitful and well-developed techniques of ODE solvers. In our experiments on both synthetic and real world datasets, the proposed method performs significantly better than alternatives.

1.6 Outline

The rest of this dissertation is organized as follows: In Chapter 3, we describe the problem of cold-start recommendation and propose our solution that combines matrix factorization and decision tree to construct the interview process. In Chapter 4, we propose the method of binary coding to improve the efficiency of collaborative filtering. In Chapter 5, we describe the model based on mutual-exciting Hawkes process to infer the hidden influence network based on temporal events. Chapter 6, we describe the method for learning nonparametric kernel for Hawkes process through solving the Euler-Lagrange equation. The thesis is concluded in Chapter 7.

CHAPTER II

BACKGROUND AND RELATED WORK

In this chapter, we review the existing studies related to the thesis. In particular, we introduce the idea of low-rank matrix factorizations and its different formulations in Section 2.1. Then, we will briefly summarize existing studies on collaborative filtering and cold-start collaborative filtering in Section 2.2 and 2.2.3, respectively. The related work of binary coding is summarized in Section 2.3 with the focus on collaborative filtering, which is related to the work in Chapter 4. Finally, we survey the related work on point process in Section 2.4.

2.1 Low-Rank Matrix Factorization

We first review the matrix factorization in its original form and illustrate the basic idea of matrix factorization which is the basis of the thesis. In particular, we will describe the singular value decomposition method and discuss its optimality.

2.1.1 Singular Value Decomposition

Singular value decomposition (SVD) is among the earliest ideas of the applications of matrix factorizations to discovery the structure of the data. It has a wide range of applications and extensions in information retrieval, data mining and other related fields.

The SVD can be defined as follows: Given any matrix $X \in \mathbb{R}^{m \times n}$. Let the singular value decomposition of X be

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

where the matrices U and V are orthonormal and Σ is diagonal—

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{\min\{m,n\}} \end{bmatrix}.$$

The values σ_1, σ_2, ..., $\sigma_{\min\{m,n\}}$ are the singular values of the matrix X. Without loss of generality, we assume that the singular values are arranged in descending order, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}}$.

For dimension reduction, we approximate the original matrix X by a rank-K approximation $\hat{X}$. This is done with a partial SVD using the singular vectors corresponding to the K largest singular values.

$$\begin{aligned} \hat{X} &= \hat{U}\hat{\Sigma}\hat{V}^T \\ &= \begin{bmatrix} \boldsymbol{U}_1 & \dots & \boldsymbol{U}_K \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_K \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^T \\ \vdots \\ \boldsymbol{V}_K^T \end{bmatrix}. \end{aligned} \tag{1}$$

A nice property is that SVD produces the rank-K matrix $\hat{X}$ that minimizes the distance from X in terms of the Frobenius norm. Specifically, we have the following theorem by Eckart and Young: [42]:

Theorem 1. $\hat{X}$ *is a solution for the following optimization problem:*

$$\min_{\hat{X}} \|X - \hat{X}\|^2$$

subject to:

$$\dim \hat{X} = K$$

The above theorem ensures that the rank-K approximation produced by SVD is optimal in terms of Frobenius norm. The property makes it possible to apply SVD in a lot of applications to discover the low-dimensional latent structures from high dimensional data. For example, latent semantic indexing (LSI), a famous technique in information retrieval, to discover the semantic space of a corpus is based on applying SVD to the document-term matrix [39], with several extensions such as fold-in process and regularizations [17, 160, 167].

In this case, the matrix X represents the documents and terms of the corpus. Although X is typically sparse, its low-rank approximation $\hat{X}$ is generally not sparse. Thus, the low-rank approximation $\hat{X}$ can be viewed as a smoothed version of X, obtained by propagating the co-occurring terms in the document corpus. This smoothing effect is achieved by discovering a latent semantic space formed by the documents. Specifically, we can observe from Equation (1) that each document d can be represented by a K-dimensional vector $\hat{\boldsymbol{V}}_d$, which is the d-th row of the matrix $\hat{V}$. The relation between the representation of document d in term space $\boldsymbol{X}_d$ and the latent semantic space $\hat{\boldsymbol{V}}_d$ is given by

$$\boldsymbol{X}_d = \hat{U}\hat{\Sigma}\hat{\boldsymbol{V}}_d.$$

Similarly, each term v can be represented by the K-dimensional vector $\hat{\boldsymbol{U}}_v$ given by

$$\boldsymbol{T}_v = \hat{V}\hat{\Sigma}\hat{\boldsymbol{U}}_v.$$

Thus, LSI projects both terms and documents into a K-dimensional latent semantic space the approximates the original document-term matrix.

2.2 Collaborative Filtering

Many studies on recommendation systems have been focused on collaborative filtering approaches. These methods can be categorized into memory-based and model-based. The reader is referred to the survey papers [4, 148] for a comprehensive summary of collaborative filtering algorithms.

Recently, matrix factorization becomes a popular direction for collaborative filtering [61, 78, 114, 125, 142]. These methods are shown to be effective in many applications. Specifically, matrix factorization methods usually seek to associate both users and items with latent profiles represented by vectors in a low dimension space that can capture their characteristics. In [126], a convex relaxation for low rank matrix factorization is proposed and applied to collaborative filtering. A probabilistic model for extracting low dimensional profiles is studies in [61], in which latent variables are introduced to capture the users and items and the EM algorithm is used to estimate the parameters. More recently, fueled by the Netflix competition, several improvements have been proposed including the use of regularized SVD [114], and the idea of matrix factorization combined with neighborhood-based methods [78]. In the following section, we describe collaborative filtering formulated as low-rank matrix factorizations with incomplete observations.

2.2.1 Low-Rank Matrix Factorizations with Incomplete Observations for Collaborative Filtering

One drawback of the singular value decomposition is that it assumes that all entries of the matrix X is observed, which is not true in a lot of applications, including collaborative filtering. Specifically, in collaborative filtering, we only observe a small fractions of ratings from the users. Therefore, the rating matrix is only partially observed and the goal is to predict the missing entries in the matrix. Therefore, the low-rank matrix factorization with incomplete observations has been developed to solve the problem.

First, let r_{ij} denote the rating of user i for item j, where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. For example, in movie recommendation systems, r_{ij} can be rating in $\{1, 2, 3, 4, 5\}$ representing the degree of preference of user i for movie j, where the rating 5 represents the user i very likes the user j, while a rating 1 represent that the user i does not like the movie j. In collaborative filtering, only a small subset of ratings are observed, denoted by $\mathcal{K} = \{r_{ij} \,|\, (i, j) \in O\}$. The goal of collaborative filtering is to predict the unknown ratings based on ratings in $\mathcal{K}$. Collaborative filtering exploits a basic heuristic that similar users tend to rate similar items in a similar way. One important class of methods for collaborative filtering are based on matrix factorization [61, 142]. Specifically, we associate a K-dimensional vector $u_i \in \mathbb{R}^K$ for each user i and $v_j \in \mathbb{R}^K$ for each item j. The vectors u_i and v_j are usually called user profiles and item profiles since they are intended to capture the characteristics of users and items. The rating r_{ij} of user i for item j can be approximated by a similarity function of u_i and v_j in the low dimensional space, for example, $r_{ij} = u_i^T v_j$.

Given the set of known ratings $\mathcal{K}$, the parameters u_i and v_j can be estimated through fitting the training data by solving the following optimization problem:

$$\min_{u_i, v_j} \sum_{(i,j)\in\mathcal{K}} (r_{ij} - u_i^T v_j)^2.$$

Regularization terms such as the Frobenius norms on the profile vectors can be introduced to avoid overfitting. The problem can be solved by existing numerical optimization methods such as alternating minimization, stochastic gradient descent and LBFGS. Here, we summarize the alternating optimization, which is a special case of the coordinate descent algorithm [22], for its amenability for the cold-start settings in Chapter 3. Specifically, the optimization process performs the following two updates alternatively.

First, for $i = 1, 2, \ldots, N$, minimizing with respect to u_i with all $u_j, j \neq i$ and all v_j fixed:

$$u_i = \operatorname*{argmin}_{u_i} \sum_{(i,j)\in O} (r_{ij} - u_i^T v_j)^2,$$

which is a linear regression problem with squared loss. The closed form solution can be expressed as

$$u_i = \left(\sum_{(i,j)\in\mathcal{K}} v_j v_j^T \right)^{-1} \left(\sum_{(i,j)\in O} r_{ij} v_j \right)$$

Then, for $j = 1, 2, \ldots, M$, minimizing with respect to v_j with all $v_i, i \neq j$ and all u_i fixed:

$$v_j = \operatorname*{argmin}_{v_j} \sum_{(i,j)\in\mathcal{K}} (r_{ij} - u_i^T v_j)^2,$$

which is also a linear regression problem with similar closed-form solution.

2.2.2 Nuclear Norm Regularization: A Convex Formulation

Recently, the nuclear norm regularization is frequently used to enforce the low-rank structure in a matrix [11, 45, 77, 126, 143], with the advantage of providing a convex formulation to the matrix factorization problem. It can be also applied together with other regularizations to incorporate other prior knowledge [128]. In particular, the nuclear norm $\|A\|_*$ of a matrix A is defined as the sum of its singular values. Therefore, the formulation with nuclear norm for matrix factorization can be expressed as:

$$\min_A \sum_{(i,j)\in\mathcal{K}} (r_{ij} - a_{ij})^2 + \lambda\|A\|_*$$

It has been shown that the solution A to the above problem is low rank [142, 143]. Moreover, the above optimization problem is convex and thus has a unique global solution, which makes it attractive in a lot of applications. The problem can be solved efficiently for relatively large problem [100]. One contribution of our work in Chapter 5 is to apply these results of nuclear norm to social influence estimation problem and we also develop a new algorithm ADM4 for solving the corresponding optimization problem efficiently.

2.2.3 Cold-Start Collaborative Filtering

There have been several studies on the elicitation strategies for new user preferences. The work [117] surveys several guidelines for user preference elicitation. Several models such as [52, 122, 123] constructed the interview process with a static seed set selected based on measures such as informativeness, popularity and coverage. The recent work of [50] proposed a greedy seed selection algorithm by optimizing the prediction performance. Such seed-based methods are not completely satisfactory because the seeds are selected statically in batch, and they do not reflect the user responses during the interview process. In [122], while discussing the IGCN (Information Gain through Clustered Neighbors) algorithm, it was mentioned the idea of adaptively selecting the interview questions by fitting a decision tree to predefined user clusters. In particular, each node represents an interview question and the user is directed to one of the child nodes according to her response to the parent question. In [51], the authors proposed an algorithm that fits the decision to the users' ratings. This seems to provide a more disciplined approach than that based on the predefined user clusters discussed in IGCN. Our proposed framework goes one step further by integrating the decision tree construction into the matrix factorization framework of collaborative filtering. We should also mention that the decision tree structure used in those methods exhibit interesting resemblance to the Bayesian network approach in [26].

A complimentary line of research for solving the cold-start problem is to utilize the features of users or items. The content features can be used to capture the similarities between users and items and thus reduce the amount of data required to make accurate predictions [7, 21, 56, 113, 133, 144]. For example, the work of [7] utilizes features of items and users as the prior distribution for latent profiles in matrix factorization. The method described in Chapter 3 is based on the interview process and does not solely rely on features of users and items.

We also want to mention active learning for collaborative filtering [24, 58, 72, 115]. These methods usually select questions that are optimal with respect to some selection criteria, such as the expected value of information or the distance to the true user model. These methods generally are not suitable for interview process since the selection process usually involves optimizing the selection criteria. In fact, the complexity of existing active learning algorithms for collaborative filtering are often too large to make online interview.

2.3 Learning Binary Codes

The problem of learning binary codes for fast similarity search has been investigated in several studies [80, 91, 132, 161] Locality sensitive hashing tries to construct binary codes that preserve certain distance (e.g., L_p distance) between different points with high probability, which is usually archived by random projection [37,66]. In [136], the problem of learning effective binary codes is solved by utilizing the idea of boosting. The work of [132] proposes to learn binary codes making use of Restricted Boltzmann Machine (RBM) for fast similarity search of documents. Recent work focuses on constructing binary codes based on a given similarity function [109, 159, 161]. The basic idea is to apply spectral analysis techniques to the data and embedding data points into a low dimension space. For example, the work [161] investigate the requirements for compact and effective binary codes. Their solution relies on spectral graph partition, which can be solved by eigenvalue decomposition of the Laplacian matrix for the graph. It has been shown that these methods archive significant performance improvements in terms of preserving the similarity between data points. Although several extension of this method has been studied [60, 168], these methods only consider the problem of obtaining binary codes for one type of entities. However, in collaborative filtering problem considered in Chapter 4, two types of entities, users and items, are naturally involved and thus should be consider simultaneously, which makes it difficult to applies these method directly.

The preference of a user on an item is usually measured by some similarity, such as the dot-product, between their low dimensional profiles. These studies are related to our work in the sense that both of them aim to find certain representations that preserve the preference between users and items. However, one key difference is that our work in Chapter 4 aims to find binary codes in Hamming space for representing users and items, which has the nice property that the retrieval of interesting items for a user can be performed in time that is independent of the total number of items [132]. The work [170] proposes to learn binary codes for both documents and terms by viewing the term-document matrix as a bipartite graph, where edges represent the occurrences of terms in documents. Then, the method proposed in [161] is applied to adjacent matrix obtain the binary codes. However, this method can not deal with the problem of unobserved/missing ratings in collaborative filtering. As shown in our experiments, the binary codes obtained by this method quickly overfit the training data and lead to poor prediction accuracy.

Another direction of collaborative filtering investigate the problem of using binary codes to create fingerprints for users [14,15,36]. The idea is create binary fingerprints for each user using randomized algorithms so that the similarity between users can be approximated according to the fingerprints. However, these studies do not address the problem of simultaneously representing users and items by binary fingerprints. Thus, the preference of users on items can not be estimated directly from these fingerprints. On the other hand, the binary codes learnt by our proposed method in Chapter 4 can be directly used to measure the preference of users over items.

2.4 Self-Exciting Point Process

Self-exciting point processes are frequently used to model continuous-time events where the occurrence of one event increases the possibility of future events. Hawkes process [59], an important type of self-exciting process, has been investigated for a wide range of applications such as market modeling [151], earth quake prediction [97], crime modeling [146]. The maximum likelihood estimation of one-dimensional Hawkes process is studied in [86] under the EM framework. Additionally, [138] models cascades of events using marked Poisson processes with marks representing the types of events while [23] propose a model based on Hawkes process that models events between pairs of nodes. However, it only considers two dimensional Hawkes process and aims to model events associated with edges. The novelty of this work is the application of multi-dimensional Hawkes process [2, 59] to model the recurrent events on nodes of a social network, and leverage its connections to convex low-rank matrix factorization techniques.

Estimating the hidden social influence from historical events is attracting increasing attention recently. For instance, [111] proposes a hidden Markov based model to model the influence between people which treats time as discrete index and hence does not lead to models predictive of the response time. The approach in [54] models the probability of a user influenced by its neighbors by a sub-modular functions, but it is not easy to incorporate recurrent events and topological priors in a principled way. In [107, 129], continuous-time models are proposed to recover sparse influence network. However, our work is different from theirs in two aspects: First, the models can proposed in [107, 129] can not handle recurrent events as we did in Chapter 5. Moreover, they only take into account the sparse network structure, while we consider the estimation of networks that are both low-rank and sparse.

2.5 Estimating the Triggering Kernels

The problem of nonparametric estimation of the triggering kernels has been addressed for special cases such as the one-dimensional Hawkes processes [86] or symmetric Hawkes processes [16]. In this work, we study the general case of multi-dimensional Hawkes process under the framework for optimizing the triggering kernels in the infinite dimensional functional space. The recent work of [41] considers the problem of learning the triggering kernel for diffusion processes based on linear combination of basis functions. Our proposed method, on the other hand, does not assume any known parametric forms of basis functions and estimate them from observed data through optimization in an infinite dimensional functional space.

Another related direction of studies is smoothing splines [124, 158] and the extensions such as kernel methods and Gaussian Processes [137], in the sense that the problem of estimating the triggering kernels can be viewed as a smoothing problem with nonnegative constraints. The goal of smoothing splines is to estimate a smooth function based on its value on finite points. The nonnegative constraints are usually studied as the more general case of the shape restrictions [96, 124, 155]. The main difference in our work is that the loss function we consider is more complex and depends on the values of the triggering kernels on infinite points, which makes it difficult to directly apply the smoothing spline methods. Moreover, as we will see later, the nonnegative constraints can be naturally enforced in our algorithm.

Positive definite kernels have also been used extensively in machine learning. This type of kernel can be viewed as similarity function between data points. Its learning has been addressed extensively in recent literature where one tries to learn a better positive definite kernel by combining several positive definite kernels [10, 12, 34, 40, 141]. Nonparametric positive kernel learning, instead of learning combination of existing positive kernels, directly learns the full Gram matrix with respect to certain constraints and prior knowledges about the data, such as pairwise constraints [62] or distribution of the data [173].

CHAPTER III

FUNCTIONAL MATRIX FACTORIZATIONS FOR COLD-START RECOMMENDATION

In this chapter, we describe *functional matrix factorization* (fMF), a novel method for the construction of the interview process for cold start recommendation. We argue that it is more effective to combine the matrix factorization model for collaborative filtering and the decision-tree based interview model into a single framework. Our proposed method is based on the low rank factorization of the incomplete user-item rating matrix with an important twist: it restricts the user profiles to be a *function* of the answers to the interview questions in the form of a decision tree, thus the name functional matrix factorization. The function — playing the role of the initial interview — is in the form of a decision tree with each node being an interview question [50, 122].

We will describe the proposed framework, functional matrix factorization, in detail in Section 3.1. The learning algorithm based on alternative optimization is then proposed with detailed derivations. Then, we evaluate the proposed method on three data sets and analyze the results in Section 3.3.

3.1 Functional Matrix Factorization

In this section, we describe the *functional matrix factorization* (fMF) method for cold-start collaborative filtering which explores the well-known matrix factorization methods for constructing the interview process. The key innovation is that we parameterize user profiles to be a function of the responses to the possible questions of the interview process and use matrix factorization to compute the profiles.

3.1.1 Functional Matrix Factorization

We consider constructing the interview process for cold-start collaborative filtering. Assume that a new user registers at the recommendation system and nothing is known about her. To capture the preferences of the user, the system initiates several interview questions to query the responses from the user. Based on the responses, the system constructs a profile for the user and provides recommendations accordingly.

In the plain matrix factorization model described in Section 2.1, the user profile u_i is estimated by optimizing the ℓ_2 loss on the history ratings r_{ij}. This model does not directly apply to cold-start settings because no rating is observed for the new user prior to the interview process. To build user profiles adaptively according to the user's responses in the course of the interview process, we propose to parameterize the user profile u_i in such a way that the profile u_i is tied to user i's responses in the form of a function, thus the name functional matrix factorization (fMF). More precisely, assume there are P possible *interview questions*. We assume that an answer to a question takes value in the finite set $\{0, 1, \text{Unknown}\}$, representing "Dislike", "Like" and "Unknown", respectively. Furthermore, let a_i denote the P-dimensional vector representing the answers of user i to the P questions. And we tie the profile to the answers by assuming $u_i = T(a_i)$, where T is a function that maps the responses a_i to the user profile $u_i \in \mathbb{R}^k$. To make recommendations for user i, we simply use

$r_{ij} = v_j^T T(a_i)$.

Our goal is to learn both T and v_j from the observed ratings $\mathcal{K}$. To this end, substituting $u_i = T(a_i)$ into the low-rank matrix factorization model, we have the following optimization problem:

$$T, V = \underset{T \in \mathcal{H}, V}{\operatorname{argmin}} \sum_{(i,j) \in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|V\|^2, \tag{2}$$

where $V = (v_1, \ldots, v_M)$ is the matrix of all item profiles, $\mathcal{H}$ is the space from which the function $T(a)$ is selected and the second term is the regularization term.

Several issues need to be addressed in order to construct the interview process by the above functional matrix factorization. First, the number of all possible interview questions can be quite large (e.g. up to millions of items in movie recommendation); yet a user is only patient enough to answer a few interview questions. Second, the interview process should be adaptive to user's responses, in other words, a follow-up question should be selected based on the user's responses to the previous questions. Therefore, the selection process should be efficient to generate interview questions in real time after the function $T(a)$ is constructed. In addition, since we allow a user to choose "Unknown" to the interview questions, we need to deal with such missing values as well.

Following prior works of [51,122], we use a ternary decision tree to represent $T(a)$. Specifically, each node of the decision tree corresponds to an interview question and has three child nodes. When the user answers the interview question, the user is directed to one of its three child nodes according to her answer. As a result, each user follows a path from the root node to a leaf node during the interview process.

A user profile is estimated at each leaf node based on the users' responses, i.e., $T(a)$. The number of interview questions presented to any user is bounded by the depth of the decision tree, generally a small number determined by the system. Also, non-responses to a question can be handled easily in the decision tree with the introduction of a "Unknown" branch.

3.1.2 Alternative Optimization

The objective function defined in Equation (2) can be optimized through an alternating minimization process. Specifically, we alternate between the following two steps:

1. Given $T(a)$, we can compute v_j by regularized least square regression. Formally, for each j, we find v_j such that

   $$v_j = \underset{v_j}{\operatorname{argmin}} \sum_{(i,j)\in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|v_j\|^2.$$

   This problem has a closed-form solution given by

   $$v_j = \left(\sum_{(i,j)\in O} T(a_i)T(a_i)^T + \lambda I \right)^{-1} \left(\sum_{(i,j)\in O} r_{ij} T(a_i) \right), \tag{3}$$

   where I is the identity matrix of appropriate size.

2. Given v_j, we try to fit a decision tree T such that

   $$T = \underset{T\in\mathcal{H}}{\operatorname{argmin}} \sum_{(i,j)\in O} (r_{ij} - T(a_i)^T v_j)^2. \tag{4}$$

A critical challenge is that the number of possible trees grows exponentially with the depth of the trees, which can be extremely large even for trees of limited depth. It is therefore computationally extensive to obtain a global optimal solution for the above. We address this problem by proposing an efficient greedy algorithm for finding an approximate solution.

3.1.3 Decision Tree Construction

Traditional decision tree algorithms such as C4.5 or CART [27] usually minimize objective functions such as classification error and square loss for regression. In our scenario, the objective function is defined in Equation (4), and we now describe how to build a decision tree in a greedy and recursive fashion to minimize it. Specifically, at each node, we select the best interview question by optimizing the objective defined in Equation (4) based on the response to this question; the decision tree then splits the user set into three subsets corresponding to the child nodes (i.e. "Like", "Dislike" and "Unknown"). We carry out this procedure recursively until the tree reaches certain depth. In our experiments, the depth is usually set to a small number between 3 and 7.

Formally, starting from the root node, the set of users at current node are partitioned into three disjoint subsets $R_L(p)$, $R_D(p)$ and $R_U(p)$ corresponding to "Like", "Dislike" and "Unknown" of their responses to the interview question p:

$$R_L(p) = \{i | a_{ip} = \text{ “Like”}\},$$

$$R_D(p) = \{i | a_{ip} = \text{ “Dislike”}\},$$

$$R_U(p) = \{i | a_{ip} = \text{ “Unknown”}\}.$$

To find the optimal question p that leads to the best split, we minimize the following objective:

$$\min_{p} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u_L^T v_j)^2 + \sum_{i \in R_D(p)} \sum_{(i,j) \in O} (r_{ij} - u_D^T v_j)^2$$
$$+ \sum_{i \in R_U(p)} \sum_{(i,j) \in O} (r_{ij} - u_U^T v_j)^2, \quad (5)$$

where u_L, u_D and u_U are the optimal profiles for users in the child nodes corresponds to the answers of "Like", "Dislike" and "Unknown", respectively:

$$u_L = \operatorname*{argmin}_{u} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2,$$
$$u_D = \operatorname*{argmin}_{u} \sum_{i \in R_D(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2,$$
$$u_U = \operatorname*{argmin}_{u} \sum_{i \in R_U(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2.$$

There also exist closed-form solutions to u_L, u_D and u_D as follows:

$$u_L = \left(\sum_{i \in R_L(p)} \sum_{(i,j) \in O} v_j v_j^T \right)^{-1} \left(\sum_{i \in R_L(p)} \sum_{(i,j) \in O} r_{ij} v_j \right), \quad (6)$$
$$u_D = \left(\sum_{i \in R_D(p)} \sum_{(i,j) \in O} v_j v_j^T \right)^{-1} \left(\sum_{i \in R_D(p)} \sum_{(i,j) \in O} r_{ij} v_j \right), \quad (7)$$
$$u_U = \left(\sum_{i \in R_U(p)} \sum_{(i,j) \in O} v_j v_j^T \right)^{-1} \left(\sum_{i \in R_U(p)} \sum_{(i,j) \in O} r_{ij} v_j \right). \quad (8)$$

After the root node is constructed, its child nodes can be constructed in a similar way, recursively. We summarize our algorithms for functional matrix factorization and decision tree construction in Algorithm 1 and Algorithm, 2 respectively.

Implementation and Computational Complexity. The time complexity for computing v_j with Equation (3) is $O(MK^3)$, where M is the number of items and K is the dimension of the latent space. In order to compute u_L, u_D and u_U at each split, we need to compute the inverse of a square matrix of size K, which takes $O(K^3)$ time. The brute-force approach for generating the matrix itself requires $O(UMK^2)$ time. In order to reduce the time complexity, we observe that the coefficient matrix can be computed based on the sum of ratings, the sum of squares of ratings and the number of ratings for each item within time $O(MK^2)$. With a similar method proposed in [51], the time complexity of computing these statistics of all items is $O(\sum N_i^2)$ for each level of the decision tree, where N_i is the number of ratings by the user i. Thus, the computation complexity for constructing the decision tree is $O(D\sum_i N_i^2 + LMK^3 + LM^2K^2)$, where D is the depth of the tree and L represents the number of nodes in the tree. The computation time can be reduced by selecting a subset of items based on some criteria such as the rating variance, and using only the selected items as candidates for the interview questions.

3.1.4 Hierarchical Regularization

In the above section, u_L, u_D and u_U for each node are estimated by linear regression. When the amount of training data is small, the estimate may overfit the training data. The problem becomes more severe especially for the nodes close to the leaves of the decision tree, because as the split process progresses, users belonging to those nodes become fewer and fewer. To avoid overfitting, regularization terms need to be introduced. Although traditional ℓ_2 regularization can be applied in this case [114], but such approach does not take into account the rich structure of the decision tree.

Here, we propose to apply hierarchical regularization that utilize the structure of the decision tree. Specifically, we shrink the coefficient u of a node toward the one at

Algorithm 1 Alternating Optimization for Functional Matrix Factorization

```
Input: The training data $\mathcal{K} = \{r_{ij} \mid (i, j) \in O\}$.
Output: Estimated decision tree $T(a)$ and item profiles $v_j, j = 1, 2, \ldots, M$.
 1: Initialize $v_j$ randomly for $j = 1, \ldots, M$.
 2: while not converge do
 3:    Fit a decision tree $T(a)$ using Algorithm 2.
 4:    Update $v_j$ with Equation (3).
 5: end while
 6: return $T(a)$ and $v_j, j = 1, 2, \ldots, M$.
```

Algorithm 2 Greedy Decision Tree Construction

```
 1: function FitTree(UserSet)
 2: // UserSet: the set of users in current node.
 3: Calculate $u_L$, $u_D$, $u_U$ by Equation (6), (7) and (8) for $p = 1, \ldots, P$.
 4: Compute the split criteria $L_p$ by Equation (5) for $p = 1, \ldots, P$.
 5: Find the best interview question $p = \text{argmax}_p L_p$.
 6: Split user into three groups $R_L(p)$, $R_D(p)$ and $R_U(p)$.
 7: if square error reduces after split and depth < maxDepth then
 8:    Call FitTree($R_L(p)$), FitTree($R_D(p)$) and FitTree($R_U(p)$) to construct the child
       nodes.
 9: end if
10: return $T(a)$.
11: end function
```

its parent node. For example:

$$u_L = \underset{u}{\text{argmin}} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2 + \lambda_h \|u - u_C\|^2,$$

where u_C is the estimation at the current node and u_L is the estimation at its child node corresponding to the answer “Like” and the parameter λ_h controls the trade-off between training error and regularization. We will evaluate the impact of λ_h in our experiments. It seems that $\lambda_h = 0.03$ gives good results in practice. The similar idea using parent node as the prior for the child node is also studied in [51], but the ratings for the parent node is used to improve the prediction in the child node.

3.2 Extensions

In this section, we describe some extensions to the proposed fMF model in order to demonstrate the flexibility of the proposed method. In particular, we first describe how to extend the model to ask multiple questions at each node. Then, we generalize the proposed model to Partially observable Markov decision process (POMDP) framework [30] which has a nice connection to matrix factorizations.

3.2.1 Multiple Questions

The proposed method can also be extended to ask multiple questions at each decision tree split. To this end, there are two challenges to overcome. First, the number of possible questions can be quite large (e.g., we may have $n \sim 10^5$ in movie recommendation system) and searching over all possible splits becomes a combinatorial problem with a prohibitive $\binom{n}{l}$ possibilities to evaluate where l is the number of questions at each node. We solve this problem by relaxing it as an L_1 regularized optimization. Second, since we are keeping the structure of the tree with 3 branches, it is possible that users with different opinions would enter the same node. Therefore, different from other decision tree approaches, a non-constant function T needs to be learned separately for each node. We solve this second problem by training a linear regressor within each node using all the previously obtained answers as input. For splitting with multiple items, the idea is to optimize for a sparse weight vector on each item that has only a few non-zeros. Formally, let w denote the weight of items, which is a n-dimensional vector. Let l denote the maximum number of questions at each split. Training users at the current node are split into 3 child nodes L, D and U according to the linear combination of the answers $x_i^T w$.

We use a modified logistic probability model to determine the split. Let p_i, q_i denote the probability that user i belongs to the L, and D branch respectively:

$$p_i = \frac{1}{1 + c\exp(-x_i^T w)}$$
$$q_i = \frac{1}{1 + c\exp(x_i^T w)}$$

Accordingly, the three subgroups are defined as:

$$L(w) = \{i | p_i > q_i, p_i > 1 - p_i - q_i\},$$

$$D(w) = \{i | q_i > p_i, q_i > 1 - p_i - q_i\},$$

$$U(w) = \{i | 1 - p_i - q_i \geq \max(p_i, q_i)\},$$

where c is a parameter controlling the likelihood of falling into different groups. In practice, we find that $c = 2$ works best. In such a case, a user belongs to the L group when $x_i^T w$ is positive, the D group when x_i^T is negative, and the middle group U only when the user answers none of the questions.

Ideally, we need to optimize such an objective function:

$$\begin{aligned} \min_{w, T_L, T_D, T_U} & \sum_{i \in L(w)} \sum_{j \in \mathcal{O}_i} (r_{ij} - T_L(x_i)^T v_j)^2 \\ & + \sum_{i \in D(w)} \sum_{j \in \mathcal{O}_i} (r_{ij} - T_D(x_i)^T v_j)^2 \\ & + \sum_{i \in U(w)} \sum_{j \in \mathcal{O}_i} (r_{ij} - T_U(x_i)^T v_j)^2 \end{aligned} \tag{9}$$

subject to

$$\|w\|_0 \leq \ell$$

where T_L, T_D and T_U are the profile functions for the nodes L, D, U, and $\|w\|_0$ denotes the number of non-zeros in w. The objective function first divides the training users into three child nodes, and then computes the squared error within each child node. The goal is to minimize the sum of errors in all the three child nodes. In addition, the constraint $\|w\|_0 \leq \ell$ determined that w cannot have more than l non-zeros. We would adopt an alternating minimization strategy that optimizes w and T_L, T_D, T_U iteratively. However, the optimization of w is a complicated non-convex combinatorial problem, therefore we make relaxations to solve it with continuous optimization. In the next two subsections we discuss separately the optimization of w and the computation of T_L, T_D, and T_U.

3.2.2 Optimization

In order to update the weight vector w in Equation (9) with continuous optimization, we adopt two relaxations. Firstly, we relax the hard partitioning L, D, and U into a soft partitioning, with the probability of x belonging to the subgroups L, D and U to be p_i, q_i, and $1-p_i-q_i$, respectively. This makes the main objective function smooth in w. The second relaxation is to append a penalty term on $\|w\|_1$ to the objective function, instead of a hard constraint on the number of non-zeros in w. This ℓ_1 relaxation approach has been popular in machine learning and signal processing in recent years. The ℓ_1 term is convex and thus there exists efficient methods to optimize it. Let $\bar{m}$ denote the number of users reaching the current node, our relaxation problem computes w which minimizes the weighted prediction loss:

$$\begin{aligned}\min_{w} &\sum_{i=1}^{\bar{m}} p_i \sum_{j \in \mathcal{O}_i} (r_{ij} - T_L(x_i)^T v_j)^2 \\ &+ \sum_{i=1}^{\bar{m}} q_i \sum_{j \in \mathcal{O}_i} (r_{ij} - T_D(x_i)^T v_j)^2 \\ &+ \sum_{i=1}^{\bar{m}} (1 - p_i - q_i) \sum_{j \in \mathcal{O}_i} (r_{ij} - T_U(x_i)^T v_j)^2 + \lambda \|w\|_1. \end{aligned} \tag{10}$$

In order to optimize Equation (10), we adopt a projected scaled sub-gradient optimization. This algorithm computes the Hessian only for the nonzero part of the current w, and adopts linear-time Barzilai-Borwein subgradient steps [20, 134] for the rest. It is suitable for our task because of its fast convergence rate thanks to the incorporated second-order information, and each iteration is a linear-time operation. The only cubic-time computation is to compute the Hessian inverse on the nonzeros. Since we have usually less than 5 nonzeros, this step would normally take constant time. By changing the parameter λ, we can find solutions with different numbers of nonzeros in w. In practice, we first binary search between 0 and $\lambda_{\max}$ to obtain λ_0 such that the number of nonzero entries of w is between 1 and ℓ. Then we search around λ_0 with a finer step size, locating ℓ different solutions with 1,2, $\cdots$, ℓ nonzeros respectively. From this solution set, we select the one that minimize Equation (9) without the constraint. We find this scheme to work well in practice.

3.2.3 POMDP formulation for cold start problems

When the choice of which question to ask in the interview process is viewed as a decision, the construction of the interview process can be naturally done in the framework of Markov decision process [118], which is the theoretical framework to model a sequential decision-making process. Formally, a Markov decision process which consists of a state space S, a set of actions A, the transition probabilities $\Pr(s'|s, a)$ and a cost function $C(s, a, s')$. More specifically:

- **States.** The set of states S consists of all question sequences of all lengths with all possible answers.

- **Actions.** The set of actions A consists of M question action, one for each candidate interview question. The action set A also consists the set of termination actions $\{\phi\}$ of predicting the user profile to be ϕ.

- **Transition Probability.** The probability $\Pr(s'|s, a)$ can be estimated from the rating data.

- **Immediate Cost.** We can define the cost of actions as follows: For termination actions $\{\phi\}$, the cost function can be defined by the negative prediction error of the user profile ϕ. Specifically, we have

$$C(s, \phi, \Psi) = \sum_{u \in s} \sum_{i:(u,i) \in \mathcal{O}} (r_{ui} - \psi_i^T \phi)^2.$$

  For question actions, we can use the cost $C(s, m) = c_m$ of asking question m, which models the cognitive burden of answering the question.

The goal is to find an optimal *policy* $\pi : S \to A$, which assigns an action $\pi(s)$ to each state s, that minimizes the expected overall cost.

Incorporating the formalism of MDP, the objective function in Equation (2) can be expressed as follows:

$$\min_{\phi,\pi} V^{\pi}(s_0)$$

where $V^{\pi}(s)$ is the *expected* cost starting from state s under *policy* π:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=1}^{D} C_t | s, \pi, \phi, \Psi\right],$$

Moreover, $s_0 = \{\}$ is the initial state, i.e., the empty set when no question has been asked. Notice that the cost during the interview process $\sum_{t=1}^{D} C_t = C(s_D, \phi, \Psi) + \sum_{t=1}^{D-1} C(s_t, a_t)$ is a generalization of the square loss function in Equation (22). In Equation (22), we have $C(s_t, a_t) = 0$ for $t = 1, \ldots, D-1$, since we do not consider the cost of answering the questions.

The problem can also be solved by alternating minimization. Given ψ and π, user profiles π can be obtained by solving least square problems. Given item profiles ψ and user profiles ϕ, the problem of constructing the policy π can be formulated by the *planing* problem of a Markov decision process. The goal is to find a policy $\pi(s)$ that for every state s returns the action that minimize the expected cost starting $V^{\pi}(s)$ from the state s. With the computed ϕ's, we can use least squares fitting to recompute Ψ as discussed before, and this process will repeat until convergence. We then use the policy π and the user profiles ϕ to construct the interview process $T(a)$ as follows: We start from the initial state $s_0 = \{\}$. For each state s, if $\pi(s)$ is a question action, we ask the question corresponds to $\pi(s)$ and modify the state to according the answer a. If $\pi(s)$ is a terminate action, we return the corresponding user profile ϕ as the value of $T(a)$.

Due to the large size of state space and action space, we need to reduce their size in order to make the formulation tractable. One idea to reduce the size of the state space in MDP is through clustering which naturally leads to exploring partially observed Markov decision process (POMDP) [118], where the states of the system is unobserved. In particular, we can formulate the cold start problem as a POMDP as follows:

- **States.** The set of hidden states S with $|S| = k$.
- **Actions.** The set of actions A consists of M question action, one for each candidate interview question. The action set A also consists the set of termination actions $\{\phi\}$ of predicting the user profile to be ϕ.
- **Transition Probability.** The probability $\Pr(s'|s, a)$ can be modeled by multinomial distribution estimated from data.
- **Observation Probability.** $\Pr(o|s')$ represents the distribution of answers of users at state s'.
- **Immediate Rewards.** For termination actions $\{\phi\}$, the rewards function can be defined by the negative prediction error of the user profile ϕ. For question actions, the rewards function can be generalized to capture the cost $C(m)$ of asking question m, which models the cognitive burden of answering the question.

In POMDP policy consists of mappings from the set of *belief* states $b(s)$, i.e., distributions over the hidden states, to the set of actions. In particular, $b(s)$ represents a probability distribution over the state space, which is real-valued vector in $\mathbb{R}^k$.

This enables us to make a natural connection between POMDP and the latent factor/matrix factorization based framework: we can equate the belief states $b(s)$ with a user profile ϕ — the states in POMDP then correspond to the factors/dimensions in latent factor/matrix factorization models. Considering the usually interpretation of the latent factors as correspond to different genres in movie recommendation, this gives an intuitive interpretation of the states of the POMDP. Thus, the predicted ratings can be expressed by

$$r_{ui} = \mathbb{E}_{b_u}[v_i] = \sum_s b_u(s) v_{is} = b_u^T v_i,$$

where v_i is the profile for item i. The update rule for belief $b(s)$ after taking action a and observe o can be expressed by:

$$b'(s') = \frac{1}{Z} \Pr(o|s') \sum_{s \in S} \Pr(s'|s, a) b(s).$$

Again, we can jointly optimize both the policy π and the item profiles v_i.

3.3 Experiments

In order to evaluate the proposed method for cold-start collaborative filtering, we carry out a set of controlled experiments on three widely used benchmark data sets: MovieLens, EachMovie and Netflix.

3.3.1 Data sets and Evaluation Metrics

We first briefly describe the data sets.

- The MovieLens[1] data set contains 3900 movies, 6040 users and about 1 million

[1]http://www.grouplens.org/node/73

ratings. In this data set, about 4% of the user-movie dyads are observed. The ratings are integers ranging from 1 (bad) to 5 (good).

- The EachMovie data set, collected by HP/Compaq Research, contains about 2.8 million ratings for 1628 movies by 72,916 users. The ratings are integers ranging from 1 to 6.

- The Netflix[2] is one of the largest test bed for collaborative filtering. It contains over 100 million ratings and was split into one training and one test subsets. The training set consists of 100,480,507 ratings for 17,770 movies by 480,189 users. The test set contains about 2.8 million ratings. All the ratings are ranged from 1 to 5.

The performance of an collaborative filtering algorithm will be evaluated in terms of the widely used root mean square error (RMSE) measure, which is defined as follows

$$\text{RMSE} = \left(\frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} (r_{ij} - \hat{r}_{ij})^2 \right)^{1/2},$$

where $\mathcal{T}$ represents the set of test ratings, r_{ij} is the ground-truth values for movie j by user i and $\hat{r}_{ij}$ is the predicted value by a collaborative filtering model. We can also normalize the error of each user by the number of their ratings. Our initial results show that the two metrics are consistent in general, so we only report RMSE in our experiments.

[2]http://www.netflixprize.com/

Table 1: Statistics for Data Sets

Dataset	No. Users	No. Items	No. Ratings
MovieLens	6040	3900	1,000,209
EachMovie	72,916	1628	2,811,983
Netflix	480,189	17,770	104,706,033

3.3.2 Deriving Interview Responses from User Ratings

The responses a_i of the interview process for user i is required in order to train and evaluate the interview process. Following the standard settings [51, 122], we restrict the format of interview questions to be "Do you like item j?" For example, for movie recommendation system, we can ask questions like "Do you like the movie Independence Day?" Then, we can infer the responses for existing users according to their ratings for the corresponding items. For instance, with ratings in 1-5 scale, we assume that the responses a_{ij} of user i to the question "Do you like item j?" can be inferred from her rating r_{ij} as follows:

$$a_{ij} = \begin{cases} 0, & \text{if } (i,j) \in O \text{ and } r_{ij} \leq 3 \\ 1, & \text{if } (i,j) \in O \text{ and } r_{ij} > 3 \\ \text{"Unknown"}, & \text{if } (i,j) \notin O \end{cases}$$

Intuitively, we assume that the user will response 0 (*dislike*) and 1 (*like*) for items she rates with ≤ 3 rating or > 3 rating. If the user did not rate an item selected for the interview process, we assume that she will respond with *unknown.* For EachMovie data set, we use a similar method except setting the threshold to be 4 since its ratings range from 1 to 6.

3.3.3 Experiment Design

We seek to address the following questions:

1. For new users, how does the proposed algorithm perform? In particular, how effective is the interview process? Moreover, how does the proposed algorithm perform compared to the baseline algorithms?

2. We utilize the ratings from users to simulate their responses to the interview questions in our evaluation. How do missing values in user ratings impact the performance of the proposed algorithm?

3. How does fMF perform in warm-start settings compared to traditional collaborative filtering methods such as plain matrix factorization?

4. How do the parameters impact the performance of the proposed model?

3.3.4 Cold-Start Performance

We first evaluate the performance of fMF in cold-start settings. For each data set, we split the users into two disjoint subsets, the training set and the test set, containing 75% and 25% users, respectively. The users in the training set are assumed to be warm-start users whose ratings are known by the system. We learn the models and construct the interview process based on these training users. In contrast, the users in the test set are assumed to be cold-start users. The ratings of each user in the test set are partitioned into two sets: the first set is called the *answer set* which is used to generate the user responses in the interview process while the second set is called the *evaluation set* which is used to evaluate the performance after the interview process.

The sizes of the answer set and the evaluation set are 75% and 25% of the rated items of each user, respectively. Note that the interview process typically includes only a small number (≤ 7 in our experiments) of questions although the answer set contains 75% of the ratings when deriving the responses for the cold-start users. The evaluation process is summarized in Figure 4.

We compare the performance with two baseline methods, named as Tree and TreeU and briefly described as follows:

- Tree: this is the method proposed in the very recent work of [51]. The method learns a decision tree for initial interview without using latent user/item profiles. It fits the tree based on the ratings of all the items in the inventory. Therefore, at the leaf node, it needs to estimate M ratings for all the M items in the system.

- TreeU: this method learns decision tree and matrix factorization through two separate steps. It first estimates the user profiles and item profiles through plain matrix factorization described in Section 2.1; then, a decision tree is constructed separately using the algorithm of [51] to fit the latent user profiles. The model predicts ratings by using the user profiles from the decision tree and the item profiles from matrix factorization.

Our proposed fMF algorithm differs from these two algorithms in that it is a natural integration of both decision tree and matrix factorization.

We use the following parameter settings: For TreeU, we use as default the regularization weight of $\lambda = 0.01$, which gives the best RMSE in our experiments. For Tree

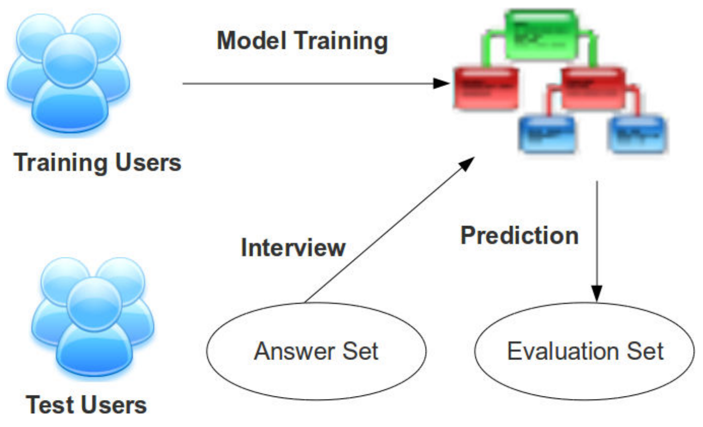


Figure 4: Evaluation process for cold-start users

and fMF, we consider only one question at each node and apply 4-fold cross validation to determine the parameters.

The results on MovieLens, EachMovie and Netflix data sets are reported in Figure 5, Table 6 and Table 7, respectively. Our first observation is that, as expected, the performance is gradually improved (i.e. the RMSE decreases) as the number of interview questions increases. This is true for all three methods, suggesting that these three algorithms are all capable of refining user preference through the interview process. Therefore, all the three methods can be applied to solve the cold-start problem.

Comparing the performance of fMF and Tree, we can see that fMF consistently outperforms Tree in all the three data sets. The improvements are significant according to t-test with significance level $p = 0.05$. This observation illustrate that the interview processes learned by fMF is more effective than those by Tree. We attribute this to the fact that fMF naturally integrates the matrix factorization model into the interview process for cold-start collaborative filtering. In particular, fMF inherits the ability of matrix factorization models in *collaboratively* uncovering the user-item ratings. In contrast, Tree assumes that users/items are independent from one another, and therefore cannot capture the vital collaborative effect.

We can also see that TreeU does not work well, too. Recall that TreeU is a two-stage method — It first extracts the user profiles and item profiles using plain matrix factorization, and then constructs a decision tree on the answers of the interview questions. A possible reason for why it does not perform well is that the user profiles and item profiles obtained from the matrix factorization in warm-start setting usually capture the refined preferences of users and refined characteristics of items. However, in cold-start setting, we are only allowed to ask users a few questions so that the model usually captures only very coarse interests of users. As a result, TreeU usually fails to fit the user profiles from matrix factorization accurately. Thus, its prediction accuracy is relatively low. On the other hand, our method estimates the decision tree and item profiles simultaneously in a combined optimization process. Thus, the user profiles obtained by decision tree and the item profiles can adapt to each other and improve the prediction accuracy.

To provide more comprehensive views of the interview process, we further look into particular cases. In Table 2 and Table 3, we present the interview process for users in Netflix data set as well as the top-5 recommendations for them after the interview process. From Table 2, we can see that the user chooses "Like" on the movie Armegeddon and Reservoir Dogs, which belong to Fiction and Adventure movies. The recommendation includes Lord of the Rings series and Star Wars. They are quite related to the movie Armegeddon based on their genres. Similarly, the interview

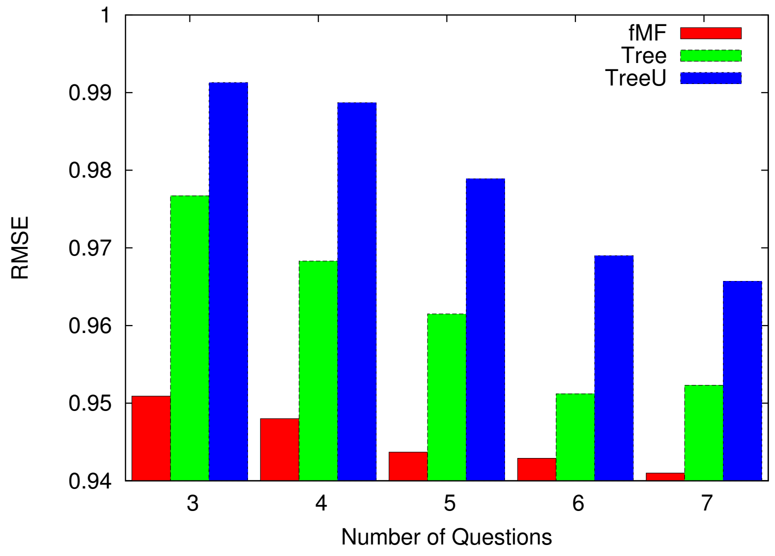


Figure 5: RMSE on MovieLens data set for cold-start users with respect to the number of interview questions

process of Table 3 shows that the user likes Drama and Romance movies and the top recommendations for her contain both those two types of movies. Those results illustrate that the recommendations generated by fMF are indeed reasonable.

3.3.5 The Impact of Non-responses

In our experiments, we utilize the ratings of users to simulate their responses to the interview questions as described in Section 3.3.2 since the users' responses to the interview questions are not available for the benchmark data sets. In particular, we assume that the user will respond "Unknown" to an interview question if she does not rate the corresponding item. This assumption, however, might be inaccurate in practice. For example, a user's rating to an item might be missing simply because she does not have time to rate it. In this case, the user may actually responds with "Like" or "Dislike" rather than "Unknown" if she is asked to respond to the

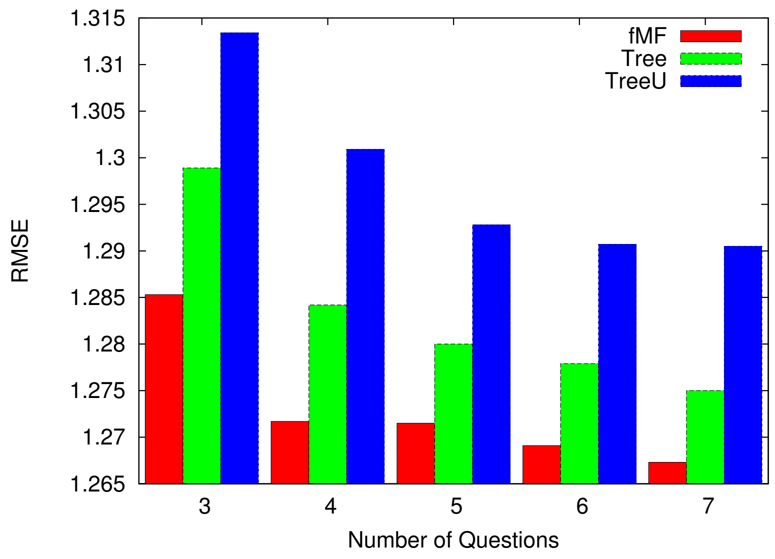


Figure 6: RMSE on MovieLens data set for cold-start users with respect to the number of interview questions

interview questions. As a result, the missing values in data sets may introduce bias in the training and interview process.[3]

Here, we explore how the missing ratings influence the initial interview process. We investigate the performance of the proposed method on users with different numbers of ratings. To this end, we sort users in the test set by the their numbers of ratings in descending order and then plot, in Figure 8, the performance measured by RMSE with respect to the fraction of users with the most ratings. We can see that the RMSE increases when more users with few ratings are included, which indicates that the performance for users with more ratings is better then the ones with less ratings. This is because that the users with more ratings are less likely to select "Unknown" in our simulation. Thus, they are less likely prone to the influences by

[3]The bias in training data seems to have been largely ignored in previous work.

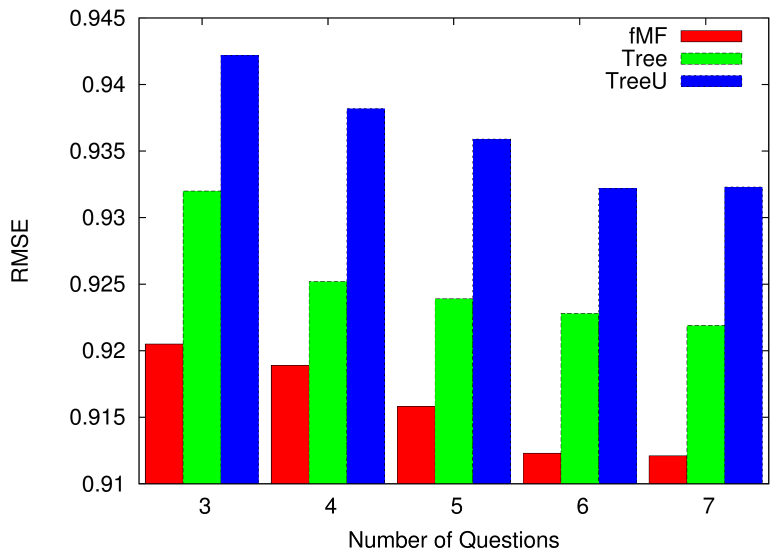


Figure 7: RMSE on MovieLens data set for cold-start users with respect to the number of interview questions

the bias introduced by the missing values in the training data.

With this basic understanding, we carry out a set of experiments to investigate the impact of missing values in training data. For each of the three data sets, we sort users in training set in descending order. Then, the users in the test set (labeled by MT) is sampled from top 20% users with the most ratings. We only consider the users with sufficiently large number of ratings since we would like to rule out the impact of bias during the interview process and focus on the missing values for training users. We generate five training sets, namely M1 to M5, from the rest of the users in the training set, including the top 20%, 40%, 60%, 80% and 100% of the users that are not selected in the test set, respectively. Again, 75% ratings for each user in MT is used to simulate the interview process and the performance is evaluated on the remaining 25% ratings.

Table 2: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Dislike
2	Armageddon	Like
3	Maid in Manhattan	Dislike
4	Reservoir Dogs	Like
5	Collateral Damage	Dislike
6	2 Fast 2 Furious	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	Lord of the Rings: The Return of the King
2	Lord of the Rings: The Two Towers
3	Mobile Suit Gundam: Char's Counterattack
4	Star Wars: Episode V: The Empire Strikes Back
5	Raiders of the Lost Ark

We perform experiments by training the proposed models on M1 to M5 and evaluate them on MT for all three data sets. The results are shown in Figure 9. It can be observed from Figure 9 that the prediction error measured by RMSE increases when more users with few training data are included in the training set. This observation suggests that the performance are indeed affected by the bias introduced by the missing values in training set.

3.3.6 Warm-Start Performance

In order to answer the third question proposed in Section 3.3.3, we evaluate the proposed algorithm in warm-start settings and show the relative performance between the cold-start methods and warm-start methods. In particular, we consider the matrix factorization method described as follows:

Table 3: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Like
2	Armageddon	Dislike
3	What Women Want	Dislike
4	The Royal Tenenbaums	Unknown
5	Pretty Woman	Like
6	Cats: The Ultimate Edition	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	The Shawshank Redemption
2	Schindler's List
3	Sex and the City: Season 5
4	Sex and the City: Season 4
5	The Usual Suspects

- MF: The matrix factorization method described in Section 2.1 is included for comparison. We include ℓ_2 regularization to avoid overfitting with $\lambda = 0.01$. The method is not designed for the cold-start problem and thus it requires the user ratings in order to construct user profiles.

We include this method to present a concrete comparisons of the relative performance of the proposed cold-start method and the warm-start methods that are widely studies in previous researches. We compare the proposed algorithm (fMF) with all three methods for warm-start settings: Tree, TreeU and MF. To this end, we perform experiments with different depths of decision trees and report the RMSE of the all the methods over three data sets. For MovieLens and EachMovie data set, we randomly split the ratings for each users into training set and test set and perform 4-fold cross validation. For Netflix data set, we follow the popular evaluation protocol on this data

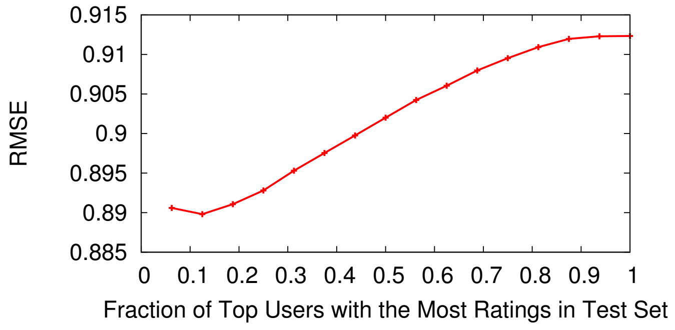


Figure 8: Performance measured by RMSE on Netflix data set with respect to the fraction of users with the most known ratings.

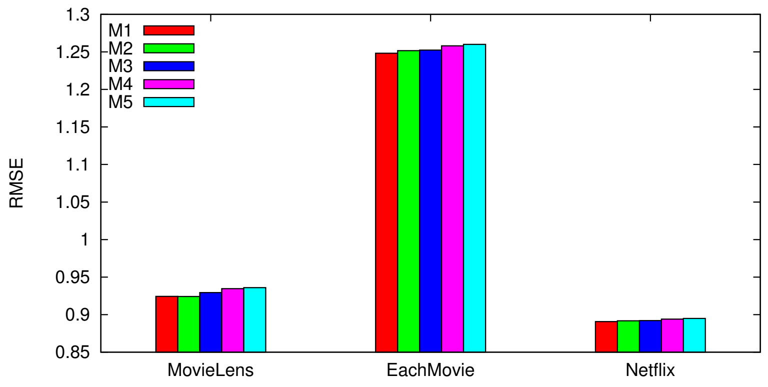


Figure 9: Performance measured by RMSE on three data sets with respect to the fraction of users with the most known ratings.

set. Therefore, we train our models on the training set and report the performance on the test set. The performance measured by RMSE with respect to the depths of decision trees is reported in Table 4, Table 5 and Table 6.

We can observe that all the cold-start methods perform worse than the matrix factorization method. This observation is consistent with our expectation because all the three cold-start algorithms are constrained in some ways in order to deal with cold-start users — the goal of the cold-start algorithms is to provide cold-start users with reasonable predictions within a few quick interview questions. For example, the fMF model is restricted in its maximum depth. We should expect comparable performance, if this constraint is eliminated.

As an empirical validation, we further carry out experiments on the MovieLens data set by fitting the decision trees in our model with relatively larger depth. We note that the decision trees of very large depth correspond to interview processes with many questions, which are not proper for initial interview process for cold-start users since users are typically not willing to answer many questions. However, we can show that the performance of the proposed method can be quite close to the matrix factorization method if we relax the constraint and allow the model to use decision trees with large depth. We report the RMSE of fMF with respect to the depth of the decision tree in Figure 10. We also include Tree and MF for comparison. We can see that the RMSE of fMF monotonically decreases as the depth of decision trees increases. Moreover, its performance can be quite close to the matrix factorization method (MF). When the depth of the trees grows, the number of leaf nodes increases. Therefore, there are only a few users at each node. Thus, the user profiles estimated by fMF can be quite close to those by MF. We conclude that our method can be a reasonably good method of general collaborative filtering as well. On the other hand, we can see that the performance of Tree is much worse than MF even with a large number of questions. Moreover, the gap between fMF and Tree becomes larger when the depth of the decision tree grows. This is because Tree predicts the ratings of different items independently at each leaf node, which is not a good model for collaborative filtering in general. We also perform similar experiments with the cold start setting. In this case, the RMSE of fMF decreases with the depth of the decision

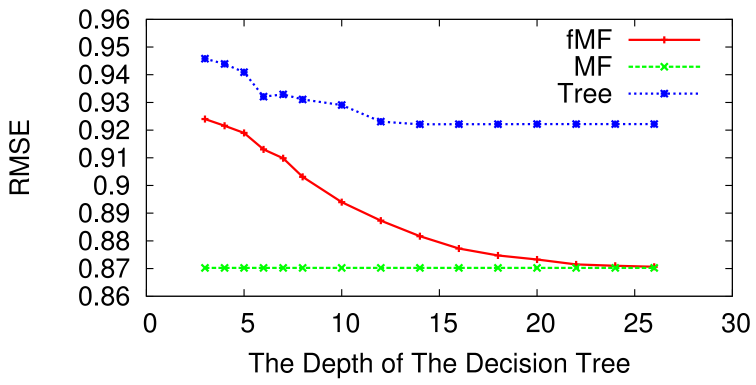


Figure 10: RMSE of fMF, Tree and MF on MovieLens data set with respect to the number of interview questions

tree when the depth is not very large (e.g. ≤ 14). We emphasize that the depth of the decision tree is usually set to be a small number since we can not ask a lot of questions to users during the interview process.

3.3.7 Impact of Model Parameters

There are several parameters that affect the performance of the proposed model. In this section, we carry out experiments to investigate the impacts of these parameters. We only report the results on MovieLens data set as the observations are similar when other data sets are sed. By default, the cold-start setting described in Section 3.3.4 is applied unless otherwise stated.

The parameter K controls the dimension of the user profiles and item profiles for the matrix factorization model. We fix the depth of the decision tree to be 6, and report the performance obtained with different K. The results are depicted in Figure 11. Particularly, as K increases, the factorization model becomes more and more flexible, as a results, the RMSE first decreases, and reaches the optima

Table 4: RMSE on MovieLens data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9240	0.9216	0.9190	0.9134	0.9098
Tree	0.9458	0.9439	0.9409	0.9321	0.9329
TreeU	0.9906	0.9850	0.9771	0.9706	0.9728
MF	0.8702				

Table 5: RMSE on EachMovie data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	1.2548	1.2499	1.2449	1.2366	1.2226
Tree	1.2709	1.2614	1.2664	1.2570	1.2549
TreeU	1.28742	1.2813	1.2790	1.2772	1.2751
MF	1.1790				

Table 6: RMSE on Netflix data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9770	0.9749	0.9721	0.9715	0.9703
Tree	0.9772	0.9761	0.9726	0.9717	0.9713
TreeU	0.9914	0.9879	0.9842	0.9792	0.9752
MF	0.9399				

around $K = 20$; thereafter, the model becomes increasingly overparameterized and the performance in turn starts to degrades. For example, the performance with $K = 30$ is worse than the that with $K = 20$. In principle, the optimal K should provide the best trade-off between fitting bias and model complexity.

One of our contributions is that we propose to use hierarchical regularization to avoid overfiting. The impact of the hierarchical regularization is controlled by the parameter λ_h. We evaluate fMF with different values of λ_h on MovieLens data set. The performance obtained by ℓ_2 regularization is also reported for comparison. We can see from Figure 12 that the hierarchical regularization always performs better than

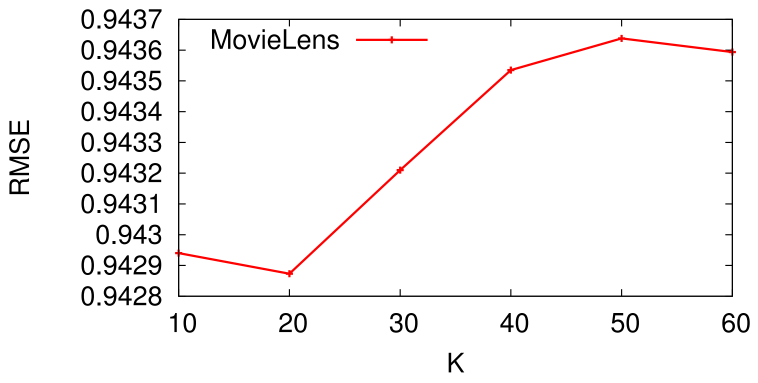


Figure 11: Performance measured by RMSE with respect to different values of K on MovieLens data set. The performance is reported by setting the depth of the decision tree to be 6.

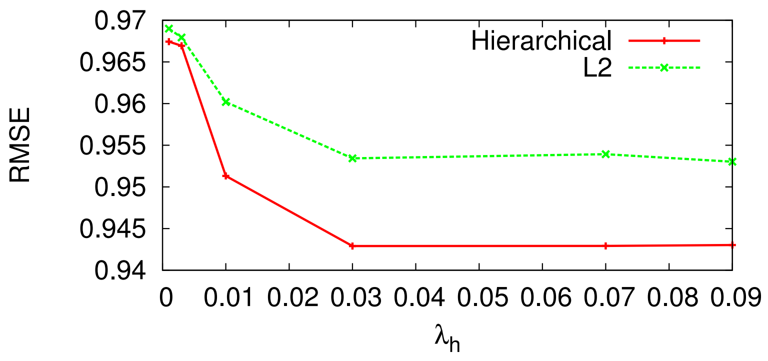


Figure 12: The performance of hierarchical regularization. The performance is reported by setting the depth of the decision tree to be 6.

the basic ℓ_2 regularization. This observation suggest that hierarchical regularization, by exploiting the structure of the decision tree, provides better regulatory to the functional matrix factorization model.

Another parameter of interest is the regularization weight λ for the item profiles v_j. We vary the value of λ and report the performance measured by RMSE in Figure 13. We can see that the optimal performance is achieved when a moderate λ is used.

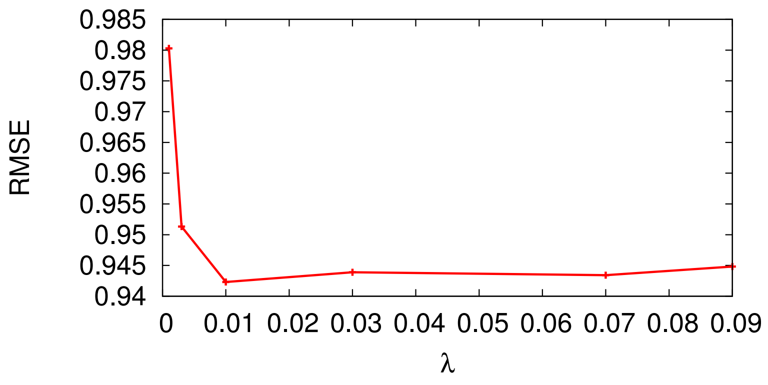


Figure 13: Performance measured by RMSE with respect to different values of λ on MovieLens data set.

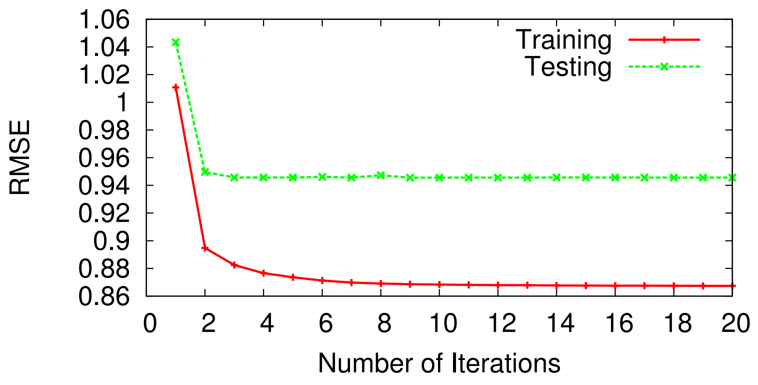


Figure 14: Performance measured by RMSE with respect to the number of iterations on MovieLens data set.

Especially, the model has a high risk of overfitting if λ is too small.

The learning algorithm of the proposed model is an iterative process. In Figure 14, we plot the both training and test performance measured by RMSE with respect to the number of iterations. We can see that the algorithm converges very quickly, usually within 5 iterations. The running time of algorithm depends largely on the size of the data sets and the number of questions asked. For MovieLens and EachMovie data set, it can usually finish within a few hours, while for Netflix data set, it can take a day to fit a model.

3.4 Summary

The main focus of this chapter is on the cold-start problem in recommender systems. We have presented the functional matrix factorization, a framework for simultaneously learning the decision tree for initial interview and latent factors for user/item profiling. The proposed fMF algorithm seamlessly integrates matrix factorization for collaborative filtering and decision-tree based interview into one unified framework by reparameterizing the latent profiles as a function of user responses to the interview questions. We have established learning algorithms based on alternating minimization and demonstrated the effectiveness of fMF on real-world recommendation benchmarks.

CHAPTER IV

LEARNING BINARY CODES FOR COLLABORATIVE FILTERING

As we discussed in Section 1.3, it is desirable to design efficient algorithms for online recommendation systems, especially for systems with a lot of items where even a linear scan of all items are unaffordable in order to respond to user requests in real-time. In this chapter, we address the problem of learning binary codes for collaborative filtering. Specifically, we propose to learn compact yet effective binary codes for both users and items from the training rating data. The learnt binary codes enable us to perform recommendations efficient in time that are independent to the number of items.

Our approach is based on the idea that the binary codes assigned to users and items should preserve the preferences of users over items. Two loss functions are applied to measure the divergence between the training data and the estimates based on the binary codes. Unfortunately, thus formulated, the resulting discrete optimization problem is difficult to solve in general. Through relaxing the binary constraints, it turns out the relaxed optimization problem can be solved effectively by existing solvers. Moreover, we propose two effective methods for rounding the relaxed solutions to obtain binary codes. One key property of the binary codes obtained by the proposed method is that the degree of preferences of a user to items can be measured by the number of common bits between their corresponding binary codes. Hence, the major advantage of representing users and items by binary codes is to enable fast search: In order to provide recommendations for a user, we need only to search items with binary codes within a small Hamming distance to the binary codes of the given user.

In Section 4.1, we first formulate the problem of learning binary codes for collaborative filtering as a discrete optimization problem and introduce the two loss functions used in this work. Then, the learning algorithm proposed with detailed derivations based on transforming and relaxing the discrete optimization problem so that it can be optimized efficiently. Moreover, we discuss two different methods for rounding real-valued solutions to obtain binary codes. The evaluations are described and analyzed in Section 4.2. We conclude our work and present several future research directions in Section 4.3.

4.1 Learning Binary Codes

In this section, we describe the proposed method for learning binary codes for collaborative filtering. We first describe the general formulation for this problem through optimization using squared and pairwise loss functions, respectively. Then, the learning method based on solving the relaxed problem is derived in detail. Finally, we discuss two methods to obtain binary codes from the real-valued solutions of the relaxed problems.

4.1.1 Problem Formulation

The goal of collaborative filtering is to recommend interesting items to users according to their past ratings on the items. Formally, we assume that r_{ui} represents the rating of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$, where $\mathcal{U}$ and $\mathcal{I}$ are the user and item space, respectively. Without loss of generality, we further assume that r_{ui} is a real number in the interval $[0, 1]$. Moreover, we assign binary codes $f_u \in \{-1, 1\}^B$ for each user u and $h_i \in \{-1, 1\}^B$ for each item i, where B is the length of the binary codes. Our goal is to construct binary codes for users and items that preserve the preferences between them — the degree of preference of user u over item i can be estimated by the similarity between their binary codes f_u and h_i. A natural way to define the similarity between user u and item i is the fraction of common bits in their binary codes f_u and h_i, leading to the similarity function,

$$\mathrm{sim}(f_u, h_i) = \frac{1}{B} \sum_{k=1}^{B} \mathbb{I}(f_u^{(k)} = h_i^{(k)}),$$

where $f_u^{(k)}$ and $h_i^{(k)}$ represent the k-th bit of the binary codes f_u and h_i, respectively. $\mathbb{I}(\cdot)$ denotes the indicator function that returns 1 if the statement in its parameter is true and zero otherwise.

It is easy to check that the following holds for the similarity function $\mathrm{sim}(\cdot, \cdot)$ defined above:

$$\mathrm{sim}(f_u, h_i) = 1 - \frac{1}{B} \mathrm{dist}_{\mathrm{H}}(f_u, h_i),$$

where $\mathrm{dist}_{\mathrm{H}}(f_u, h_i)$ is the Hamming distance between two binary codes f_u and h_i. The above fact suggests that the smaller the Hamming distance is, the more similar their binary codes become. Therefore, in order to find items with similar binary codes to a user represented by f_u, it is sufficient to search items i within a small Hamming distance $\text{dist}_{\text{H}}(f_u, h_i)$. This allows us to find similar items in time that is independent to the total number of items [132].

In order to make accurate recommendations to users, we need to find binary codes f_u and h_i for users and items such that the preferences between them are preserved by the similarities between their respective binary codes. In addition, in collaborative filtering, we only observe a subset of all the possible ratings $\{r_{ui} | (u, i) \in \mathcal{O}\}$ where $\mathcal{O} \subset \mathcal{U} \times \mathcal{I}$ and we need to recommend items to users according to their preferences over items whose ratings are unobserved. Therefore, the key for accurate recommendations is to construct binary codes that can not only preserve the observed ratings but also accurately predict the preferences of users on unobserved items.

Our approach to learn binary codes is to estimate them from observed ratings. Specifically, we propose to construct binary codes that minimize the degree of divergence between the observed ratings and the ratings estimated from the binary codes. To this end, we apply two objective functions to measure the degree of divergence between the observed ratings and the model estimates:

- **Squared Loss.** Using this loss function, we seek to minimize the squared error of the observed ratings and the similarity estimations from the binary codes, which is a commonly used loss function for collaborative filtering:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{sq} = \sum_{(u,i) \in \mathcal{O}} (r_{ui} - \text{sim}(f_u, h_i))^2. \tag{11}$$

- **Pairwise Loss.** Since we are more interested in preserving the relative orders between items rather than their absolute values, it is natural to consider the pairwise loss function described as follows:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{pair} = \sum_u \sum_{i,j \in \mathcal{O}_u} \Big((r_{ui} - r_{uj}) - (\mathrm{sim}(f_u, h_i) - \mathrm{sim}(f_u, h_j)) \Big)^2, \quad (12)$$

where $\mathcal{O}_u = \{(u,i) \mid (u,i) \in \mathcal{O}\}$. Minimizing the above loss function requires the binary codes to preserve the relative difference between each pair of items rated by the user. The loss function has been applied to learning ranking functions for information retrieval [33].

Additionally, we also require the binary codes to be *balanced* — we would like each bit of the binary codes to have equal chance to be 1 or -1. The balance constraint is equivalent to maximizing the entropy of each bit of the binary codes, which indicates that each bit carries as much information as possible. Specifically, we will enforce the following constraints to the binary codes:

$$\sum_u f_u = 0, \quad \text{and} \sum_i h_i = 0.$$

The above constraints motivate the following regularized objective function for learning binary codes. For example, for squared loss, we have the following objective function:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_{(u,i) \in \mathcal{O}} (r_{ui} - \mathrm{sim}(f_u, h_i))^2 + \lambda(\| \sum_u f_u \|^2 + \| \sum_i h_i \|^2), \quad (13)$$

where the first term is the loss over observed ratings and the second term represents that we prefer balanced binary codes. The parameter λ controls the trade-off between minimizing the empirical errors and the enforcement of the constraints. $\| \cdot \|$ indicates Euclidean norm of a vector.

Similarly, we have the following regularized objective function for the pairwise loss function:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_u \sum_{i,j \in \mathcal{O}_u} \Big((r_{ui} - r_{uj}) - (\mathrm{sim}(f_u, h_i) - \\ \mathrm{sim}(f_u, h_j)) \Big)^2 + \lambda (\| \sum_u f_u \|^2 + \| \sum_i h_i \|^2). \tag{14}$$

4.1.2 Learning

The objective functions defined in Equation (13) and Equation (14) are defined over the discrete space $\{\pm 1\}^B$, which makes them difficult to optimize in general. Therefore, we propose to solve it approximately by transforming the objective functions and then relaxing the space of solutions to be $[-1, 1]^B$. For the sake of concreteness, we describe our method for solving the squared loss in Equation (13) in detail. The pairwise loss in Equation (14) can be optimized in a similar approach.

First, we notice that for binary codes $f, h \in \{\pm 1\}^B$, the following property holds:

$$\begin{aligned}
\mathrm{sim}(f, h) &= \frac{1}{B} \sum_{k=1}^{B} \mathbb{I}(f^{(k)} = h^{(k)}) \\
&= \frac{1}{2B} \left(\sum_{k=1}^{B} \mathbb{I}(f^{(k)} = h^{(k)}) + (B - \sum_{k=1}^{B} \mathbb{I}(f^{(k)} \neq h^{(k)})) \right) \\
&= \frac{1}{2B} \left(B + \sum_{k=1}^{B} f^{(k)} h^{(k)} \right) \\
&= \frac{1}{2} + \frac{1}{2B} f^T h.
\end{aligned}$$

Thus, by substituting the above equation into the regularized objective function defined in Equation (13), we can express the objective function for squared loss as follows:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{reg} = \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B} f_u^T h_i))^2$$
$$+ \lambda(\| \sum_u f_u \|^2 + \| \sum_i h_i \|^2). \qquad (15)$$

A widely used approach to obtain approximate solutions to the above discrete optimization problem is to relax the space of solution to be real values and thus enables the application of the continuous optimization techniques to solve the problem. To this end, we first relax the space of solution to be real vectors in $[-1, 1]^B$ and then we will round the real-valued solutions into $\{\pm 1\}^B$. The details of rounding are discussed in Section 4.1.3.

It is also interesting to note that the above formulation also reveals a nice connection between learning binary codes and the matrix factorization approaches that are widely applied in collaborative filtering. In particular, the first term in (15) is the objective function that factorizes the linearly transformed matrix of observed ratings to find low-dimensional representations for users and items. The second term is different from the usual ℓ_2 regularization used in traditional matrix factorization since we would like the binary codes to balanced rather than close to zero in this case.

Given the relaxed problem, the partial derivatives of the objective function $\mathcal{L}_{reg}$ with respect to f_u and h_i can be expressed as follows:

$$\frac{\partial \mathcal{L}_{reg}}{\partial f_u} = -\frac{1}{B} \sum_{i \in \mathcal{O}_u} (r_{ui} - \frac{1}{2} - \frac{1}{2B} f_u^T h_i) h_i + 2\lambda \sum_{u'} f_{u'},$$
$$\frac{\partial \mathcal{L}_{reg}}{\partial h_i} = -\frac{1}{B} \sum_{u \in \mathcal{O}_i} (r_{ui} - \frac{1}{2} - \frac{1}{2B} f_u^T h_i) f_u + 2\lambda \sum_{i'} h_{i'}.$$

The relaxed problem can be solved by methods such as LBFGS [172] and stochastic gradient descent.

Similarly, for the pairwise loss, we can compute the gradient as follows:

$$\frac{\partial \mathcal{L}_{reg}}{\partial f_u} = -\frac{1}{B} \sum_{i,j \in \mathcal{O}_u} \left((r_{ui} - r_{uj}) - \frac{1}{2B} f_u^T (h_i - h_j) \right) (h_i - h_j) + 2\lambda \sum_{u'} f_{u'},$$
$$\frac{\partial \mathcal{L}_{reg}}{\partial h_i} = -\frac{1}{B} \sum_{u \in \mathcal{O}_i} \sum_{j \in \mathcal{O}_u, j \neq i} \left((r_{ui} - r_{uj}) - \frac{1}{2B} f_u^T (h_i - h_j) \right) f_u + 2\lambda \sum_{i'} h_{i'}.$$

4.1.3 Obtaining Binary Codes

After solving the relaxed optimization problem defined in Equation (15), we obtain real-valued vectors $\tilde{f}_u$ and $\tilde{h}_i \in [-1, 1]^B$ for each user u and item i. In this section, we propose two methods to obtain binary codes f_u and $h_i \in \{\pm 1\}^B$ from these real-valued vectors.

4.1.3.1 Rounding to Closest Binary Codes

A straightforward method is to find binary vectors f_u and $h_i \in \{\pm 1\}^B$ that are closest to $\tilde{f}_u$ and $\tilde{h}_i$. Specifically, we seek to optimize the following objective function to obtain f_u for all $u \in \mathcal{U}$:

$$\min_{f_u \in \{\pm 1\}^B} \sum_u \| f_u - \tilde{f}_u \|^2 \tag{16}$$

subject to $\sum_u f_u = 0$. Similarly, we can obtain h_i for all item $i \in \mathcal{I}$ by:

$$\min_{h_i \in \{\pm 1\}^B} \sum_i \| h_i - \tilde{h}_i \|^2 \tag{17}$$

subject to $\sum_u h_i = 0$.

It turns out that the optimization problems defined in Equation (16) and Equation

(17) have the following solution:

$$f_u^{(k)} = \begin{cases} 1, & \tilde{f}_u^{(k)} > \text{median}(\tilde{f}_u^{(k)} : u \in \mathcal{U}), \\ -1, & \text{Otherwise}, \end{cases}$$

and

$$h_i^{(k)} = \begin{cases} 1, & \tilde{h}_i^{(k)} > \text{median}(\tilde{h}_i^{(k)} : i \in \mathcal{I}), \\ -1, & \text{Otherwise}, \end{cases}$$

where median$(\cdot)$ represents the median of a set of real numbers. The above solutions can be verified by first observing that they satisfies the constraints from the definition of the median. Also, any feasible solution can be obtained by switching the codes for some pairs of users u, u' such that $\tilde{f}_u^{(k)} < \tilde{f}_u'^{(k)}$. Moreover, we show that if $\tilde{f}_u^{(k)} \leq \tilde{f}_u'^{(k)}$, $[(\tilde{f}_u^{(k)}+1)^2 + (\tilde{f}_u'^{(k)}-1)^2] - [(\tilde{f}_u^{(k)}-1)^2 + (\tilde{f}_u'^{(k)}+1)^2] = 4(\tilde{f}_u^{(k)} - \tilde{f}_u'^{(k)}) \leq 0$. Therefore, switching the code can not improve the objective function. Similarly, we can verify the solution for item codes.

4.1.3.2 Improved Rounding by Orthogonal Transformations

Another method to obtain the binary codes from the relaxed solution $\tilde{f}_u$ and $\tilde{h}_i$ makes use of the structure of the solutions to the relaxed optimization problem. Similar ideas have been investigated for spectral clustering [166] and we extend the idea to the context of learning binary codes. First, we observe that if $\tilde{f}_u$ and $\tilde{h}_i$ are optimal solutions for (15), then $Q\tilde{f}_u$ and $Q\tilde{h}_i$ are also optimal solutions achieving the same value of the objective function for an arbitrary orthogonal matrix $Q \in \mathbb{R}^{B \times B}$, i.e.,

$Q^TQ = I$. This observation can be proved as follows:

$$\begin{aligned}
\mathcal{L}_{reg}(Q\tilde{f}_u, Q\tilde{h}_i) &= \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B}(Q\tilde{f}_u)^T(Q\tilde{h}_i))^2 + \lambda(\|\sum_u (Q\tilde{f}_u)\|^2 + \|\sum_i (Q\tilde{h}_i)\|^2) \\
&= \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B}\tilde{f}_u^T\tilde{h}_i)^2 + \lambda(\|\sum_u \tilde{f}_u\|^2 + \|\sum_i \tilde{h}_i\|^2) \\
&= \mathcal{L}_{reg}(\tilde{f}_u, \tilde{h}_i),
\end{aligned}$$

where the second equation utilizes the fact the Q is an orthogonal matrix. The above observation shows that applying orthogonal transformations to an optimal solution of relaxed optimization problem does not change the value of the objective function, which motives the following method to obtain binary codes from the relaxed solution:

$$\min_{Q, f_u, h_i \in \{\pm 1\}^B} \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2 \tag{18}$$

subject to:

$$\sum_u f_u = 0, \quad \sum_i h_i = 0, \quad Q^TQ = I.$$

Intuitively, instead of directly finding binary codes that are close to the relaxed solutions, we seek binary codes that are close to some orthogonal transformation of the relaxed solutions. Introducing the orthogonal transformation Q not only preserves the optimality of the relaxed solutions but provides us more flexibility to obtain better binary codes.

The optimization problem defined in Equation (18) can be solved by minimizing with respect to f_u, h_i and Q alternatively.

Optimization with respect to f_u and h_i. Specifically, we first fix the orthogonal transformation Q and optimizing with respect to f_u and h_i:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2$$

subject to $\sum_u f_u = 0, \sum_i h_i = 0$. The solution can be expressed as follows:

$$f_u^{(k)} = \begin{cases} 1, & (Q\tilde{f}_u)^{(k)} > \text{median}((Q\tilde{f}_u)^{(k)} : u \in \mathcal{U}), \\ -1, & \text{Otherwise,} \end{cases}$$

and

$$h_i^{(k)} = \begin{cases} 1, & (Q\tilde{h}_i)^{(k)} > \text{median}((Q\tilde{h}_i)^{(k)} : i \in \mathcal{I}), \\ -1, & \text{Otherwise} \end{cases}$$

where $(Q\tilde{f}_u)^{(k)}$ represents the k-th element of the transformed vector $Q\tilde{f}_u$.

Optimization with Respect to Q. In this case, we fix f_u and h_i for all $u \in \mathcal{U}$ and $i \in \mathcal{I}$. Then we solve the following optimization problem to update the orthogonal transformation Q:

$$\begin{aligned} \min_{Q \in \mathbb{R}^{B \times B}} L(Q) &= \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2 \\ &= \|F - Q\tilde{F}\|_F^2 + \|H - Q\tilde{H}\|_F^2 \end{aligned} \tag{19}$$

subject to the constraint $Q^T Q = I$, where $F = [f_1, \dots, f_{|\mathcal{U}|}]$, $\tilde{F} = [\tilde{f}_1, \dots, \tilde{f}_{|\mathcal{U}|}]$, $H = [h_1, \dots, h_{|\mathcal{I}|}]$ and $\tilde{H} = [\tilde{h}_1, \dots, \tilde{h}_{|\mathcal{I}|}]$. $\|\cdot\|_F$ indicates the Frobenius norm. The following theorem enables us to solve the optimization problem efficiently by singular value decomposition:

Theorem 2. *Let UDV^T be the singular value decomposition of the matrix $(F\tilde{F}^T + H\tilde{H}^T)$. Then, $Q = UV^T$ minimizes the objective function defined in Equation (19).*

Proof. First notice that the objective function $L(Q) = \|F\|^2 + \|\tilde{F}\|^2 - \mathrm{tr}(F\tilde{F}^TQ^T) + \|H\|^2 + \|\tilde{H}\|^2 - \mathrm{tr}(H\tilde{H}^TQ^T)$. Therefore, the optimization problem is equivalent to maximizing the following function subject to the orthogonality constraint $Q^TQ = I$:

$$\mathrm{tr}(F\tilde{F}^TQ^T) + \mathrm{tr}(H\tilde{H}^TQ^T) = \mathrm{tr}((F\tilde{F}^T + H\tilde{H}^T)Q^T).$$

Let us consider the Lagrange

$$L(Q, \Lambda) = \mathrm{tr}((F\tilde{F}^T + H\tilde{H}^T)Q^T) - \frac{1}{2}\mathrm{tr}(\Lambda(Q^TQ - I)),$$

where Λ is a symmetric matrix. By taking the gradient with respect to Q, we have

$$(F\tilde{F}^T + H\tilde{H}^T) - \Lambda Q = 0,$$

Thus, $\Lambda = (F\tilde{F}^T + H\tilde{H}^T)Q^T = UDV^TQ^T$, which implies that $\Lambda^2 = UD^2U^T$. Hence, $\Lambda = UDU^T$. Substituting it into the above equation, we have $Q = UD^{-1}U^TUDV^T = UV^T$. $\square$

In general, we perform the above two steps alternatively until the solution converges and obtain the binary codes f_u and h_i.

4.2 Experiments

In this section, we describe the experiments conducted to evaluate the proposed method for learning binary codes. For the sake of simplicity, we denote the proposed method with squared loss and pairwise loss defined in Equation (13) and Equation (14) by CFCodeReg and CFCodePair, respectively.

4.2.1 Evaluation Metrics

We apply two evaluation metrics to evaluate the performance of CFCodeReg and CFCodePair. Our goal is to evaluate whether the obtained binary codes can accurately preserve the preferences of users to items. The evaluation metrics are described as follows:

- **Discounted Cumulative Gain (DCG).** DCG [75] is widely used to evaluate the quality of rankings. In order to compute DCG, we sort the items according to the Hamming distance between their binary codes to the binary codes for the user. The DCG value of a ranking list is calculated by the following equation:

$$\text{DCG@n} = \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log(i+1)},$$

  where r_i is the rating assigned by a user to the i-th item in the ranking list. DCG mainly focuses on evaluating whether the obtained binary codes can accurately preserve the relative orders of the of items rated by each user. We use DCG@5 as a evaluation metric in our experiments. When computing DCG, we only consider the observed ratings in the test set.

In order to evaluate the performance of using binary codes for recommending top-K items to users, we apply the following evaluation metrics:

- **Precision:** For each user u, we retrieve the set of items S_u with binary codes within Hamming distance 3 to the binary codes of the user. The precision is defined as the fraction of relevant items in S_u. Formally,

$$\text{Prec} = \frac{|\{i : i \in S_u \text{ and item } i \text{ is relevant to user } u\}|}{|S_u|}.$$

In our experiments, all items with ratings that are greater than or equal to 5 are regarded as relevant items of users.

4.2.2 Data Sets

We use the three data sets, MovieLens, EachMovie and Netflix, described in Section 3.3.1 to evaluate the proposed method. We split the three data sets into training and test sets as follows: for Movielens and EachMovie, we randomly sample 80% ratings for each user as the training set and the rest 20% is used as the test set. These two data sets are very sparse and thus a lot of ratings are not observed, which may lead to biased evaluation results for precision. Therefore, we also construct a dense data set from the Netflix data as follows: We first select 5000 items with the most ratings and then sample 10000 users with at least 100 ratings to construct a relatively dense data set. For this data set, we sample 20% ratings for each users as the training set and the rest ratings are used as the test set. For all three data sets, we generate five independent splits and report the averaged performance in our evaluations. Moreover, we exclude all ratings in the training set and use only the ratings in the test set when computing the evaluation metrics.

4.2.3 Comparison of Rounding Methods

In Section 4.1.3, we describe two methods for obtaining binary codes from the approximate real-valued solutions. We now compare the performance of these methods. To this end, we denote the method that rounding to closest binary codes described in Section 4.1.3.1 as Closest and the method using orthogonal transformation described in Section 4.1.3.2 by OrthTrans. We apply CFCodeReg and CFCodePair with binary

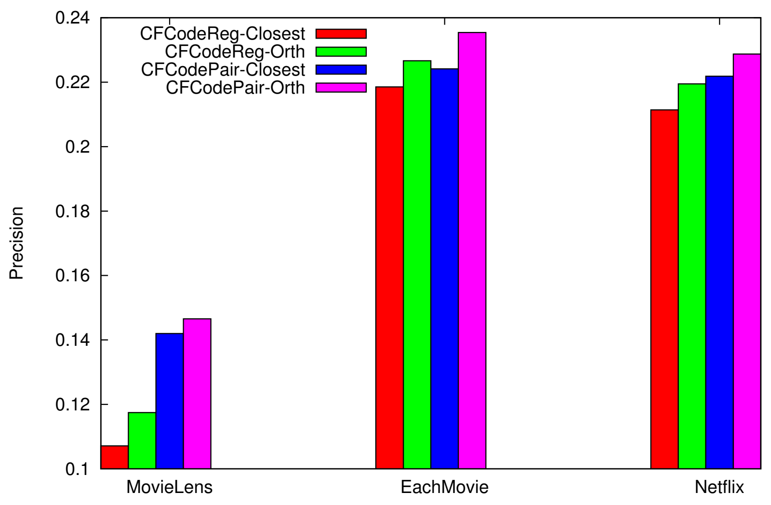


Figure 15: Performance of two rounding methods on MovieLens, EachMovie and Netflix data sets

codes of length 10 to all the three data sets and compare the binary codes obtained by Closest and OrthTrans. We report the performance on three data sets measured by precision in Figure 15. From Figure 15, we can see that the binary codes obtained by OrthTrans outperform the corresponding binary codes obtained by Closest, which suggests that the OrthTrans can obtain better approximations for binary codes. Intuitively, OrthTrans is more flexible than Closest through introducing the orthogonal transformation and thus enable us to obtain better approximation. We will use OrthTrans to obtain binary codes in the rest of our evaluations.

4.2.4 General Performance Results

4.2.4.1 Baseline Alternative Methods

We compare CFCodeReg and CFCodePair to the following baselines:

- **Spectral Hashing [161] (SH):** This method has been shown to be effective to learning binary codes. Specifically, it formulates the problem of learning binary codes as an eigenvalue problem for the similarity graph. In particular, the training ratings are viewed as the similarities between users and items and a bipartite graph is constructed between nodes representing users and items [170]. Then, spectral hashing is applied to obtain binary codes for users and items based on this graph.

- **BinMF:** In this baseline, we first apply low-rank matrix factorization to fit the training ratings and obtain low dimensional real-valued latent profiles for users and items. Then, we binarize these vectors to obtain binary codes through the orthogonal transformations described in Section 4.1.3.2 since it archives better performance as a rounding method.

4.2.4.2 Performance Analysis

We apply the proposed CFCodeReg and CFCodePair to all three data sets and compare the obtained binary codes to the two baselines described in Section 4.2.4.1. Specifically, we plot the performance measured by DCG and precision with respect to the length of the binary codes in Figure 16, Figure 17 and Figure 18 for MoveiLens, EachMovie and Netflix data sets, respectively.

We can observe that DCG and precision of both CFCodeReg and CFCodePair increase in most cases with larger length of binary codes. Hence, the performance of CFCodeReg and CFCodePair improves when the number of bits increases. Therefore, we conclude that both methods can utilize the available bits to preserve the preference of users more accurately. We can also observe from the figures that the binary codes obtained by CFCodeReg and CFCodePair outperform other baselines in terms of DCG. Thus, the proposed method can better preserve the relative orders between items according to the preference of users. Moreover, the improvement over baselines in terms of precision indicates that the binary codes obtained by CFCodeReg and CFCodePair can be used to recommend interesting items to users accurately.

Comparing the performance of CFCodeReg and CFCodePair, we can find that CF-CodePair outperforms CFCodeReg in most cases, which suggests that the pairwise loss function is more suitable for learning binary codes. This is because the pairwise loss function emphasis more on the orders between different items rather than their absolute ratings, which makes it a more reasonable loss function in the ranking scenario for collaborative filtering.

Another interesting observation is that SH does not work very well in our case. Specifically, it overfits the training data very quickly when the length of binary codes increases. By observing the results, we find that SH usually fits the training data very well. However, it frequently assign similar distances for users and items whose ratings are not in the training set. Therefore, its performance over the test set is reduced. In order to further investigate this point, we vary the length of binary codes and plot the variance of Hamming distances on unobserved ratings for binary codes generated by SH and CFCodeReg in Figure 19. We can observe that the variances produced by SH decrease when the length of binary codes grows. On the other hand, the variance generated by CFCodeReg are generally much higher than those generated by SH, which indicates that CFCodeReg generates more diverse codes when the length of binary codes increases. We think the reason is that SH usually fits the observed similarities while fails to predict the unobserved ones. This observation verifies that

CFCodeReg can not only fit the observed preferences very well, but it also can predict the unobserved preference accurately.

4.2.4.3 Impact of Parameters

We investigate the impact of the regularization parameter λ for the propose methods. To this end, we report the performance of CFCodeReg and CFCodePair measured by DCG with respect to different values of λ in Figure 20. We only show the results on the MovieLens data set due to the lack of space. From Figure 20, we can observe that the performance measured by DCG first increases and then decreases in most cases, which indicates that a good value of λ can enhance the learning process and thus improves the accuracy of learnt binary codes. In general, the value of λ can be determined by *cross validation*.

In our experiments, the relaxed optimization problem of Equation (15) is solved by LBFGS, which is an effective iterative solver for optimization problems. In Figure 21, we present the performance measured by DCG with respect to the number of iterations on MovieLens data set. We can observe from Figure 21 that the performance measured by DCG both training and test set increases when the number of iterations grows. The training process usually converges in about one hundred iterations.

4.2.4.4 Compare with Low-rank Matrix Factorizations

It is also interesting to compare CFCodeReg to the low-rank matrix factorization method that is widely exploited for collaborative filtering. Since CFCodeReg is restricted to use binary codes in order to facilitate fast search, it can be viewed as an approximation of the low-rank factorization methods. Thus, it is natural to investigate how close CFCodeReg can approximate the performance of low-rank matrix factorization. To this end, we vary to length of the binary codes from 10 to 110 and report the performance measured by DCG on MovieLens data set in Figure 22. We also report the performance of low-rank matrix factorizations when varying the rank of the factorization. We can see that the performance of CFCodeReg increases in general when the length of the binary codes grows and become very close to the performance of low-rank matrix factorizations. On the other hand, the performance of low-rank matrix factorizations is slightly reduced when the number of latent dimensions increases which is generally explained by the overfitting of the training data.

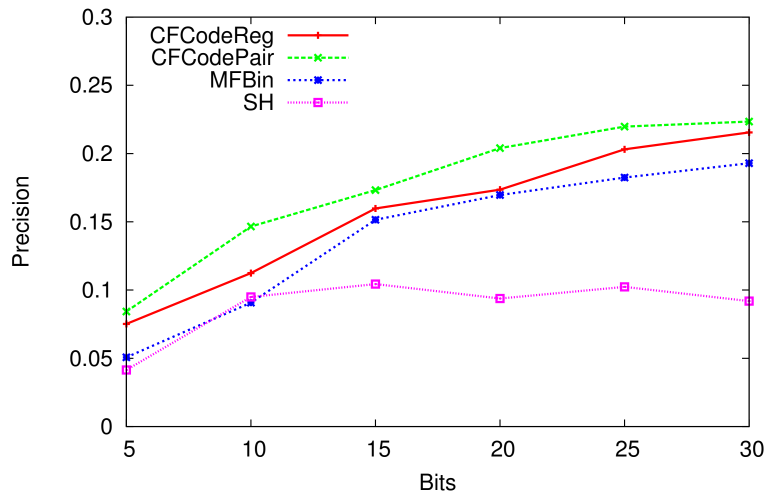
4.2.5 Recommendation Efficiency

We also compare the efficiency of obtaining top-K recommendations. In particular, for MF we compute the predicted scores for every item for a given user and then select top-10 items with the highest scores, which is linear to the number of items for each users. For CFCodeReg, we retrieve items with binary codes within Hamming distance 3 to the binary codes of the user. We measure efficiency by the total time required to generate recommendations for all users. To this end, we run the recommendation program for 10 times and report the average running time. The evaluation is conducted on a server with 16G main memory and use one of its eight 2.5GHz cores. On MovieLens data set, CFCodeReg takes 0.586 seconds to process all users while MF takes 64.9 seconds. For CFCodeReg if we further select top-10 item among retrieved items with

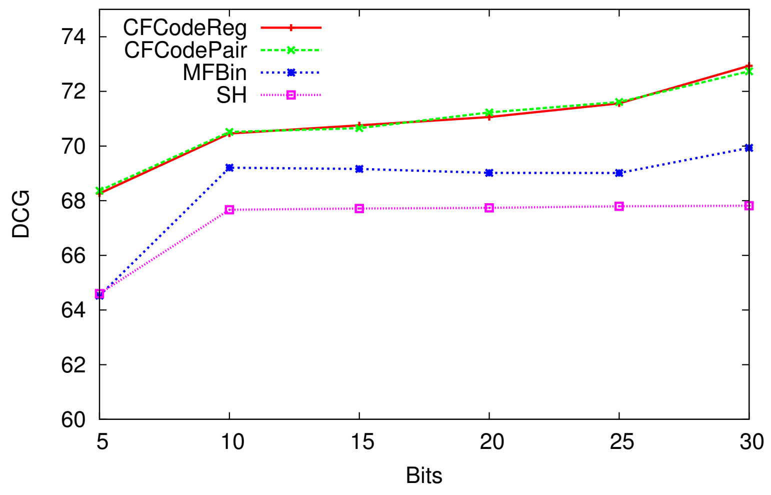
MF, the time spend is 2.92 seconds, which is still much faster than MF. The significant efficiency improvement is expected and can be explained by the fact that CFCodeReg only goes through a small fraction of items while MF computes the prediction for all items. This confirms that recommendation efficiency can be significantly improved by utilizing binary codes.

4.3 Summary

In this chapter, we address the problem of learning binary codes that preserves the preferences of users to items. In particular, we propose a framework that constructs binary codes such that the Hamming distances of a user and her preferred items are small. By applying two loss functions, the problem is formulated as a discrete optimization problem defined on the training ratings data set. It turns out that the resulting optimization problem can be solving approximately by transforming the objective function and relaxing the variables to real values. Moreover, we study two methods to obtain the binary codes from the real-valued approximations. Experiments on three data sets show that the proposed methods outperform several baselines and thus and can preserve the preference of users more accurately.

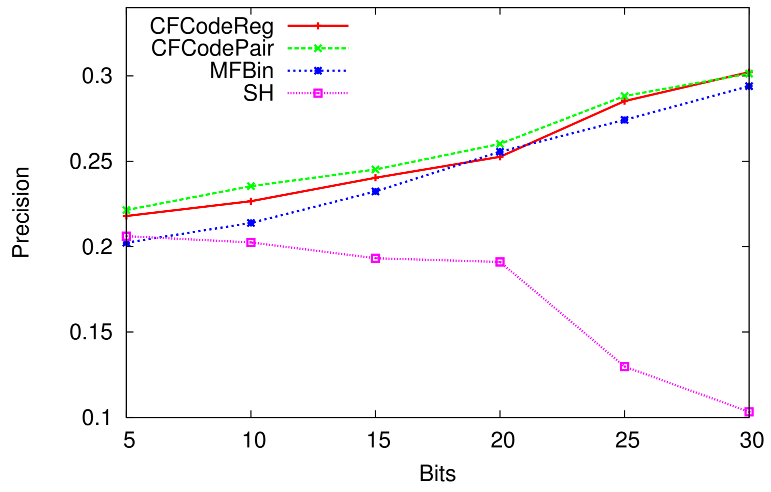


(a) Precision

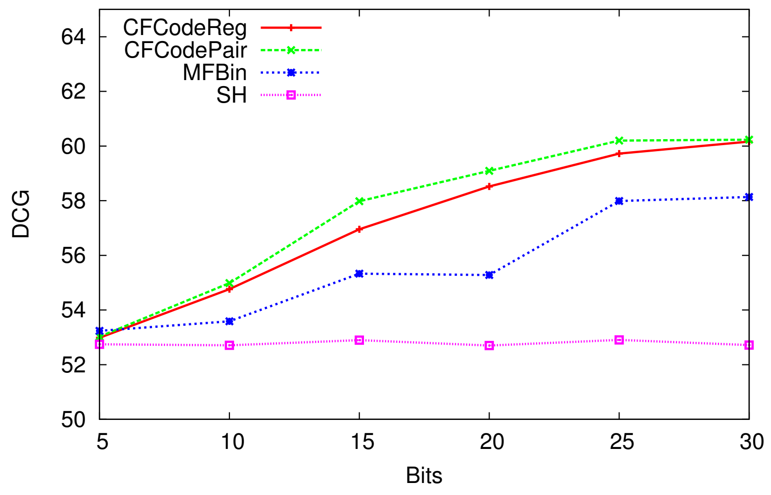


(b) DCG

Figure 16: Performance with respect to the length of binary codes on Movielens data set

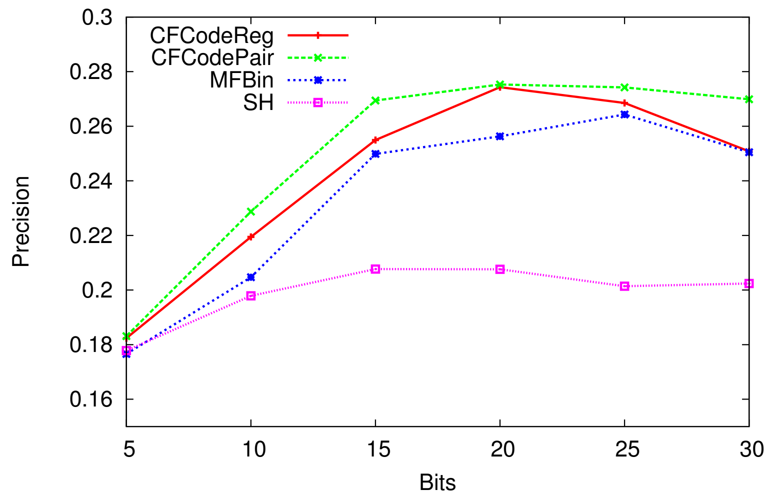


(a) Precision

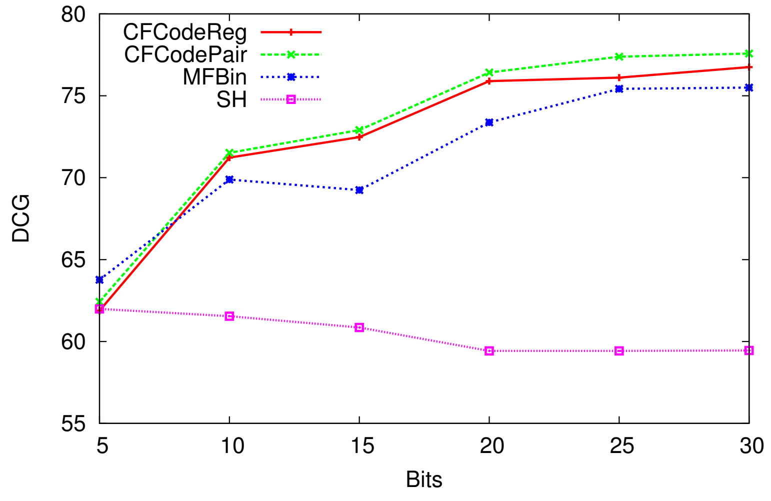


(b) DCG

Figure 17: Performance with respect to the length of binary codes on EachMoive data set



(a) Precision



(b) DCG

Figure 18: Performance with respect to the length of binary codes on Netflix data set

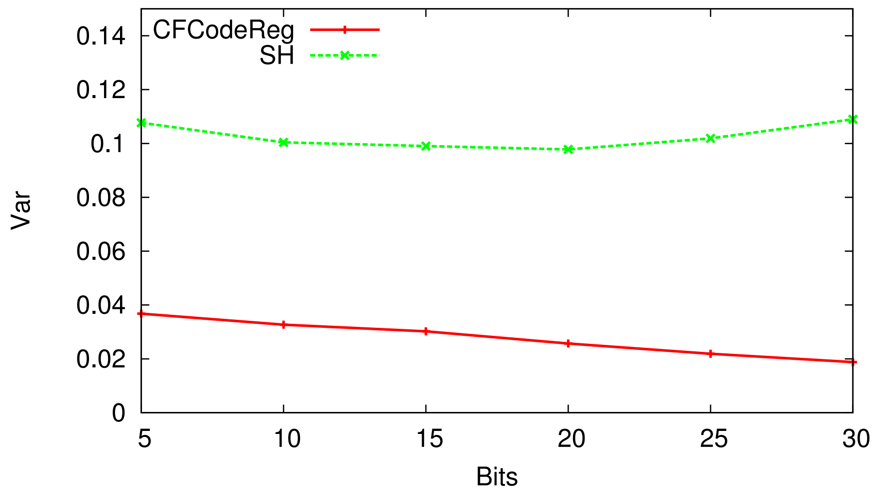


Figure 19: Variance of predicted similarity on unknown ratings with respect to the length of binary codes

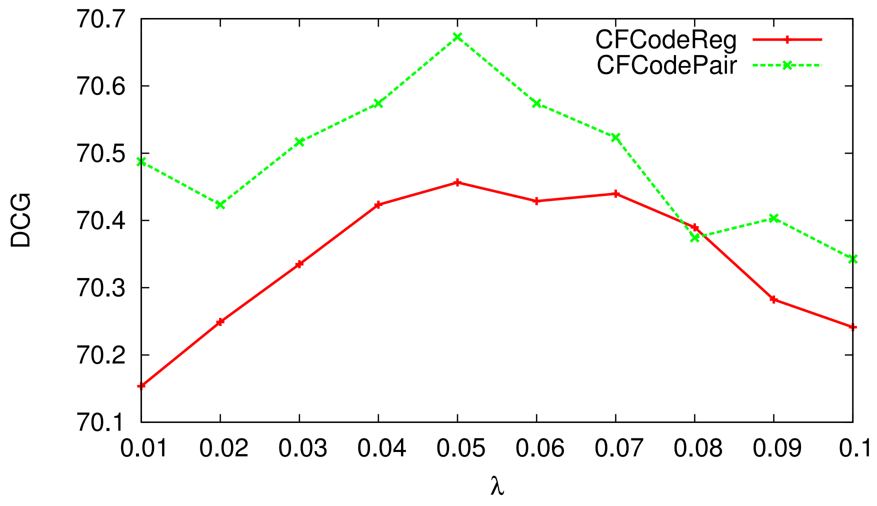
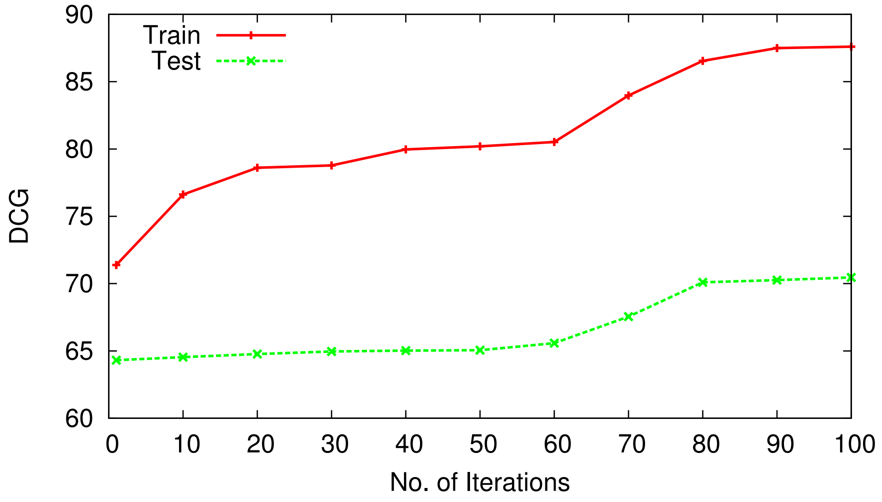


Figure 20: Performance measured by DCG with respect to the regularization parameter λ on MovieLens data set



d

Figure 21: Performance measured by DCG with respect to the number of iterations on MovieLens data set

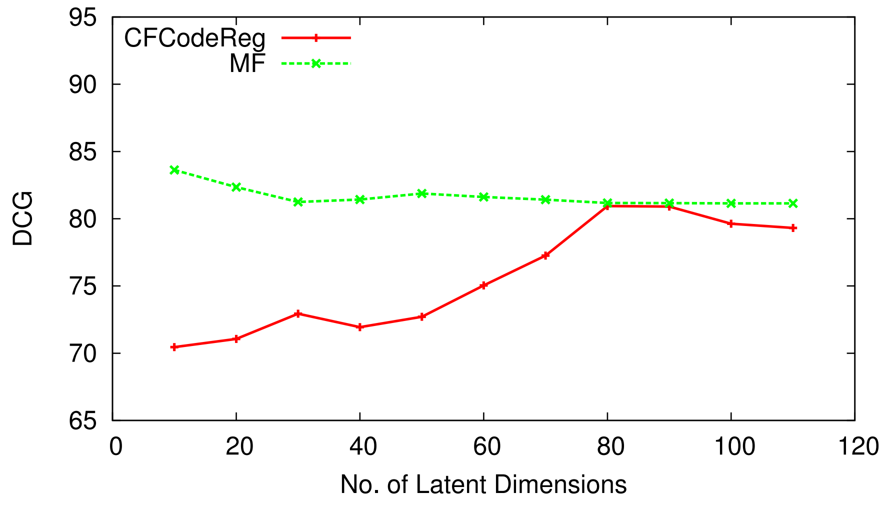


Figure 22: Comparison of low-rank matrix factorization and CF-CodeReg on MovieLens data set

CHAPTER V

LEARNING SOCIAL INFECTIVITY IN SPARSE LOW-RANK NETWORKS USING MULTI-DIMENSIONAL HAWKES PROCESSES

How will the behaviors of individuals in a social network be influenced by their neighbors, the authorities and the communities in a quantitative way? Such critical and valuable knowledge is unfortunately not readily accessible and we tend to only observe its manifestation in the form of recurrent and time-stamped events occurring at the individuals involved in the social network. It is an important yet challenging problem to infer the underlying network of social inference based on the temporal patterns of those historical events that we can observe.

In this chapter, we propose a regularized convex optimization approach to discovering the hidden network of social influence based on a multi-dimensional Hawkes process. The multi-dimensional Hawkes process captures the mutually-exciting and recurrent nature of individual behaviors, while the regularization using nuclear norm and ℓ_1 norm simultaneously on the infectivity matrix allows us to impose priors on the network topology (sparsity and low-rank structure). The advantage of our formulation is that the corresponding network discovery problem can be solved efficiently by bringing a large collection of tools developed in the optimization communities. In particular, we developed an algorithm, called ADM4, to solve the problem efficiently by combining the idea of alternating direction method of multipliers [25] and majorization minimization [64]. In our experiments on both synthetic and real world datasets, the proposed method performs significantly better than alternatives in term of accurately discovering the hidden network and predicting the response time of an individual.

5.1 *Multi-dimensional Hawkes Processes with Low-rank and Sparse Structures*

5.1.1 One-dimensional Hawkes Processes

Before introducing multi-dimensional Hawkes processes, we first describe the basic concept of counting process and one-dimensional Hawkes process [90] briefly.

Let $\{t_i\}_{i\in\mathbb{N}}$ be a one dimensional point process. Intuitively, a stochastic process $\{N_t, t \geq 0\}$ is a *counting process* if

$$N_t = \sum_{t<t_i} \mathbb{I}[t \leq t_i].$$

Formally,

Definition 1. *A random process $N_t, t \geq 0$ is a counting process if*

- $N_t \in \mathbb{N}$
- $N_s \leq N_t, \quad \forall s < t.$
- $dN_t = N_t - N(t^-) \in \{0, 1\}$[1]
- $\mathbb{E}N_t$ *is well-defined.*

[1] *We use the notation dN_s without a formal mathematical definition of the random measures, since we are focused on the modeling perspective. Intuitively, it can be viewed as the number of events in time interval $[s, s + ds]$. The readers are referred to [90] for formal definitions.*

The process $\{\delta t_i\}_{i\in\mathbb{N}}$ is the *duration process*, where $\delta t_i = t_i - t_{i-1}$.

Conditional intensity function. Let $\mathcal{H}_t$ represents the history of the random process up to time t. Let $f^*(t) = f(t|\mathcal{H}_t)$ be the conditional density function of the time of the next event t_{n+1} given the history of previous events $(\dots, t_{n-1}, t_n)$.

Then, the joint density

$$f(t_1, t_2, \dots) = \prod_i f(t_i| \dots, t_{i_1}, t_i) = \prod_i f^*(t_i)$$

Let $F^*(t) = \int_{-\infty}^{t} f^*(s)\mathrm{d}s$ be the cumulative distribution function of f^*. Then the conditional density function (or hazard function) is defined by:

$$\lambda(t) = \frac{f^*(t)}{1 - F^*(t)} \tag{20}$$

The conditional intensity function can be interpreted in the following way:

$$\begin{aligned}
\lambda(t) &= \lim_{h\to 0^+} \frac{1}{h} \frac{\Pr(N_{t+h} - N_t > 0|\mathcal{H}_t)}{\Pr(N_t - N_{t_{n-1}} = 0|\mathcal{H}_t)} \\
&= \lim_{h\to 0^+} \frac{1}{h} \frac{\Pr(N_{t+h} - N_t > 0, N_t - N_{t_n} = 0|\mathcal{H}_t)}{\Pr(N_t - N_{t_n} = 0|\mathcal{H}_t)} \\
&= \lim_{h\to 0^+} \frac{1}{h} \Pr[N_{t+h} - N_t > 0|\mathcal{H}_t] \\
&= \lim_{h\to 0^+} \mathbb{E}\left[\frac{N_{t+h} - N_t}{h}|\mathcal{H}_t\right]
\end{aligned}$$

We have the following theorem that establishes the relationship between conditional intensity and density functions of a counting process:

Theorem 3. *The reverse relation of Equation (20) is given by:*

$$\begin{aligned}
f^*(t) &= \lambda(t) \exp\left(-\int_{t_n}^{t} \lambda(s)\mathrm{d}s\right) \\
F^*(t) &= 1 - \exp\left(-\int_{t_n}^{t} \lambda(s)\mathrm{d}s\right)
\end{aligned}$$

Proof. From Equation (20), we have

$$\lambda(t)dt = \frac{dF^*(t)}{1 - F^*(t)}$$

Integrating over t, we have

$$\int_{t_n}^{t} \lambda(s)ds = \int_{t_n}^{t} \frac{dF^*(s)}{1 - F(s)} ds = -\log(1 - F^*(t))$$

Therefore,

$$F^*(t) = 1 - \exp(-\int_{t_n}^{t} \lambda(s)ds.$$

Substitute the above equation into Equation (20), we have

$$f^*(t) = \lambda(t) \exp\left(-\int_{t_n}^{t} \lambda(s)\mathrm{d}s\right)$$

□

Given the conditional intensity function, the likelihood function can be computed according to the following theorem:

Theorem 4. *Given point pattern* $(t_1, \ldots, t_n)$ *on an observation interval* $[0, T)$*, the likelihood function is given by*

$$L = \left(\prod_{i=1}^{n} \lambda(t_i)\right) \exp\left(-\int_{0}^{T} \lambda(s)\mathrm{d}s\right)$$

Proof.

$$\begin{aligned}
L &= f^*(t_1) \ldots f^*(t_n)(1 - F^*(T)) \\
&= \left(\prod_{i=1}^{n} f^*(t_i)\right) \frac{f^*(T)}{\lambda(t)} \\
&= \left(\prod_{i=1}^{n} \lambda(t_i) \exp\left(-\int_{t_{i-1}}^{t_i} \lambda(s)\mathrm{d}s\right)\right) \exp\left(-\int_{t_n}^{T} \lambda(s)\mathrm{d}s\right) \\
&= \left(\prod_{i=1}^{n} \lambda(t_i)\right) \exp\left(-\int_{0}^{T} \lambda(s)\mathrm{d}s\right)
\end{aligned}$$

□

Now, we can describe one-dimensional Hawkes process. In its most basic form, a one-dimensional Hawkes process is a point process N_t with its conditional intensity expressed as follows [59]

$$\lambda(t) = \mu + a \int_{-\infty}^{t} g(t-s) dN_s = \mu + a \sum_{i:t_i<t} g(t-t_i),$$

where $\mu > 0$ is the base intensity, t_i are the time of events in the point process before time t, and $g(t)$ is the decay kernel. We focus on the case of exponential kernel $g(t) = we^{-wt}$ as a concrete examples in this chapter, but the framework discussed in this chapter can be easily adapted to other positive kernels. In the above conditional intensity function, the sum over i with $t_i < t$ captures the self-exciting nature of the point process: the occurrence of events in the past has a positive contribution of the event intensity in the future. Given a sequence of events $\{t_i\}_{i=1}^{n}$ observed in the time interval $[0, T]$ that is generated from the above conditional intensity, the log-likelihood function can be expressed as follows according to Theorem 4:

$$\mathcal{L} = \log \frac{\prod_{i=1}^{n} \lambda(t_i)}{\exp(\int_0^T \lambda(t)dt)} = \sum_{i=1}^{n} \log \lambda(t_i) - \int_0^T \lambda(t)dt.$$

5.1.2 Multi-dimensional Hawkes Processes

In order to model social influence, one-dimensional Hawkes process discussed above needs to be extended to the multi-dimensional case [90]. In this case, we have U Hawkes processes that are coupled with each other: each of the Hawkes processes corresponds to an individual and the influence between individuals are explicitly modeled. Formally, the multi-dimensional Hawkes process is defined by a U-dimensional point process $N_t^u, u = 1, \dots, U$, with the conditional intensity for the u-th dimension expressed as follows:

$$\lambda_u(t) = \mu_u + \sum_{i:t_i<t} a_{uu_i} g(t - t_i),$$

where $\mu_u \geq 0$ is the base intensity for the u-th Hawkes process. The coefficient $a_{uu'} \geq 0$ captures the mutually-exciting property between the u-th and u'-th dimension. Intuitively, it captures the degree of influence of events occurred in the u'-th dimension to the u-th dimension. Larger value of $a_{uu'}$ indicates that events in u'-th dimension are more likely to trigger a event in the u-th dimension in the future. We collect the parameters into matrix-vector forms, $\boldsymbol{\mu} = (\mu_u)$ for the base intensity, and $\mathbf{A} = (a_{uu'})$ for the mutually-exciting coefficients, called infectivity matrix. We use $\mathbf{A} \geq 0$ and $\boldsymbol{\mu} \geq 0$ to indicate that we require both matrices to be entry-wise nonnegative.

Suppose we have m samples, $\{c_1, \dots, c_m\}$, from the multi-dimensional Hawkes process. Each sample c is a sequence of events observed during a time period of $[0, T_c]$, which is in the form of $\{(t_i^c, u_i^c)\}_{i=1}^{n_c}$. Each pair (t_i^c, u_i^c) represents an event occurring at the u_i^c-th dimension at time t_i^c. Thus, the log-likelihood of model parameters $\Theta = \{\mathbf{A}, \boldsymbol{\mu}\}$ can be expressed as follows

$$\begin{aligned}\mathcal{L}(\mathbf{A}, \boldsymbol{\mu}) =& \sum_c \left(\sum_{i=1}^{n_c} \log \lambda_{u_i^c}(t_i^c) - \sum_{u=1}^{U} \int_0^{T_c} \lambda_u(t) dt \right) \\ =& \sum_c \left(\sum_{i=1}^{n_c} \log \left(\mu_{u_i^c} + \sum_{t_j^c < t_i^c} a_{u_i^c u_j^c} g(t_i^c - t_j^c) \right) \right. \\ & \left. - T_c \sum_{u=1}^{U} \mu_u - \sum_{u=1}^{U} \sum_{j=1}^{n_c} a_{uu_j^c} G(T_c - t_j^c) \right), \end{aligned} \tag{21}$$

where $G(t) = \int_0^t g(s) ds$. In general, the parameters $\mathbf{A}$ and $\boldsymbol{\mu}$ can be estimated by maximizing the log-likelihood, *i.e.*, $\min_{\mathbf{A} \geq 0, \boldsymbol{\mu} \geq 0} \; -\mathcal{L}(\mathbf{A}, \boldsymbol{\mu})$.

5.1.3 Sparse and Low-Rank Regularization

As we mentioned earlier, we would like to take into account the structure of the social influence in the proposed model. We focus on two important properties of the social influences: sparsity and low-rank. The sparsity of social influences implies that most individuals only influence a small fraction of users in the network while there can be a few hubs with wide-spread influence. This can be reflected in the sparsity pattern of $\mathbf{A}$. Furthermore, the communities structure in the influence network implies low-rank structures, which can also be reflected in matrix $\mathbf{A}$. Thus, we consider incorporating these prior knowledge by imposing both low-rank and sparse regularization on $\mathbf{A}$. That is we regularize our maximum likelihood estimator with

$$\min_{\mathbf{A}\geq 0, \boldsymbol{\mu}\geq 0} -\mathcal{L}(\mathbf{A}, \boldsymbol{\mu}) + \lambda_1\|\mathbf{A}\|_* + \lambda_2\|\mathbf{A}\|_1, \tag{22}$$

where $\|\mathbf{A}\|_*$ is the nuclear norm of matrix $\mathbf{A}$, which is defined to be the sum of its singular value $\sum_{i=1}^{\text{rank}\mathbf{A}} \sigma_i$. The nuclear norm has been used to estimate low-rank matrices effectively [143]. Moreover, $\|\mathbf{A}\|_1 = \sum_{u,u'} |a_{uu'}|$ is the ℓ_1 norm of the matrix $\mathbf{A}$, which is used to enforce the sparsity of the matrix $\mathbf{A}$. The parameter λ_1 and λ_2 control the strength of the two regularization terms.

5.2 *Efficient Optimization*

It can be observed that the objective function in Equation (22) is non-differentiable and thus difficult to optimize in general. We apply the idea of alternating direction method of multipliers (ADMM) [49] to convert the optimization problem to several sub-problems that are easier to solve. The ADMM has been shown to be a special case of the more general Douglas-Rachford splitting method, which has good convergence properties under some fairly mild conditions [43] compared with other alternatives such proximal gradient methods [32].

Specifically, we first rewrite the optimization problem in Equation (22) to an equivalent form by introducing two auxiliary variables $\mathbf{Z_1}$ and $\mathbf{Z_1}$

$$\min_{\mathbf{A}\geq 0,\boldsymbol{\mu}\geq 0,\mathbf{Z_1},\mathbf{Z_2}} -\mathcal{L}(\mathbf{A},\boldsymbol{\mu}) + \lambda_1\|\mathbf{Z_1}\|_* + \lambda_2\|\mathbf{Z_2}\|_1, \tag{23}$$
$$\text{s.t. } \mathbf{A} = \mathbf{Z_1}, \quad \mathbf{A} = \mathbf{Z_2}.$$

In ADMM, we optimize the augmented Lagrangian of the above problem that can be expressed as follows:

$$\begin{aligned}\mathcal{L}_\rho =& -\mathcal{L}(\boldsymbol{\mu},\mathbf{A}) + \lambda_1\|\mathbf{Z}_1\|_* + \lambda_2\|\mathbf{Z}_2\|_1\\ &+\rho\text{tr}(\mathbf{U}_1^T(\mathbf{A}-\mathbf{Z}_1)) + \rho\text{tr}(\mathbf{U}_2^T(\mathbf{A}-\mathbf{Z}_2))\\ &+\frac{\rho}{2}(\|\mathbf{A}-\mathbf{Z}_1\|^2 + \|\mathbf{A}-\mathbf{Z}_2\|^2),\end{aligned}$$

where $\rho > 0$ is called the penalty parameter and $\|\cdot\|$ denotes the Frobenius norm. The matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ are the dual variable associated with the constraints $\mathbf{A} = \mathbf{Z}_1$ and $\mathbf{A} = \mathbf{Z}_2$, respectively.

In ADMM, we consider the *augmented Lagrangian* of the above constrained optimization problem by writing it as follows:

$$\min -\mathcal{L}(\boldsymbol{\mu},\mathbf{A}) + \lambda_1\|\mathbf{Z}_1\|_* + \lambda_2\|\mathbf{Z}_2\|_1 + \frac{\rho}{2}(\|\mathbf{A}-\mathbf{Z}_1\|^2 + \|\mathbf{A}-\mathbf{Z}_2\|^2) \tag{24}$$

subject to

$$\mathbf{A} = \mathbf{Z}_1, \mathbf{A} = \mathbf{Z}_2.$$

Clearly, for all ρ the optimization problem defined above is equivalent to Equation (23) and thus equivalent to the problem defined in Equation (22). The augmented Lagrangian of Equation (23) is the (standard) Lagrangian of Equation (24), which can be expressed as follows:

$$\begin{aligned}\mathcal{L}_\rho = &-\mathcal{L}(\boldsymbol{\mu}, \mathbf{A}) + \lambda_1\|\mathbf{Z}_1\|_* + \lambda_2\|\mathbf{Z}_2\|_1\\ &+ \mathrm{tr}(\mathbf{Y}_1^T(\mathbf{A}-\mathbf{Z}_1)) + \mathrm{tr}(\mathbf{Y}_2^T(\mathbf{A}-\mathbf{Z}_2))\\ &+ \frac{\rho}{2}(\|\mathbf{A}-\mathbf{Z}_1\|^2 + \|\mathbf{A}-\mathbf{Z}_2\|^2)\end{aligned}$$

Thus, we can solve the optimization problem defined in Equation (23) applying the gradient ascent algorithm to the dual variables $\mathbf{Y}_1$ and $\mathbf{Y}_2$. It can be shown that the update of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ has the following form at the k-th iteration:

$$\begin{aligned}\mathbf{Y}_1^{k+1} &= \mathbf{Y}_1^k + \rho(\mathbf{A}^{k+1} - \mathbf{Z}_1^{k+1})\\ \mathbf{Y}_2^{k+1} &= \mathbf{Y}_2^k + \rho(\mathbf{A}^{k+1} - \mathbf{Z}_2^{k+1}),\end{aligned}$$

where $\mathbf{A}^{k+1}$, $\mathbf{Z}_1^{k+1}$ and $\mathbf{Z}_2^{k+1}$ are obtained by optimizing $\mathcal{L}_\rho$ with $\mathbf{Y}_1 = \mathbf{Y}_1^k$ and $\mathbf{Y}_2 = \mathbf{Y}_2^k$ fixed:

$$\underset{\mathbf{A},\boldsymbol{\mu},\mathbf{Z}_1,\mathbf{Z}_2}{\mathrm{argmin}}\ \mathcal{L}_\rho(\mathbf{A}, \boldsymbol{\mu}, \mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Y}_1^k, \mathbf{Y}_2^k)$$

In ADMM, the above problem is solved by updating $\mathbf{A}$, $\boldsymbol{\mu}$, $\mathbf{Z}_1$ and $\mathbf{Z}_2$ sequentially as follows:

$$\begin{aligned}\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1} &= \underset{\mathbf{A},\boldsymbol{\mu},}{\mathrm{argmin}}\,\mathcal{L}_\rho(\mathbf{A}, \boldsymbol{\mu}, \mathbf{Z}_1^k, \mathbf{Z}_2^k, \mathbf{Y}_1^k, \mathbf{Y}_2^k),\\ \mathbf{Z}_1^{k+1} &= \underset{\mathbf{Z}_1}{\mathrm{argmin}}\,\mathcal{L}_\rho(\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1}, \mathbf{Z}_1, \mathbf{Z}_2^k, \mathbf{Y}_1^k, \mathbf{Y}_2^k),\\ \mathbf{Z}_2^{k+1} &= \underset{\mathbf{Z}_2}{\mathrm{argmin}}\,\mathcal{L}_\rho(\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1}, \mathbf{Z}_1^{k+1}, \mathbf{Z}_2, \mathbf{Y}_1^k, \mathbf{Y}_2^k).\end{aligned}$$

It is usually more convenient to consider the scaled form of ADMM. Let $\mathbf{U}_1^k = \mathbf{Y}_1^k/\rho$ and $\mathbf{U}_2^k = \mathbf{Y}_2^k/\rho$, we obtain the following iterative steps:

$$\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1} = \underset{\mathbf{A}\geq 0, \boldsymbol{\mu}\geq 0}{\operatorname{argmin}} \mathcal{L}_\rho(\mathbf{A}, \boldsymbol{\mu}, \mathbf{Z}_1^k, \mathbf{Z}_2^k, \mathbf{U}_1^k, \mathbf{U}_2^k), \tag{25}$$

$$\mathbf{Z}_1^{k+1} = \underset{\mathbf{Z}_1}{\operatorname{argmin}} \mathcal{L}_\rho(\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1}, \mathbf{Z}_1, \mathbf{Z}_2^k, \mathbf{U}_1^k, \mathbf{U}_2^k), \tag{26}$$

$$\mathbf{Z}_2^{k+1} = \underset{\mathbf{Z}_2}{\operatorname{argmin}} \mathcal{L}_\rho(\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1}, \mathbf{Z}_1^{k+1}, \mathbf{Z}_2, \mathbf{U}_1^k, \mathbf{U}_2^k), \tag{27}$$

$$\mathbf{U}_1^{k+1} = \mathbf{U}_1^k + (\mathbf{A}^{k+1} - \mathbf{Z}_1^{k+1}),$$

$$\mathbf{U}_2^{k+1} = \mathbf{U}_2^k + (\mathbf{A}^{k+1} - \mathbf{Z}_2^{k+1}).$$

The advantage of sequential update is that we separate multiple variables and thus can optimize them one at a time. We first consider the optimization problem for $\mathbf{Z}_1$ and $\mathbf{Z}_2$ and then describe the algorithm used to optimize with respect to $\mathbf{A}$ and $\boldsymbol{\mu}$.

5.2.1 Solving for $\mathbf{Z}_1$ and $\mathbf{Z}_2$.

When solving for $\mathbf{Z}_1$ in Equation (26), the relevant terms from $\mathcal{L}_\rho$ are

$$\lambda_1\|\mathbf{Z}_1\|_* + \rho\operatorname{tr}((U_1^k)^T(\mathbf{A}^{k+1} - \mathbf{Z}_1)) + \frac{\rho}{2}\|\mathbf{A}^{k+1} - \mathbf{Z}_1\|^2,$$

which can be simplified to an equivalent problem,

$$\mathbf{Z}_1^{k+1} = \underset{\mathbf{Z}_1}{\operatorname{argmin}}\ \lambda_1\|\mathbf{Z}_1\|_* + \frac{\rho}{2}\|\mathbf{A}^{k+1} - \mathbf{Z}_1 + \mathbf{U}_1^k\|^2.$$

The above problem has a closed form solution [143]

$$\mathbf{Z}_1^{k+1} = \mathcal{S}_{\lambda_1/\rho}(\mathbf{A}^{k+1} + \mathbf{U}_1^k), \tag{28}$$

where $\mathcal{S}_\alpha(\mathbf{X})$ is a soft-thresholding function defined as $\mathcal{S}_\alpha(\mathbf{X}) = \mathbf{U}\operatorname{diag}((\sigma_i - \alpha)_+)\mathbf{V}^T$ for all matrix $\mathbf{X}$ with singular value decomposition $\mathbf{X} = \mathbf{U}\operatorname{diag}(\sigma_i)\mathbf{V}^T$.

Similarly, the optimization for $\mathbf{Z}_2$ can be simplified into the following equivalent form

$$\mathbf{Z}_2^{k+1} = \underset{\mathbf{Z}_2}{\operatorname{argmin}} \ \lambda_2 \|\mathbf{Z}_2\|_1 + \frac{\rho}{2} \|\mathbf{A}^{k+1} - \mathbf{Z}_2 + \mathbf{U}_2^k\|^2,$$

which again has the closed form solution. In this case, depending on the magnitude of the ij-th entry of the matrix $(\mathbf{A}^{k+1} + \mathbf{U}_2^k)$, the corresponding $(\mathbf{Z}_2^{k+1})_{ij}$ is updated as [48]

$$\begin{cases} (\mathbf{A}^{k+1} + \mathbf{U}_2^k)_{ij} - \frac{\lambda_2}{\rho}, & (\mathbf{A}^{k+1} + \mathbf{U}_2^k)_{ij} \geq \frac{\lambda_2}{\rho}, \\ (\mathbf{A}^{k+1} + \mathbf{U}_2^k)_{ij} + \frac{\lambda_2}{\rho}, & (\mathbf{A}^{k+1} + \mathbf{U}_2^k)_{ij} \leq -\frac{\lambda_2}{\rho}, \\ 0, & |(\mathbf{A}^{k+1} + \mathbf{U}_2^k)_{ij}| < \frac{\lambda_2}{\rho}. \end{cases} \tag{29}$$

5.2.2 Solving for $\mathbf{A}$ and $\boldsymbol{\mu}$

The optimization problem for $\mathbf{A}$ and $\boldsymbol{\mu}$ defined in Equation (25) can be equivalently written as

$$\mathbf{A}^{k+1}, \boldsymbol{\mu}^{k+1} = \underset{\mathbf{A} \geq 0, \boldsymbol{\mu} \geq 0}{\operatorname{argmin}} \ f(\mathbf{A}, \boldsymbol{\mu})$$

where $f(\mathbf{A}, \boldsymbol{\mu}) = -\mathcal{L}(\mathbf{A}, \boldsymbol{\mu}) + \frac{\rho}{2}(\|\mathbf{A} - \mathbf{Z}_1^k + \mathbf{U}_1^k\|^2 + \|\mathbf{A} - \mathbf{Z}_2^k + \mathbf{U}_2^k\|^2)$. We propose to solve the above problem by a majorization-minimization algorithm [64] which is a generalization of the EM algorithm. Since the optimization is convex, we still obtain global optimum for this subproblem. Specifically, given any estimation $\mathbf{A}^{(k)}$ and $\boldsymbol{\mu}^{(k)}$ of $\mathbf{A}$ and $\boldsymbol{\mu}$, we minimize a surrogate function $Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$ which is a tight upper bound of $f(\mathbf{A}, \boldsymbol{\mu})$. Indeed, $Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$ can be defined as follows:

$$\begin{aligned}
& Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) \\
= & -\sum_c \left(\sum_{i=1}^{n^c} \left(p_{ii}^c \log \frac{\mu_{u_i^c}}{p_{ii}^c} + \sum_{j=1}^{i-1} p_{ij}^c \log \frac{a_{u_i^c u_j^c} g(t_i^c - t_j)}{p_{ij}^c} \right) \right. \\
& \left. - \left(T_c \sum_u \mu_u + \sum_{u=1}^{U} \sum_{j=1}^{n^c} a_{u u_j^c} G(T - t_j^c) \right) \right) \\
& + \frac{\rho}{2} (\|\mathbf{A} - \mathbf{Z}_1^k + \mathbf{U}_1^k\|^2 + \|\mathbf{A} - \mathbf{Z}_2^k + \mathbf{U}_2^k\|^2), \qquad (30)
\end{aligned}$$

where

$$p_{ii}^c = \frac{\mu_{u_i^c}^{(k)}}{\mu_{u_i^c}^{(k)} + \sum_{j=1}^{i-1} a_{u_i^c u_j^c}^{(k)} g(t_i^c - t_j^c)},$$
$$p_{ij}^c = \frac{a_{u_i^c u_j^c}^{(k)} g(t_i^c - t_j^c)}{\mu_{u_i^c}^{(k)} + \sum_{j=1}^{i-1} a_{u_i^c u_j^c}^{(k)} g(t_i^c - t_j^c)}.$$

Intuitively, p_{ij}^c can be interpreted as the probability that the i-th event is influenced by a previous event j in the network and p_{ii}^c is the probability that i-th event is sampled from the base intensity. Thus, the first two terms of $Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$ can be viewed as the joint probability of the unknown infectivity structures and the observed events.

An important property is that optimizing $Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$ ensures that $f(\mathbf{A}, \boldsymbol{\mu})$ is decreasing monotonically.

First, we claim that the following properties hold for $Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)};)$ defined in Equation (30) :

- **1.** For all $\mathbf{A}$, $\boldsymbol{\mu}$,

$$Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) \geq f(\mathbf{A}, \boldsymbol{\mu})$$

- **2.**

$$Q(\mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) = f(\mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$$

Proof. The first claim can be shown by utilizing the Jensen's inequality: For all c and i, we have

$$\log(\mu_{u_i^c} + \sum_{j=1}^{i-1} a_{u_i^c u_j^c} g(t_i^c - t_j^c)) \geq p_{ii}^c \log \frac{u_i^c}{p_{ii}^c} + \sum_{j=1}^{i-1} p_{ij}^c \frac{a_{u_i^c u_j^c} g(t_i^c - t_j^c))}{p_{ij}^c}$$

Summing up over c and i proves the claim.

The second claim can be checked by setting $\mathbf{A} = \mathbf{A}^{(m)}$ and $\boldsymbol{\mu} = \boldsymbol{\mu}^{(m)}$. □

The above two properties imply that if $(\mathbf{A}^{(m+1)}, \boldsymbol{\mu}^{(m+1)}) = \operatorname{argmin}_{\mathbf{A},\boldsymbol{\mu}} Q(\mathbf{A}, \boldsymbol{\mu}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)})$, we have

$$\begin{aligned} f(\mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) &= Q(\mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) \\ &\geq Q(\mathbf{A}^{(m+1)}, \boldsymbol{\mu}^{(m+1)}; \mathbf{A}^{(k)}, \boldsymbol{\mu}^{(k)}) \\ &\geq f(\mathbf{A}^{(m+1)}, \boldsymbol{\mu}^{(m+1)}). \end{aligned}$$

Thus, optimizing Q with respect to $\mathbf{A}$ and $\boldsymbol{\mu}$ ensures that the value of $f(\mathbf{A}, \boldsymbol{\mu})$ decrease monotonically.

Moreover, the advantage of optimizing $Q(\mathbf{A}, \boldsymbol{\mu})$ is that all parameters $\mathbf{A}$ and $\boldsymbol{\mu}$ can be solved independently with each other with closed forms solutions, and the nonnegativity constraints are automatically taken care of. That is

$$\mu_u^{(m+1)} = \frac{\sum_c \sum_{i:i\leq n^c, u_i^c = u} p_{ii}^c}{\sum_c T_c} \tag{31}$$

$$a_{uu'}^{(m+1)} = \frac{-B + \sqrt{B^2 + 8\rho C}}{4\rho}, \tag{32}$$

Algorithm 3 ADMM-MM (ADM4) for estimating $\mathbf{A}$ and $\boldsymbol{\mu}$

```
Input: Observed samples $\{c_1, \ldots, c_m\}$.
Output: $\mathbf{A}$ and $\boldsymbol{\mu}$.
  Initialize $\boldsymbol{\mu}$ and $\mathbf{A}$ randomly; Set $\mathbf{U}_1 = 0$, $\mathbf{U}_2 = 0$.
  while $k = 1, 2, \ldots, K$ do
    Update $\mathbf{A}^{k+1}$ and $\boldsymbol{\mu}^{k+1}$ by optimizing $Q$ defined in (30) as follows:
    while not converge do
      Update $\mathbf{A}$, $\boldsymbol{\mu}$ using (32) and (31) respectively.
    end while
    Update $\mathbf{Z}_1^{k+1}$ using (28); Update $\mathbf{Z}_2^{k+1}$ using (29).
    Update $\mathbf{U}_1^{k+1} = \mathbf{U}_1^k + (\mathbf{A}^{k+1} - \mathbf{Z}_1^{k+1})$ and $\mathbf{U}_2^{k+1} = \mathbf{U}_2^k + (\mathbf{A}^{k+1} - \mathbf{Z}_2^{k+1})$.
  end while
  return $\mathbf{A}$ and $\boldsymbol{\mu}$.
```

where

$$B = \sum_c \sum_{j: u_j^c = u'} G(T - t_j^c) + \rho(-z_{1,uu'} + u_{1,uu'} - z_{2,uu'} + u_{2,uu'}),$$

$$C = \sum_c \sum_{i=1, u_i^c = u}^{n^c} \sum_{j<i, u_j^c = u'} p_{ij}^c.$$

The overall optimization algorithm is summarized in Algorithm 3.

5.3 Experiments

In this section, we conducted experiments on both synthetic and real-world datasets to evaluate the performance of the proposed method.

5.3.1 Synthetic Data

Data Generation. The goal is to show that our proposed algorithm can reconstruct the underlying parameters from observed recurrent events. To this end, we consider a U-dimensional Hawkes process with $U = 1000$ and generate the true parameters $\boldsymbol{\mu}$ from a uniform distribution on $[0, 0.001]$. In particular, the infectivity matrix $\mathbf{A}$ is generated by $\mathbf{A} = \mathbf{U}\mathbf{V}^T$. We consider two different types of influences in our experiments: assortative mixing and disassortative mixing:

- In the assortative mixing case, $\mathbf{U}$ and $\mathbf{V}$ are both 1000×9 matrices with entries in $[100(i-1)+1 : 100(i+1), i], i = 1, \ldots, 9$ sampled randomly from $[0, 0.1]$ and all other entries are set to zero. Assortative mixing examples capture the scenario that influence are mostly coming from members of the same group.

- In the disassortative mixing case, $\mathbf{U}$ is generated in the same way as the assortative mixing case, while the $\mathbf{V}$ has non-zero entries in $[100(i-1)+1 : 100(i+1), 10-i], i = 1, \ldots, 9$. Disassortative mixing examples capture the scenario that influence can come outside of the group, possibly from a few influential hubs.

We scale $\mathbf{A}$ so that the spectral radius of $\mathbf{A}$ is 0.8 to ensure the point process is well-defined, i.e., with finite intensity. Then, 50000 samples are sampled from the multi-dimensional Hawkes process specified by $\mathbf{A}$ and $\boldsymbol{\mu}$. The proposed algorithms are applied to the samples to obtain estimations $\hat{\mathbf{A}}$ and $\hat{\boldsymbol{\mu}}$.

Evaluation Metric. We use three evaluation metrics to measure the performance:

- RelErr is defined as the averaged relative error between the estimated parameters and the true parameters, *i.e.* $\frac{|a_{ij} - \hat{a}_{ij}|}{|a_{ij}|}$ for $a_{ij} \neq 0$ and $|a_{ij} - \hat{a}_{ij}|$ for $a_{ij} = 0$.

- PredLik is defined as the log-likelihood of the estimated model on a separate held-out test set containing 50,000 samples.

- RankCorr is defined as the averaged Kendall's rank correlation coefficient between each row of $\mathbf{A}$ and $\hat{\mathbf{A}}$. It measures whether the relative order of the estimated social influences is correctly recovered.

Results. We included 5 methods in the comparisons:

- TimeWindow. This method, we first discretize time into time windows with equal length and then represent each node by a vector where each dimension is to the number of events occurred within the corresponding time window at the node. The cosine similarity are used to estimate the infectivity matrix.
- NetRate. This method is proposed in [129] for modeling information diffusion in networks. It can not model the recurrent events, so we only keep the first occurrences at each node in the training data.
- Full. The infectivity matrix $\mathbf{A}$ is estimated as a general $U \times U$ matrix without any structure.
- LowRank. Only the nuclear norm is used to obtain a low-rank estimation of $\mathbf{A}$.
- Sparse. Only the ℓ_1 norm is used to obtain a sparse estimation of $\mathbf{A}$.
- LowRankSparse. This is the proposed method ALGORITHM 1. Both nuclear norm and ℓ_1 norm are used to estimate $\mathbf{A}$.

For each method, the parameters are selected on a validation set that are disjoint from both training and test set. We run each experiment for five times with different samples and report the averaged performance metrics over all the five runs.

Figure 23 plots the results on assortative mixing networks measured by RelErr, PredLik and RankCorr with respect to the number of training data. It can be observed from Figure 23 that when the number of training samples increase, the RelErr decreases and both PredLik and RankCorr increase, indicating that all methods can improve accuracy of estimation with more training samples. Moreover, LowRank and Sparse outperforms Full in all cases. Therefore, we conclude that utilizing the structure of the matrix can improve the estimation very significantly. LowRankSparse outperforms all other baselines since it fully utilizes prior information about the infectivity matrix. It is interesting to observe that when the number of training samples are small, the improvements of LowRankSparse over other baselines are very large. We think this is because when the number of training samples are not sufficiently large to get a good estimation, the prior knowledge from the structure of the infectivity matrix becomes more important and useful. TimeWindow is not as good as other methods since it can not capture the time pattern very accurately. Similarly in Figure 24, the proposed method LowRankSparse outperforms other methods in disassortative mixing networks. These two sets of experiments indicate that LowRankSparse can handle well different network topologies.

In Figure 25 and Figure 26, we plot the performance with respect to the values of the two parameters λ_1 and λ_2 in LowRankSparse. It can be observed that the performance first increases and then decreases when the value λ_1 grows. Note that when $\lambda_1 = 0$, we obtain a sparse solution that is not low-rank, which is underperformed by solutions that are both low-rank and sparse. Similar observations can be made for λ_2.

In order we investigate the convergence of the proposed algorithm, in Figure 27, we present the performance measured by PredLik with respect to the number of outer iterations K in Algorithm 3. We can observe that the performance measured by PredLik grows with the number of outer iterations and converges within about 50 iterations. We also illustrate the impact of the number of inner iterations in Figure 27. It can be observed that larger number of inner iterations leads to better convergence speed and slightly better performance.

5.3.2 Real-world Data

We also evaluate the proposed method on a real world data set. To this end, we use the MemeTracker data set[2]. The data set contains the information flows captured by hyper-links between different sites with timestamps. In particular, we first extract the top 500 popular sites and the links between them. The events are in the form that a site created a hyper-link to another site at a particular time. We use 50% data as training data and 50% as test data.

In Figure 28, we show that negative log-likelihood of Full, Sparse, LowRank and LowRankSparse on the test set. We can see that LowRankSparse outperforms the baselines. Therefore, we conclude that LowRankSparse can better model the influences in social networks.

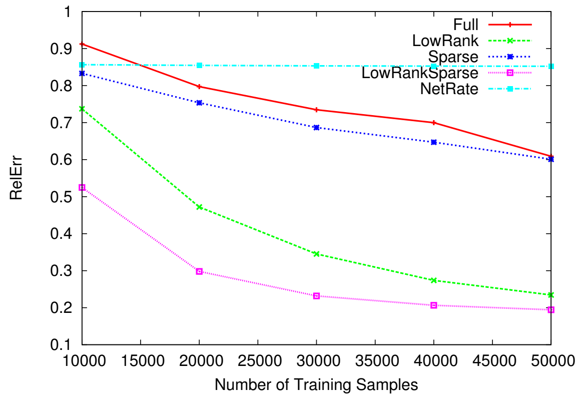
We also study whether the proposed model can discover the influence network between users from the recurrent events. To this end, we present the RankCorr of Full, Sparse, LowRank and LowRankSparse in Figure 29. We also include NetRate as a

[2]http://memetracker.org

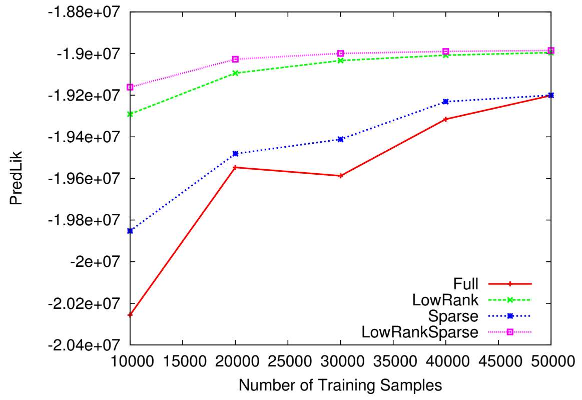
baseline. It can be observed that the LowRankSparse obtains better rank correlation than other models, which indicates that it can capture the influence network better than other models. In Figure 30, we visualize the influence network estimated from the MemeTracker data. We can observe that there is a quite dense region near the bottom right in the infectivity matrix. This region represents that the corresponding sites are the centric of the infectivity networks. Example sites in this region include `news.cnet.com`, `blogs.zdnet.com` and `blogs.abcnews.com`. The first two are both famous IT news portal and the third one is a blog site that belongs to a general news portal. It is clear that all of these sites are popular sites that can quickly detect trending events and propragate them to a lot of other sites.

5.4 Summary

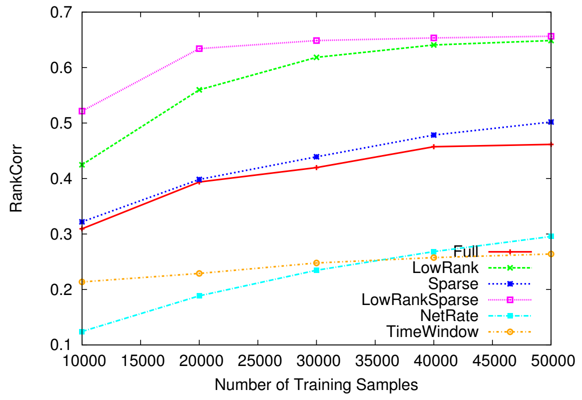
In this chapter, we propose to infer the network social influence from the observed recurrent events indicating users' activities in the social networks. The proposed model utilizes the mutually-exciting multi-dimensional Hawkes model to capture the temporal patterns of user behaviors. Moreover, we estimate the infectivity matrix for the network that is both low-rank and sparse by optimizing nuclear norm and ℓ_1 norm simultaneously. The resulting optimization problem is solved through combining the ideas of alternating direction method of multipliers and majorization-minimization. The experimental results on both simulation and real-world datasets suggest that the proposed model can estimate the social influence between users accurately.



(a) RelErr

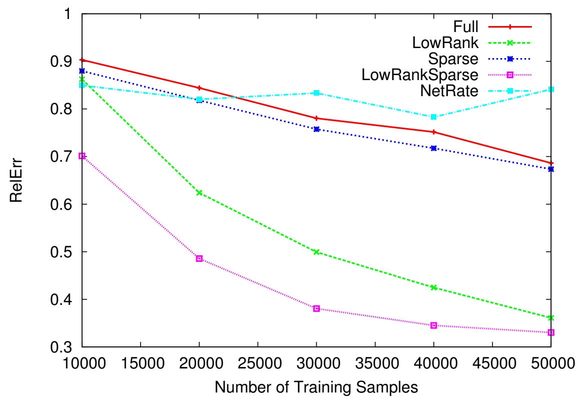


(b) PredLik

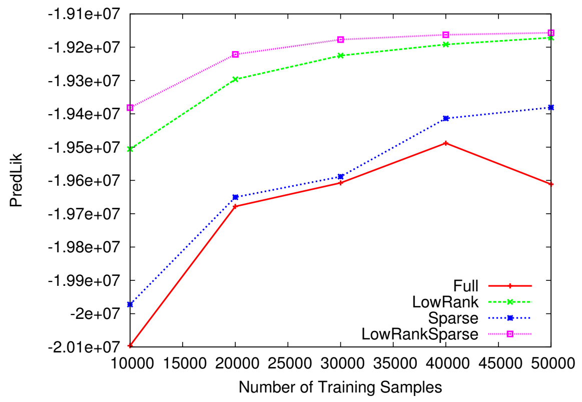


(c) RankCorr

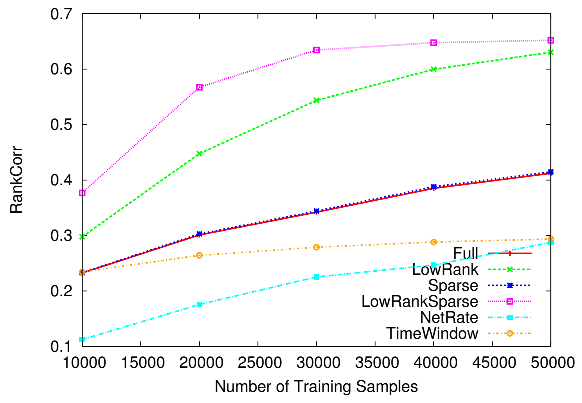
Figure 23: Assortative mixing networks: performance measured by RelErr, PredLik and RankCorr with respect to the number of training samples.



(a) RelErr



(b) PredLik



(c) RankCorr

Figure 24: Disassortative mixing networks: Performance measured by RelErr, PredLik and RankCorr with respect to the number of training samples.

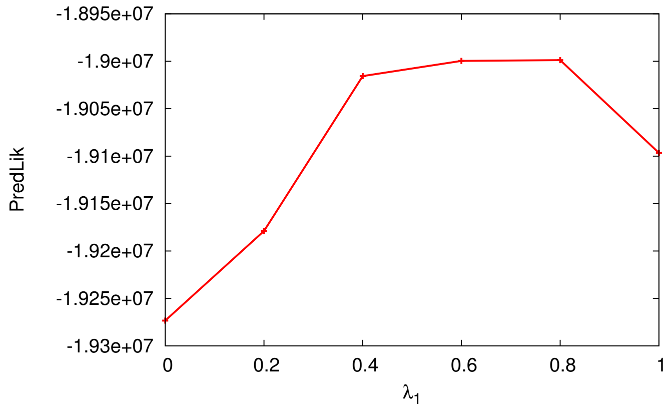


Figure 25: Performance measured by PredLik with respect to the value of λ_1.

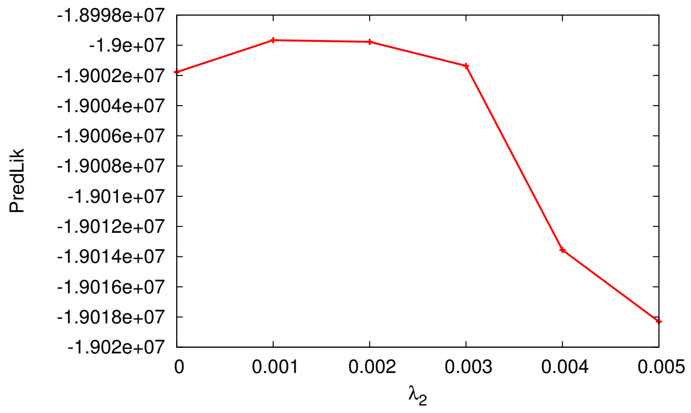


Figure 26: Performance measured by PredLik with respect to the value of λ_2.

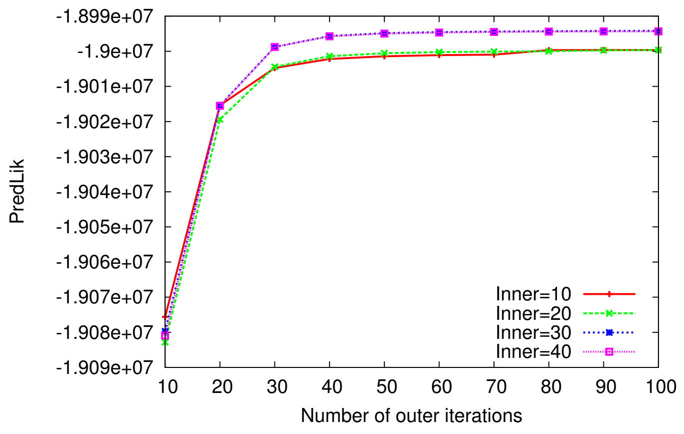


Figure 27: Performance measured by PredLik with respect to the number of inner/outer iterations.

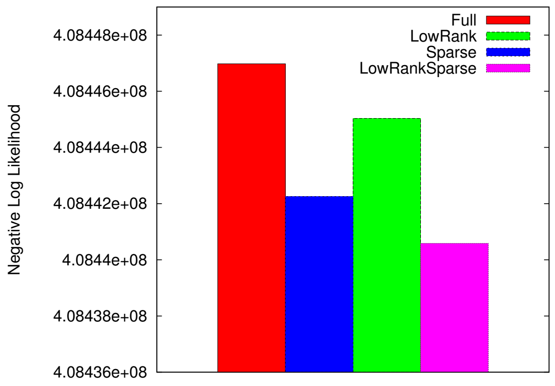


Figure 28: Performance measured by PredLik on MemeTracker data set.

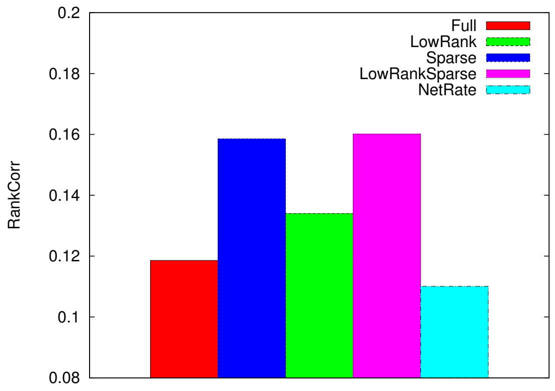


Figure 29: Performance measured by RankCorr on MemeTracker data set.

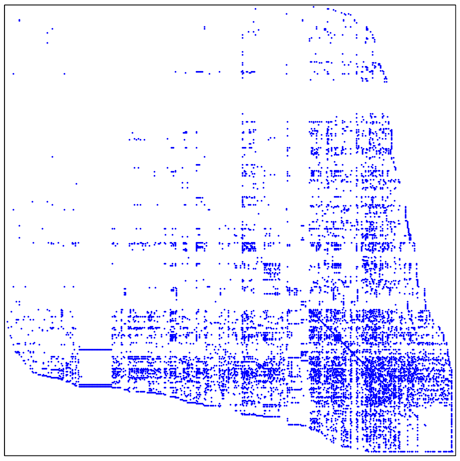


Figure 30: Influence structure estimated from the MemeTracker data set.

CHAPTER VI

LEARNING TRIGGERING KERNELS FOR MULTI-DIMENSIONAL HAWKES PROCESSES

In this chapter, we describe the work for estimating triggering kernels for multi-dimensional Hawkes process. As we shown in Chapter 5, given a multi-dimensional Hawkes process, the conditional intensity for the u-th dimension expressed as follows:

$$\lambda_u(t) = \mu_u + \sum_{i:t_i<t} g_{uu_i}(t - t_i),$$

where $\mu_u \geq 0$ is the base intensity for the u-th Hawkes process. The kernel $g_{uu'}(t) \geq 0$ captures the mutually-exciting property between the u-th and u'-th dimension. Intuitively, it captures the dynamics of influence of events occurred in the u'-th dimension to the u-th dimension. Larger value of $g_{uu'}(t)$ indicates that events in u'-th dimension are more likely to trigger a event in the u-th dimension after a time interval t. As we have mentioned before, the triggering kernel $g_{uu'}(t)$ plays a central role in modeling the dynamics of temporal events. Moreover, it can be quite complex in real-world applications. Therefore, it is desirable to estimate it from the observer data as we propose in this chapter. Specifically, we propose to estimate the triggering kernels from a infinite dimensional functional space through combining the ideas of Euler-Lagrange equation and majorization minimization. This method also exploits the low-rank structure in functional spaces.

6.1 Nonparametric Triggering Kernel Estimation using Euler-Lagrange Equations

First we introduce some notations: We collect the parameters of the multi-dimensional Hawkes process into matrix-vector forms, $\boldsymbol{\mu} = (\mu_u)$ for the base intensity and $\mathbf{G} = (g_{uu'}(t))$ into a matrix. These parameters can be estimated by optimizing the log-likelihood over the observed events that are sampled from the process.

6.1.1 Optimization Problem and Space

Suppose we have m samples, $\{c_1, \dots, c_m\}$, from the multi-dimensional Hawkes process. Each sample c is a sequence of events observed during a time period of $[0, T_c]$, which is in the form of $\{(t_i^c, u_i^c)\}_{i=1}^{n_c}$. Each pair (t_i^c, u_i^c) represents an event occurring at the u_i^c-th dimension at time t_i^c. Thus, the log-likelihood of model parameters $\Theta = \{\mathbf{G}, \boldsymbol{\mu}\}$ can be expressed as follows [90]:

$$\begin{aligned}\mathcal{L}(\Theta) =& \sum_c \left(\sum_{i=1}^{n_c} \log \lambda_{u_i^c}(t_i^c) - \sum_{u=1}^{U} \int_0^{T_c} \lambda_u(t) dt \right) \\ =& \sum_c \left(\sum_{i=1}^{n_c} \log \left(\mu_{u_i^c} + \sum_{t_j^c < t} g_{u_i^c u_j^c}(t_i^c - t_j^c) \right) \right. \\ & \left. - T_c \sum_{u=1}^{U} \mu_u - \sum_{u=1}^{U} \sum_{j=1}^{n_c} \int_0^{T_c - t_j} g_{u u_j^c}(s) ds \right). \end{aligned} \tag{33}$$

In general, the triggering kernels $g_{uu'}(t)$ as well as the base intensity $\boldsymbol{\mu}$ can be estimated by maximizing the log-likelihood, *i.e.*, $\min_{g_{uu'}(t) \geq 0, \boldsymbol{\mu} \geq 0} \; -\mathcal{L}(\Theta)$.

We assume that the triggering kernels $g_{uu'}(t)$ can be expressed by a linear combination of a set of D base kernels. Formally, we have

$$g_{uu'}(t) = \sum_{d=1}^{D} a_{uu'}^d g_d(t),$$

where $\{g_d(t)|d = 1, 2, \ldots, D\}$ are the base kernels and $a^d_{uu'}$ are the coefficients for the linear combination. In our work, both $a^d_{uu'}$ and $g_d(t)$ are estimated from the data. In particular, we propose to estimate the base kernels $g_d(t)$ from an infinite dimensional functional space. To this end, we consider the following penalized log-likelihood function function, i.e.,

$$\min_{\Theta} \mathcal{L}_\alpha(\Theta),$$

where the penalized log-likelihood function $\mathcal{L}_\alpha(\Theta)$ is defined as follows:

$$\mathcal{L}_\alpha(\Theta) = -\mathcal{L}(\Theta) + \alpha \left(\sum_d \mathcal{R}(g_d) + \sum_{u,u',d} (a^d_{uu'})^2 \right).$$

Here the first term is the negative log-likelihood of the parameters and the second term represents the regularization of both the base function $g_d(t)$ and the coefficient $a^d_{uu'}$. The parameter α determines the trade-off between these two terms. Moreover, the functional $\mathcal{R}(g_d)$ is a penalty term preferring smooth base kernels. In general, the choice of the penalty should take into account of the prior knowledge of the triggering kernels and thus is application-dependent. For the sake of concreteness and tractability, we use $\mathcal{R}(g) = \int_0^\infty g'(t)^2 dt$ in the rest of this chapter, where $g'(t)$ is the derivative of $g(t)$ with respect to t.

It appears that the above problem is similar to the smoothing splines and can be solved through methods that transform the above problem to a finite dimensional least squares optimization problem [158]. As we discussed in Section 2.5, however, the main difference is that the log-likelihood function defined in Equation (33) contains the integral over the triggering kernels that depends on the values of the triggering kernels over the whole time interval rather than only a finite number of points as required by the smoothing splines. As a result, it is difficult to directly apply the smoothing spline methods in our case.

Even more challenging is that the above objective function is difficult to optimize in general due to the fact that the parameters are not only infinite dimensional but also are coupled. In this chapter, inspired by the work of [86] on one-dimensional Hawkes process, we propose MMEL to estimate the triggering kernels for multi-dimensional Hawkes process, which combines the idea of constructing a tight upper-bound as a surrogate to decouple parameters and the application of Euler-Lagrange equations to deal with the infinite dimensionality of the parameters.

6.1.2 Iterative Algorithm

Our algorithm updates the parameters Θ in an iterative manner which, as we will show later, ensures that the objective function $\mathcal{L}_\alpha$ decrease monotonically. In particular, we construct a tight upper-bound $Q(\Theta|\Theta^{(k)})$ for current parameter estimation $\Theta^{(k)}$ and optimize the upper-bound $Q(\Theta|\Theta^{(k)})$ to obtain the updates for the parameters. Specifically, the upper-bound $Q(\Theta|\Theta^{(k)})$ is defined as follows:

$$
\begin{aligned}
Q(\Theta|\Theta^{(k)}) = & -\sum_c \left[\sum_{i=1}^{n_c} \left(p_{ii}^c \log \frac{\mu_{u_i^c}}{p_{ii}^c} + \sum_{j=1}^{i-1} \sum_{d=1}^{D} p_{ijd}^c \log \frac{a_{u_i^c u_j^c}^d g_d(t_i^c - t_j^c)}{p_{ijd}^c} \right) \right. \\
& \left. + \left(T_c \sum_u \mu_u + \sum_{u=1}^{U} \sum_{j=1}^{n_c} \sum_{d=1}^{D} \int_0^{\tau_j^c} \left((a_{uu_j^c}^d)^2 \frac{g_d^{(k)}(t)}{2a_{uu_j^c}^{d,(k)}} + g_d^2(t) \frac{a_{uu_j^c}^{d,(k)}}{2g_d^{(k)}(t)} \right) dt \right) \right] \\
& + \alpha \left(\sum_d \mathcal{R}(g_d) + \sum_{u,u',d} (a_{uu'}^d)^2 \right),
\end{aligned} \tag{34}
$$

where $\tau_j^c = T_c - t_j^c$ and p_{ij}^c and p_{ii}^c are defined as follows:

$$p_{ii}^c = \frac{\mu_{u_i^c}^{(k)}}{\mu_{u_i^c}^{(k)} + \sum_{j=1}^{i-1} \sum_d a_{u_i u_j}^{d,(k)} g_d^{(k)}(t_i^c - t_j^c)},$$
$$p_{ijd}^c = \frac{a_{u_i u_j}^{d,(k)} g_d^{(k)}(t_i^c - t_j^c)}{\mu_{u_i^c}^{(k)} + \sum_{j=1}^{i-1} \sum_d a_{u_i u_j}^{d,(k)} g_d^{(k)}(t_i^c - t_j^c)}.$$

Intuitively, p_{ijd}^c can be interpreted as the probability that the i-th event is influenced by a previous event j through the d-th base kernel and p_{ii}^c is the probability that i-th event is sampled from the base intensity. Thus, the first two terms of $Q(\Theta|\Theta^{(k)})$ can be viewed as the joint probability of the unknown influence structures and the observed events.

We first show that the following properties hold for $Q(\Theta; \Theta^{(k)})$ defined in Equation (34):

Theorem 5. *The following properties hold for $Q(\Theta; \Theta^{(k)})$:*

1. *For all Θ and $\Theta^{(k)}$, $Q(\Theta; \Theta^{(k)}) \geq \mathcal{L}_\alpha(\Theta)$.*

2. $Q(\Theta^{(k)}; \Theta^{(k)}) = \mathcal{L}_\alpha(\Theta^{(k)})$.

Proof. The first claim can be shown by utilizing the Jensen's inequality: For all c and i, we have

$$\log(\mu_{u_i^c} + \sum_{j=1}^{i-1} \sum_{d=1}^{D} a_{u_i^c u_j^c}^d g_d(t_i^c - t_j^c)) \geq p_{ii}^c \log \frac{u_i^c}{p_{ii}^c} + \sum_{j=1}^{i-1} \sum_{d=1}^{D} p_{ijd}^c \log \frac{a_{u_i^c u_j^c}^d g_d(t_i^c - t_j^c)}{p_{ijd}^c} \quad (35)$$

Moreover, by the inequality of arithmetic and geometric means:

$$(a_{uu_j^c}^d)^2 \frac{g_d^{(k)}(t)}{2a_{uu_j^c}^{d,(k)}} + g_d^2(t) \frac{a_{uu_j^c}^{d,(k)}}{2g_d^{(k)}(t)} \geq a_{uu_j^c}^d g_d(t)$$

By noting that summation and integration preserve the above two inequalities, we prove the first claim.

The second claim can be checked by setting $g_d(t) = g_d^{(k)}(t)$, $a_{uu'}^d = a_{uu'}^{d,(k)}$ and $\mu_u = \mu_u^{(k)}$. $\square$

The above two properties imply that if $\Theta^{(k+1)} = \operatorname{argmin}_\Theta Q(\Theta; \Theta^{(k)})$, we have $\mathcal{L}_\alpha(\Theta^{(k)}) \geq \mathcal{L}_\alpha(\Theta^{(k+1)})$. Thus, optimizing Q with respect to Θ at each iteration ensures that the value of $\mathcal{L}_\alpha(\Theta)$ decrease monotonically.

Update for μ_u and $a_{uu'}^d$. Moreover, the advantage of optimizing $Q(\Theta|\Theta^{(k)})$ is that all parameters g_d and $a_{uu'}^d$ can be solved independently from each other in closed form, and the non-negativity constraints are automatically taken care of. Specifically, we have the following update rules for μ_u and $a_{uu'}^d$:

$$\mu_u^{(k+1)} = \frac{1}{\sum_c T_c} \left(\sum_c \sum_{i=1, u_i^c = u}^{n_c} p_{ii}^c \right) \tag{36}$$

$$a_{uu'}^{d,(k+1)} = \left(\frac{a_{uu'}^{d,(k)} \sum_c \sum_{i:u_i^c = u} \sum_{j<i, u_j^c = u'} p_{ijd}^c}{\sum_c \sum_{j:u_j^c = u'} \int_0^{T_c - t_j^c} g_d^{(k)}(t) dt + \alpha} \right)^{\frac{1}{2}} \tag{37}$$

Update for g_d. The corresponding update for g_d can be derived by optimizing in an infinite dimensional space. Specifically, for every $d = 1, \ldots, D$, we consider the terms in $Q(\Theta|\Theta^{(k)})$ that are related to g_d as follows:

$$\begin{aligned} \min_{g_d \in L_1(\mathbb{R})} - \sum_c & \left(\sum_{i=1}^{n_c} \sum_{j=1}^{i-1} p_{ijd}^c \log g_d(t_i^c - t_j^c) \right. \\ & \left. - \sum_{u=1}^{U} \sum_{j=1}^{n_c} \int_0^{T_c - t_i^c} g_d^2(t) \frac{a_{uu_j^c}^{d,(k)}}{2 g_d^{(k)}(t)} dt \right) + \alpha \mathcal{R}(g_d). \end{aligned} \tag{38}$$

The optimization problem in Equation (38) is equivalent to minimize $L[t, g, g'] =$

$\int_0^\infty F(t, g, g')dt$, where

$$
\begin{aligned}
F(t, g, g') = & -\sum_c \sum_{i=1}^{n_c} \sum_{j=1}^{i-1} p_{ijd}^c \log g_d(t) \mathbb{I}[t = t_i^c - t_j^c] \\
& + \sum_c \sum_{u=1}^{U} \sum_{j=1}^{n_c} g_d^2(t) \frac{a_{uu_j^c}^{d,(k)}}{2g_d^{(k)}(t)} \mathbb{I}[t \le T_c - t_i^c] + \alpha_2 (g_d'(t))^2
\end{aligned}
$$

By Euler-Lagrange equation, the solution satisfies

$$
\frac{\partial F}{\partial g_d} - \frac{d}{dt}[\frac{\partial F}{\partial g_d'}] = 0
$$

Substitute F into the above equation, we get the follows:

$$
-\frac{D(t)}{g_d(t)} + C(t) g_d(t) - 2\alpha g_d''(t) = 0, \tag{39}
$$

where

$$
\begin{aligned}
C(t) = & \sum_c \sum_{u=1}^{U} \sum_{j=1}^{n_c} \frac{a_{uu_j^c}^{d,(k)}}{g_d^{(k)}(t)} \mathbb{I}[t \le T_c - t_i^c] \\
D(t) = & \sum_c \sum_{i=1}^{n_c} \sum_{j=1}^{i-1} p_{ijd}^c \mathbb{I}[t = t_i^c - t_j^c],
\end{aligned}
$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 if the predicate in parameter is true and 0 otherwise. We solve the above ODE numerically using the following Seidel type iterations which is quite efficient. Specifically, we discretized the differential equation over small intervals $t_m = m\Delta t$, for $m = 1, \ldots, M$. Let $g_{d,m} = g_d(t_m)$. We can express the derivative of $g(t)$ as follows:

$$
\begin{aligned}
g_d'(t_m) & \approx \frac{g_d(t_{m+1}) - g_d(t_m)}{t_{m+1} - t_m} = \frac{g_{d,m+1} - g_{d,m}}{\Delta t} \\
g_d''(t_m) & \approx \frac{g_d(t_{m+1}) - 2g_d(t_m) + g_d(t_m)}{\Delta t^2}
\end{aligned}
$$

Therefore, the discretized ODE can be expressed as follows:

$$
-2\alpha \frac{g_{d,m+1} - 2g_{dm} + g_{d,m-1}}{\Delta t^2} + C_m g_{dm} = \frac{D_m}{g_{dm}}, \tag{40}
$$

Algorithm 4 (MMEL) for estimating parameters

```
Input: Observed samples $\{c_1, \ldots, c_m\}$.
Output: Estimation of $\mu_u$, $a^d_{uu'}$ and $g_d$.
  Initial $\mu_u$, $a_{uu'}$ and $g_d$ randomly.
  while not converge do
    Update $\mu_u$, $a_{uu'}$ by Equation (36) and (37) for $u, u' = 1, \ldots, U$.
    for d=1,..., D do
      while not converge do
        Solve Equation (40) for $m = 1, 2, \ldots, M$.
      end while
    end for
  end while
```

where

$$C_m = \frac{1}{g_d^{(k)}(m\Delta t)} \sum_c \sum_{u=1}^{U} \sum_{j=1}^{n_c} a^{d,(k)}_{uu^c_j} \mathbb{I}[m\Delta t \le T_c - t^c_i]$$

$$D_m = \frac{1}{\Delta t} \sum_c \sum_{i,j: m\Delta t \le t^c_i - t^c_j < (m+1)\Delta t} p^c_{ijd}$$

Therefore, we can solve for $g_{d,m}$ by fixing all other $g_{d,m'}$, $m' \neq m$ but solving the above quadratic equation. We summary the proposed algorithm in Algorithm 4.

6.2 Extension to Spatial-Temporal Process

The proposed idea can be extended to estimate the kernels for spatial temporal process. In this section, we describe the application of the Euler-Lagrange equation to estimate the kernel of spatial temporal process. In this case, the conditional intensity function can be expressed as follows:

$$\begin{aligned}\lambda(t, x, y|\mathcal{H}) &= \mu(x, y) + \sum_{i: t_i < t} g(t - t_i, x - x_i, y - y_i) \\ &= \mu(x, y) + \int_0^t \int_\Omega g(t - s, x - \xi, y - \psi) N(ds, d\xi, d\psi),\end{aligned}$$

where the kernel $g(t, x, y)$ captures the dynamic over both the spatial and temporal dimensions and the function $\mu(x, y)$ is base intensity. For the sake of simplicity, we assume that $\mu(x, y)$ does not depend on time, but the method we proposed can be applied to the case where μ is also time-dependent.

Similar to the case of temporal Hawkes process, the samples from the above process can be expressed as $\{(t_i, x_i, y_i)\}_{i=1}^{N_c}$, where t_i, x_i, y_i are the time and location of the i-th event. Given samples from the spatial-temporal process, the log-likelihood function can be expressed as follows:

$$
\begin{aligned}
\mathcal{L}(\Theta) &= \sum_c \sum_{i=1}^{N_c} \left(\log \lambda(t_i^c, x_i^c, y_i^c) - \int_0^{T_c} \int_\Omega \lambda(t, x, y) dtdxdy \right) \\
&= \sum_c \sum_{i=1}^{N_c} \log \lambda(t_i^c, x_i^c, y_i^c) \\
- &\quad \sum_c \sum_{i=0}^{N_c} \int_{t_i^c}^{t_{i+1}^c} \int_\Omega \left(\mu(x, y) + \sum_{j:j<i} g(t - t_j^c, x - x_j^c, y - y_j^c) \right) dtdxdy \\
&= \sum_c \sum_{i=1}^{N_c} \log \lambda(t_i^c, x_i^c, y_i^c) - CTU \\
&\quad - \sum_c \int_\Omega \sum_{i=0}^{N_c} \sum_{j<i} \int_{t_i^c}^{t_{i+1}^c} g(t - t_j^c, x - x_j^c, y - y_j^c) dtdxdy \\
&= \sum_c \sum_{i=1}^{N_c} \log \lambda(t_i^c, x_i^c, y_i^c) - CTU - \sum_c \sum_{j=1}^{N_c} \int_{t_j^c}^{T^c} \int_\Omega g(t - t_j^c, x - x_j^c, y - y_j^c) dtdxdy,
\end{aligned}
$$

where $U = \int_\Omega \mu(x, y) dxdy$.

In order to estimate the triggering kernel $g(t, x, y)$ and the base intensity $\mu(x, y)$, we can optimize the penalized likelihood function defined as follows:

$$
\min_{\Theta=(g,\mu)} \mathcal{L}_\alpha = -\mathcal{L}(\Theta) + \alpha(\mathcal{R}_1[\mu] + \mathcal{R}_2[g]), \tag{41}
$$

where $\mathcal{R}_1$ and $\mathcal{R}_2$ are penalty functional defined as follows:

$$
\begin{aligned}
\mathcal{R}_1[\mu] &= \int_\Omega (\frac{\partial \mu}{\partial x})^2 + (\frac{\partial \mu}{\partial y})^2 dxdy \\
\mathcal{R}_2[g] &= \int_0^T \int_\Omega (\frac{\partial g}{\partial t})^2 + (\frac{\partial g}{\partial x})^2 + (\frac{\partial g}{\partial y})^2 dtdxdy
\end{aligned}
$$

Both $\mathcal{R}_1[\mu]$ and $\mathcal{R}_2[g]$ prefers smooth functions, which plays the rule of regularization in the estimation process.

The objective function defined in Equation (41) can be difficult to optimize for similar reasons as the temporal case: 1) Its parameters are coupled with each other. 2) We would like to optimize with respect to a infinite dimensional space. To solve the problem, we can apply the idea of majorization minimization. Define $Q(\Theta|\Theta^{(k)})$ as follows:

$$\begin{aligned}
Q(\Theta|\Theta^{(k)}) =& \sum_c \sum_{i=1}^{N_c} \left(p_{ii}^c \log \frac{\mu(x_i^c, y_i^c)}{p_{ii}^c} + \sum_{j=1}^{i-1} p_{ij}^c \log \frac{g(t_i^c - t_j^c, x_i^c - x_j^c, y_i^c - y_j^c)}{p_{ij}^c} \right) \\
& - CTU - \sum_c \sum_{j=1}^{N_c} \int_{t_j^c}^{T} \int_\Omega g(t - t_j^c, x - x_j^c, y - y_j^c) dt dx dy
\end{aligned}$$

where

$$\begin{aligned}
p_{ii}^c &= \frac{\mu^{(k)}(x_i^c, y_i^c)}{\mu^{(k)}(x_i^c, y_i^c) + \sum_{j=1}^{i-1} g^{(k)}(t_i^c - t_j^c, x_i^c - x_j^c, y_i^c - y_j^c)} \\
p_{ij}^c &= \frac{g^{(k)}(t_i^c - t_j^c, x_i^c - x_j^c, y_i^c - y_j^c)}{\mu^{(k)}(x_i^c, y_i^c) + \sum_{j=1}^{i-1} g(t_i^c - t_j^c, x_i^c - x_j^c, y_i^c - y_j^c)}
\end{aligned}$$

The nice property of $Q(\Theta|\Theta^{(k)})$ is that the parameters g and μ are decouple and thus can be optimized independently.

Optimizing with respect to $\mu(x, y)$.

We first collect the terms in $Q(\Theta|\Theta^{(k)})$ that are related to μ:

$$\min_g \int_\Omega \left(-\sum_c \sum_i p_{ii}^c \log \frac{\mu(x,y)}{p_{ii}^c} \mathbb{I}[x = x_i^c \wedge y = y_i^c] + CT\mu(x,y) + \alpha \left((\frac{\partial \mu}{\partial x})^2 + (\frac{\partial \mu}{\partial y})^2 \right) \right) dxdy$$

We can apply Euler-Lagrange equation to show that the solution satisfies the following ordinary differential equations:

$$-\sum_{i,c}\frac{p_{ii}^c}{\mu(x,y)}\mathbb{I}[x=x_i^c\wedge y=y_i^c]+T-2\alpha\frac{\partial^2\mu}{\partial x^2}-2\alpha\frac{\partial^2\mu}{\partial y^2}=0$$

We can discretize the above equations by letting $u_{mn}=\mu(m\Delta x,n\Delta y)$:

$$-\frac{C_{mn}}{\mu_{mn}}+T-2\alpha\frac{u_{m-1,n}-2u_{mn}+u_{m+1,n}}{\Delta x^2}-2\alpha\frac{u_{m,n-1}-2u_{mn}+u_{m,n+1}}{\Delta y^2}=0 \tag{42}$$

where

$$C_{mn}=\frac{1}{|S_{mn}|}\sum_{i,c}p_{ii}^c\mathbb{I}[(x_i^c,y_i^c)\in S_{mn}],$$

Optimizing with respect to $g(t,x,y)$.

Similarly, we first collect the terms in $Q(\Theta|\Theta^{(k)})$ that are related to $g(t,x,y)$:

$$\min_g\int_0^T\int_\Omega\Bigg(-\sum_{c,i}\sum_{j=1}^{i-1}p_{ij}^c\log\frac{g(t,x,y)}{p_{ij}^c}\mathbb{I}[t=t_i^c\wedge x=x_i^c\wedge y=y_i^c]$$
$$+\sum_c\sum_{j=1}^{N_c}g(t,x,y)\mathbb{I}[t\geq t_j^c\wedge(x-x_j^c,y-y_j^c)\in\Omega]+2\alpha(\frac{\partial g}{\partial t})^2+(\frac{\partial g}{\partial x})^2+(\frac{\partial g}{\partial y})^2\Bigg)\,dxdydt$$

Therefore, the optimal kernel $g(t,x,y)$ satisfies the following equations:

$$-\sum_{c,i}\sum_{j=1}^{i-1}\frac{p_{ij}^c\mathbb{I}[t=t_i^c-t_j\wedge x=x_i^c-x_j^c\wedge y=y_i^c-y_j^c]}{g(t,x,y)}+\sum_c\sum_{j=1}^{N_c}\mathbb{I}[t\geq t_j^c\wedge(x-x_j^c,y-y_j^c)\in\Omega]$$
$$-2\alpha\left(\frac{\partial^2 g}{\partial t^2}+\frac{\partial^2 g}{\partial x^2}+\frac{\partial^2 g}{\partial y^2}\right)=0$$

We can discretize the above equation over $u_{mnp}=g(m\Delta t,n\Delta x,p\Delta y)$ where $m=0,\ldots,M_t$, $n=0,\ldots,M_x$ and $p=0,\ldots,M_y$.

$$-\frac{A_{mnp}}{u_{mnp}}+B_{mnp}-2\alpha\frac{u_{m-1,np}-2u_{mnp}+u_{m+1,np}}{\Delta t^2}-2\alpha\frac{u_{m,n-1,p}-2u_{mnp}+u_{m,n+1,p}}{\Delta x^2}$$
$$-2\alpha\frac{u_{m,n,p-1}-2u_{mnp}+u_{m,n,p+1}}{\Delta y^2}=0, \tag{43}$$

```
Algorithm 5 Estimating nonparametric kernels for spatial-temporal Hawkes process
Input: Observed samples $\{c_1, \ldots, c_m\}$ .
Output: Estimation of the triggering kernel $g(t, x, y)$ and base intensity $\mu(x, y)$.
  Initialize $\mu(x, y)$, $g(t, x, y)$ randomly.
  while not converge do
    while not converge do
      Update $g(t, x, y)$ by Equation (43) for all $m, n, p$.
    end while
    while not converge do
      Update $\mu(x, y)$ by Equation (42) for all $m, n$.
    end while
  end while
```

where

$$A_{mnp} = \frac{1}{|S_{mnp}|} \sum_{c,i} \sum_{j=1}^{i-1} p_{ij}^c \mathbb{I}[(t_i^c - t_j^c, x_i^c - x_j^c, y = y_i^c - y_j^c) \in S_{mnp}],$$

$$B_{mnp} = \sum_{c} \sum_{j=1}^{N_c} \mathbb{I}[m\Delta t \geq t_j^c \wedge (n\Delta x - x_j^c, p\Delta y - y_j^c) \in \Omega]$$

We summary the proposed algorithm in Algorithm 5.

6.3 Experiments

In this section, we conduct experiments on both synthetic and real-world datasets to evaluate the performance of the proposed method MMEL.

6.3.1 Toy Data

In order to illustrate that the proposed method can estimate the triggering kernels from data very accurately, we first conduct a set of experiments in toy data sets of 2-dimensional Hawkes processes. The goal is to visualize and compare the triggering kernels estimated from data to the ground-truth.

Data Generation. The true parameters of the 2-dimensional Hawkes process are generated as follows: the base intensity $\mu_1 = \mu_2 = 0.1$. We generate two data sets with

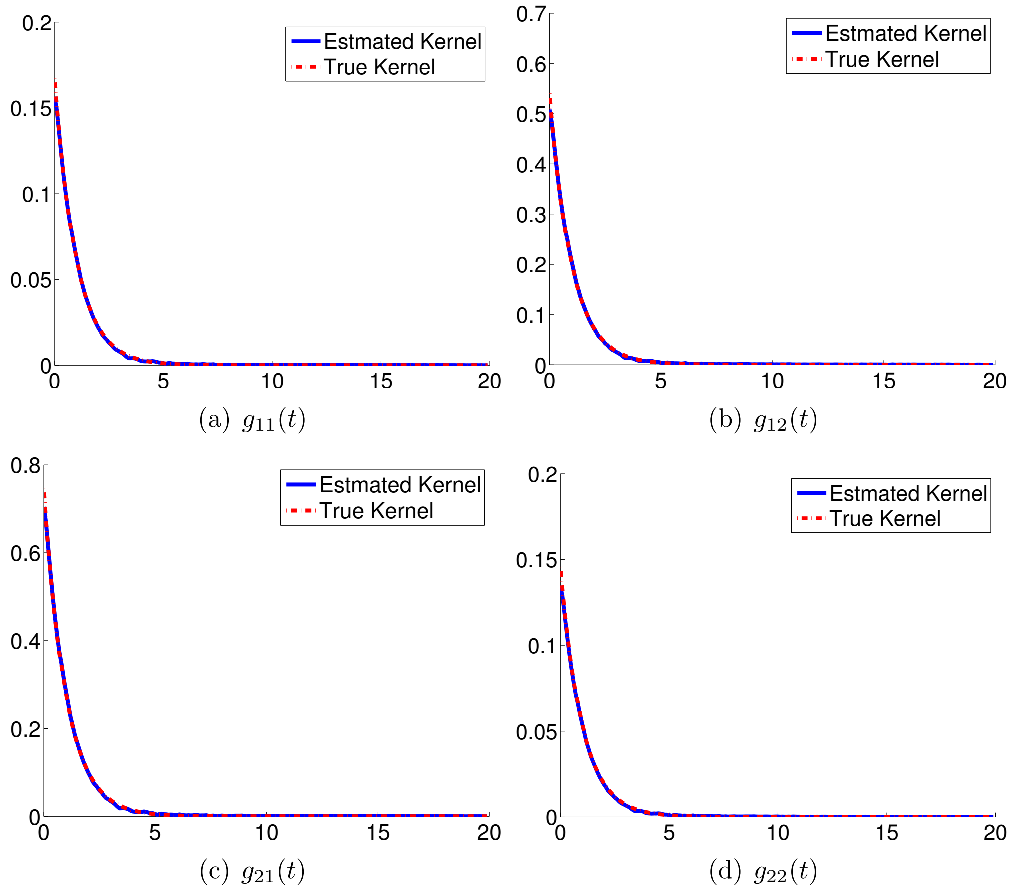
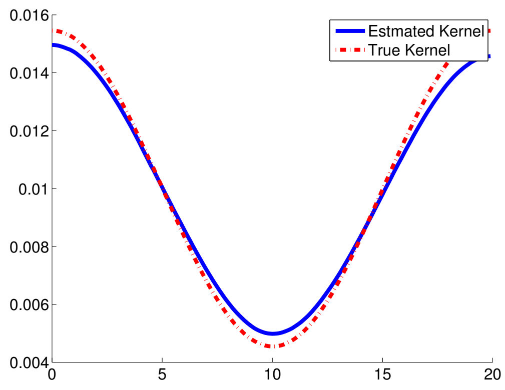
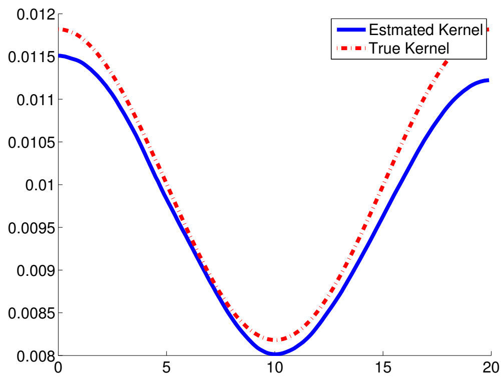


Figure 31: Estimated vs. True Triggering Kernels on toy data DataExp with exponential kernels. The four sub-figures show both the true triggering kernels and the estimated ones from the data

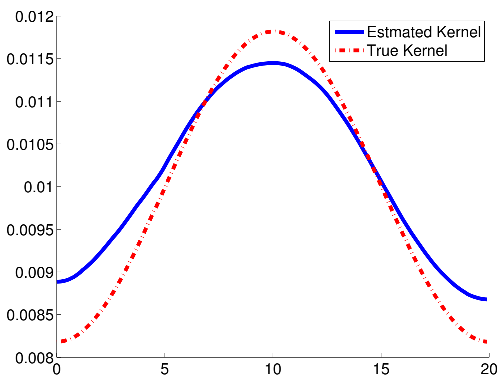
different triggering kernels: 1) DataExp. In the first data set, we use the exponential kernel $g(t) = \exp(-t)$, one of the most widely used trigger kernel for Hawkes processes, to demonstrate that the proposed algorithm can obtain good estimations in this case. 2) DataCos. In this case, we consider relatively complex kernels rather than simple exponential kernels. Specifically, in this case, the triggering kernels are generated by the linear combination of two base functions: $g_1(t) = \cos(\pi t/10) + 1.1$ and $g_2(t) = \cos(\pi(t/10 + 1)) + 1.1$. In both data sets, the coefficients of the linear combinations are generated from a uniform distribution on $[0.1, 0.2]$.



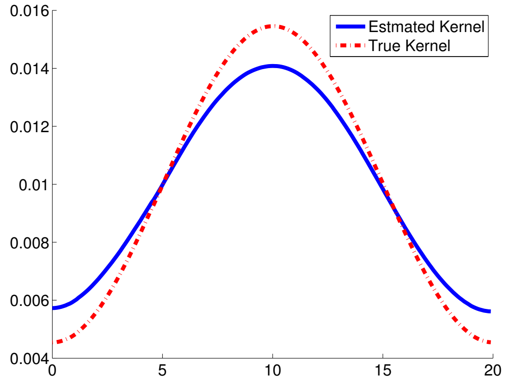
(a) $g_{11}(t)$



(b) $g_{12}(t)$



(c) $g_{21}(t)$



(d) $g_{22}(t)$

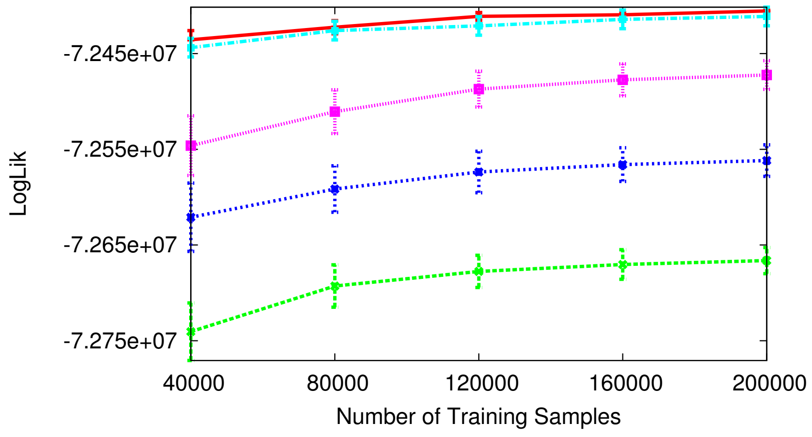
Figure 32: Estimated vs. True Triggering Kernels on toy data DataCos. The four sub-figures show both the true triggering kernels and the estimated ones from the data

Results. We generate 100,000 samples from the two 2-dimensional Hawkes processes described above on time interval $[0, 20]$ as the training sets and run MMEL on the sampled data to obtain the estimations for the triggering kernels. In order to visualize the estimated kernels and compare them to the ground truth, we plot both the estimated and the true kernels for each pairs of dimensions. In Figure 31, we plot the triggering kernel obtained from the data by MMEL together with the true exponential kernel. We can observe that the estimated and true kernels almost overlap each other, which indicates that the estimated triggering kernels are very accurate in this case.

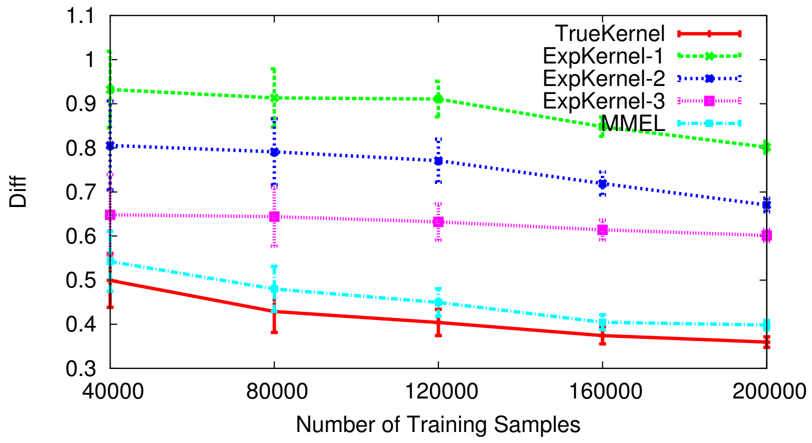
In Figure 32, we visualize the triggering kernels learnt from the DataCos data set. It can be observed that the proposed MMEL can reconstruct the kernels quite accurately from the data: The estimated kernel is very close to the true kernel in most places, although the estimated kernel may diverge from the ground-true in a few points. The proposed algorithm generally works quite well in this case since our method do not assume any parametric forms of the triggering kernels. Another observation is that MMEL tends to underestimate the triggering kernels at their peak points while overestimate them at valleys. In fact, similar bias exists in a lot of nonparametric estimators which is related to the curvature of the function. Several methods has been proposed to correct this problem [131].

6.3.2 Synthetic Data

Data Generation. A relatively large synthetic data set is generated as follows: We consider Hawkes processes of $U = 300$ dimensions with base intensity μ_u sampled from a uniform distribution over $[0, 0.001]$ for each u. The triggering kernels are the linear combinations of three base functions: $g_d(t) = \frac{\cos(2\pi t/w_d)+2}{t+2}$ where $w_d = 1, 3, 5$ for $d = 1, 2, 3$, respectively. The coefficients of the linear combinations are generated from a uniform distribution on $[0, 0.1]$. We generate 200,000 samples from the multi-dimensional Hawkes process as training set and another disjoint 200,000 samples as test set. We run the above process for five times and the performance are reported by the average over five data sets.



(a) LogLik



(b) Diff

Figure 33: Performance measured by LogLik and Diff with respect to the number of training samples on the synthetic data.

Evaluation Metrics. We use two evaluation metrics LogLik and Diff to evaluate the proposed method. Specifically, LogLik is defined as the log-likelihood on the test set of 200,000 samples that is disjoint with the training set. Diff measures the relative distances between the estimated and true kernels as follows:

$$\text{diff} = \frac{1}{U^2}\sum_{u=1}^{U}\sum_{u'=1}^{U}\frac{\int(\hat{g}_{uu'}(t) - g_{uu'}(t))^2 dt}{\int g_{uu'}(t)^2 dt},$$

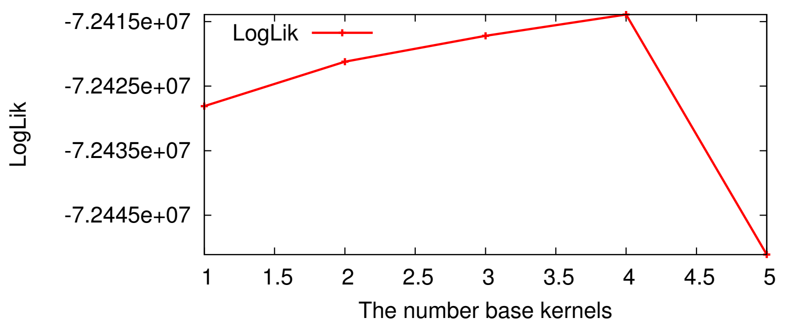


Figure 34: Performance measured by LogLik with respect to number of base kernels D.

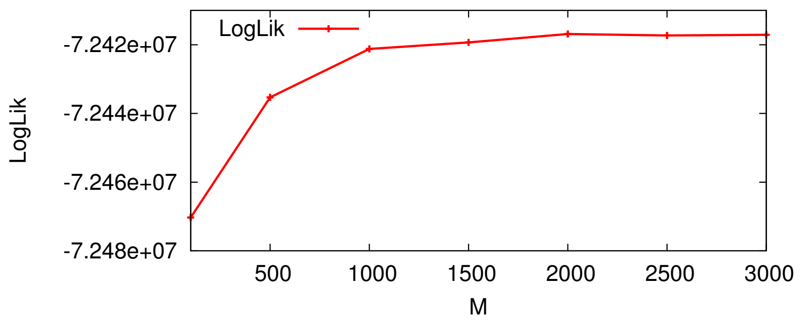


Figure 35: Performance measured by LogLik with respect to point used to discretize the ODE.

where $\hat{g}_{uu'}(t)$ and $g_{uu'}(t)$ are the estimated and true kernels between dimension u and u', respectively.

Baselines. We compare the proposed MMEL with the following baselines to demonstrate its effectiveness:

- ExpKernel. In this method, the triggering kernel is assumed to be fixed to be the exponential kernel $g(t) = \frac{1}{w}\exp(-t/w)$ which is one of the most widely used kernels for multi-dimensional Hawkes process. In this work, we use $w =$

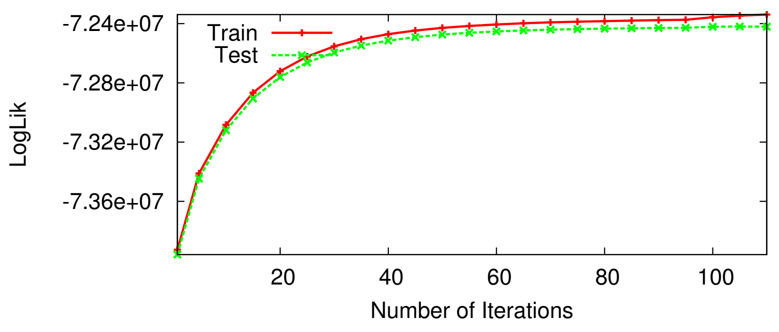


Figure 36: Performance measured by LogLik with respect to number of iterations.

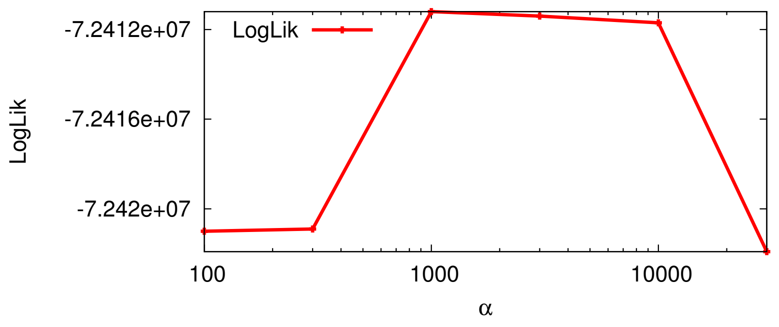


Figure 37: Performance measured by LogLik with respect to regularization parameter α.

$1, 3, 5$ as baselines and label them as ExpKernel-1, ExpKernel-3 and ExpKernel-5, respectively.

- TrueKernel. In this method, we assume that the bases $g_d(t)$ used to generate the data are known and fixed. Only the coefficients are estimated from the data. This method is used as an upper-bound to show that how the proposed MMEL algorithm can perform in the ideal situation that the true bases for the triggering kernels are known.

Results. We train models with MMEL as well as the baselines on the training set. For MMEL, we use regularization parameter $\alpha = 1000$ and discretize the ODE with $M = 3000$ intervals. In Figure 33, we present the performance on the synthetic data set with respect to the number of training data. From Figure 33, we can observe that the performance of all methods improves as the number of training data grows, which indicates that more training data can improve the performance. Comparing the performance of methods using exponential kernels, i.e., ExpKernel-1, ExpKernel-3 and ExpKernel-5, we can observe that the selection of the parameters for the exponential kernel can greatly impact the performance, which confirms that the triggering kernels play a central role in multi-dimensional Hawkes processes.

The proposed method MMEL performs significantly better than the method using exponential kernels with respect to both metrics. Moreover, its performance is very close to TrueKernel, the method that fixes the base kernels to be the ground-truth and does not estimate them from data. From the error bars, the performance MMEL and True Kernel are quite close to each other in terms of LogLik. Therefore, we conclude that the MMEL can estimate the triggering kernels very accurately.

In Figure 34, we present the performance of MMEL measured by LogLik with respect to the number of base kernels. In our previous experiments on the synthetic data set, we set the number of base kernel to be 3, which is the true value used to generate the data. It is interesting to observe that slightly larger number of base kernels such as 4 can archive better performance.

In Figure 35, we investigate the number of intervals M used to discretize the ordinary differential equation in Equation (39). In particular, we vary M in range of

100 to 3000 and plot the performance measured by LogLik. From Figure 35, we can see that when the number of intervals is relatively large (≥ 1000), the performance is quite good and stable.

In Figure 36, we show that the performance measured by LogLik on both training and test sets with respect to the number of iterations of MMEL. It can be observed that the performance on both training and test sets increases as the number of iterations grows and converges after 100 iterations. In Figure 37, we present the performance measured by LogLik on the test set with respect to the value of regularization parameter α. We can observe that small values of α usually reduce the performance, while the performance is quite good for relatively large $\alpha = 1000$ or 10000].

6.3.3 Real World Data

We also evaluate the proposed method on a real world data set. To this end, we use the MemeTracker data set[1]. The data set contains the information flows captured by hyper-links between different sites with timestamps. In particular, we first extract the top 100 popular sites and the links between them. The events are in the form that a site created a hyper-link to another site at a particular time. In particular, each site corresponds to a dimension of the multi-dimensional Hawkes process. The observed events are the propagation of hyper-links through the network. We use 50% data as training data and 50% as test data. In this data set, we use $D = 1$ and $\alpha = 10$.

In Figure 38, we present the performance measured by the negative log-likelihood on test set for MMEL, ExpKernel-1, ExpKernel-3 and ExpKernel-5 on the MemeTracker

[1]`http://memetracker.org`

data set. We can observe that MMEL outperforms all the baselines, which indicates that MMEL can capture the dynamics of the temporal events more accurately. We also tried to fix the kernel to be other forms such as cosine functions. The performance is much worse, which suggests the importance of the triggering kernel. In order to further investigate the performance, we transform the events by the fitted model based on the time rescaling theorem [112], and generate the quantile-quantile (Q-Q) plot with respect to the exponential distribution, since it is the theoretical distribution for the perfect model of intensity functions as shown by the theorem. Generally speaking, Q-Q plot visualizes the goodness-of-fit for different models. The perfect model follows a straight line of $y = x$. In Figure 39, we present the Q-Q plot for MMEL, ExpKernel-1 and Poisson, which is the Poisson process model with constant intensity. We can observe that MMEL are generally closer to the straight line, which suggests that MMEL can fit the data better than other models.

In Figure 40, we plot the base kernel estimated from the data by MMEL. The base kernel has quite intuitive in the sense that in the first several days, the estimated base kernel has relatively large values, which can be explained by the fact that new blogs or webpages are more likely to be related to hot topics and thus are more likely to trigger further discussions. The base has relatively small values at almost all other points, except for two small peaks as we can observe in the figure. We think it reflects the long-term discussions of some topics.

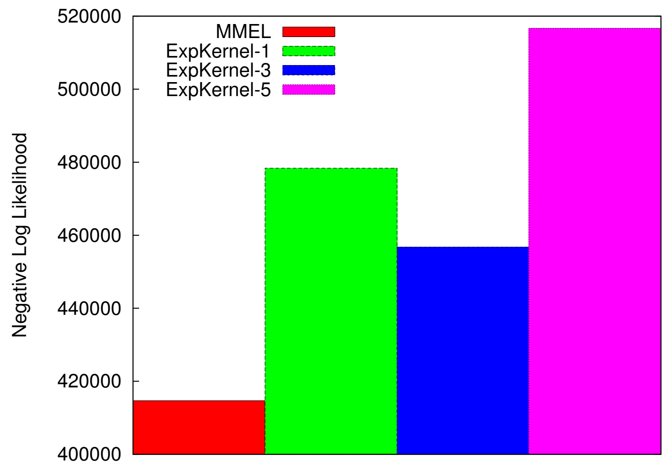


Figure 38: Performance measured by negative log-likelihood on MemeTracker data set.

6.4 Summary

In this chaper, we address the problem of learning the triggering kernels, which capture the underlying temporal dynamics of the observed events, for multi-dimensional Hawkes processes. In particular, we estimate the triggering kernels from the data through optimizing the penalized log-likelihood function in infinite dimensional spaces. An iterative algorithm MMEL is proposed to optimized the penalized log-likelihood function effectively. The optimization problem in infinite dimensional functional space is transformed to solving an ordinary differential equation which is derived from the Euler-Lagrange equation. Experimental results on both synthesis and real-world data set show that the proposed method can estimate the triggering kernels more accurately and thus provide better models for recurrent events.

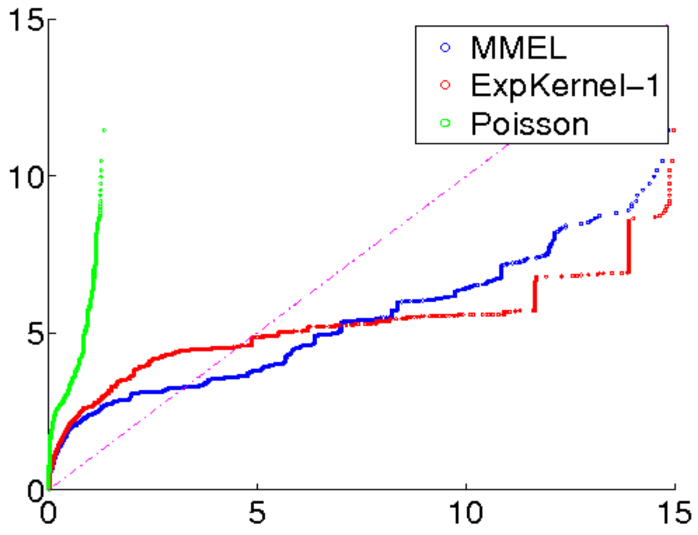


Figure 39: Q-Q plot for comparing the transformed events with respect to the exponential distribution.

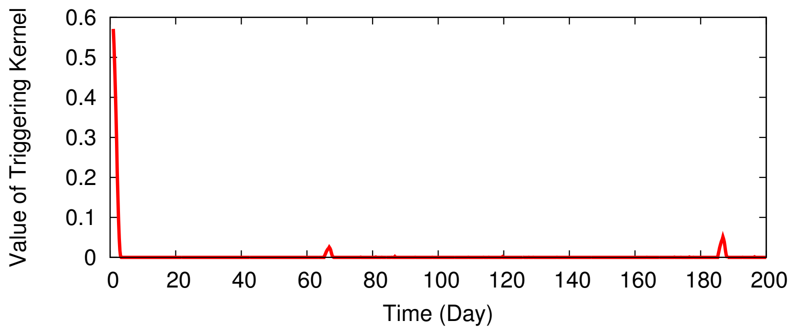


Figure 40: The base kernel estimated from the MemeTracker data set.

CHAPTER VII

CONCLUSIONS AND DISCUSSIONS

7.1 Summary

In this thesis, we investigate several applications and extensions of the ideas based on low-rank matrix factorizations to solve the problems from real-world applications. In particular, we adapt and extend the matrix factorizations to incorporate the requirements from the specific application domains. We introduce structured latent factors into traditional low-rank matrix factorizations by restricting the latent factors to be decision trees, binary codes and functions. We also investigate the problem of optimizing the complex loss function introduced by modeling temporal events in social networks.

We briefly summary the contributions of this thesis as follows:

- **Cold-start recommendation.** We extend the low-rank matrix factorization by allowing the latent factors to be decision trees to solve the problem of cold-start recommendations. In particular, we present functional matrix factorization (fMF), which combines the ideas of the decision tree for adaptive interview and latent factors for matrix factorization in a unified framework.

- **Binary coding for collaborative filtering.** We solve the problem of making recommendations efficiently by constructing binary codes for collaborative filtering. In particular, we restrict the latent factors for users and items to be binary vectors so that the recommendation can be performed by searching in the Hamming space.

- **Learning social infectivity from temporal events.** We proposed a framework to combine the estimation of mutually exciting process and low-rank matrix factorization. Specifically, we propose a convex optimization approach to estimate the hidden social influence, which is both low-rank and sparse, from recurrent events modeled by multi-dimensional Hakwes process.

- **Learning nonparametric kernels for multi-dimensional Hawkes process.** We estimate the triggering kernels from an infinite dimensional functional space. The estimation process makes use of the Euler Lagrange equation to derive the ordinary differential equation that the optimal kernel should satisfy. Moreover, the method can be viewed as applying the idea of low-rank factorization in the functional space.

In summary, low-rank matrix factorizations provide a flexible and effective framework to capture the correlations in high dimensional spaces in a wide range of applications with a diverse set of requirements.

7.2 Discussions and Future Directions

The current fMF model is based on a basic matrix factorization formulation and does not take into account the content features such as the demographical information of users. For future work, we can investigate how to utilize such content features to further enhance the performance of fMF in cold-start recommendations.

For learning binary codes, we can investigate other methods for solving the discrete problem more accurately. Specifically, we can investigate how to apply semidefinite programming for relaxing the original problem. Another direction is to study how to learn binary codes incrementally. In particular, we would like to construct the binary codes bit by bit in a sequential and incremental manner. It has the advantage that new bits of binary codes can be introduced to improve the accuracy without re-training all of the bits. Finally, the problem of incorporating features for users and items, such as demographical features for users and descriptions of items, to learn the binary codes can be also very interesting.

For all the discussed models, it is important to design efficient algorithms over parallel and distributed computing environments, such as MPI, Hadoop and GraphLab, in order to scale the algorithms to the case where the training data can not fit into a single machine.

REFERENCES

[1] Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P., "Low-rank matrix factorization with attributes," *Arxiv*, p. 12, Nov. 2006.

[2] Adamopoulos, L., "Some counting and interval properties of the mutually-exciting processes," *Journal of Applied Probability*, vol. 12, p. 78, Mar. 1975.

[3] Adams, R. P., Murray, I., and MacKay, D. J. C., "Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, (New York, New York, USA), pp. 1–8, ACM Press, 2009.

[4] Adomavicius, G. and Tuzhilin, a., "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

[5] Agarwal, D., Agrawal, R., Khanna, R., and Kota, N., "Estimating rates of rare events with multiple hierarchies through scalable log-linear models," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 213–222, ACM, 2010.

[6] Agarwal, D., Chen, B., and Elango, P., "Fast online learning through offline initialization for time-sensitive recommendation," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 703–712, ACM, 2010.

[7] Agarwal, D. and Chen, B.-C., "Regression-based latent factor models," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 19, 2009.

[8] Agarwal, D. and Chen, B.-C., "fLDA," in *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*, (New York, New York, USA), p. 91, ACM Press, 2010.

[9] Ahn, S., "Nearest Neighbor Method in Learning and Vision," vol. 1, pp. 93–7, July 2011.

[10] Argyriou, A., Hauser, R., Micchelli, C. A., and Pontil, M., "A DC-programming algorithm for kernel selection," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, (New York, New York, USA), pp. 41–48, ACM Press, 2006.

[11] Argyriou, A., Micchelli, C., Pontil, M., and Ying, Y., "A spectral regularization framework for multi-task structure learning," *Advances in Neural Information Processing Systems*, vol. 20, pp. 25–32, 2008.

[12] Bach, F., "Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning," *Advances in Neural Information Processing Systems (NIPS)*, vol. 21, no. 2, pp. 105–112, 2008.

[13] Bach, F., Mairal, J., and Ponce, J., "Convex Sparse Matrix Factorizations," *CoRR, abs/0812.1869*, pp. 1–12, Dec. 2008.

[14] Bachrach, Y. and Herbrich, R., "Fingerprinting Ratings For Collaborative Filtering Theoretical and Empirical Analysis," *String Processing and Information Retrieval*, pp. 25–36, 2010.

[15] Bachrach, Y., Porat, E., and Rosenschein, J., "Sketching techniques for collaborative filtering," in *Proceedings of the 21st international joint conference on Artifical intelligence*, pp. 2016–2021, Morgan Kaufmann Publishers Inc., 2009.

[16] Bacry, E., Dayri, K., and Muzy, J. F., "Non-parametric kernel estimation for symmetric Hawkes processes. Application to high frequency financial data," *The European Physical Journal B*, vol. 85, May 2012.

[17] Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Cortes, C., and Mohri, M., "Polynomial semantic indexing," *Advances in neural information processing systems (NIPS 2009)*, no. 1, pp. 1–9, 2009.

[18] Baluja, S. and Covell, M., "Learning Forgiving Hash Functions : Algorithms and Large Scale Tests," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[19] Banerjee, A. V., "A Simple Model of Herd Behavior," *The Quarterly Journal of Economics*, vol. 107, pp. 797–817, Aug. 1992.

[20] BARZILAI, J. and BORWEIN, J. M., "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, vol. 8, pp. 141–148, Jan. 1988.

[21] Basilico, J. and Hofmann, T., "Unifying collaborative and content-based filtering," *Twenty-first international conference on Machine learning - ICML '04*, p. 9, 2004.

[22] Bertsekas, D. P., *Nonlinear Programming.* Athena Scientific, 2nd ed., Sept. 1999.

[23] Blundell, C., Heller, K., and Beck, J., "Modelling reciprocating relationships with Hawkes processes," *NIPS*, 2012.

[24] Boutilier, C., Zemel, R., and Marlin, B., "Active collaborative filtering," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 98–106, Citeseer, 2003.

[25] Boyd, S., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[26] Breese, J., Heckerman, D., Kadie, C., and Others, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[27] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks, 1984.

[28] Candes, E., Li, X., Ma, Y., and Wright, J., "Robust principal component analysis," *preprint*, 2009.

[29] Candes, E. J. and Plan, Y., "Matrix Completion With Noise," *Arxiv*, vol. 98, p. 11, Mar. 2009.

[30] Cassandra, A. R., *Exact and approximate algorithms for partially observable markov decision processes.* PhD thesis, Providence, RI, USA, 1998. AAI9830418.

[31] Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S., "Rank-Sparsity Incoherence for Matrix Decomposition," *Arxiv*, vol. 02139, pp. 1–24, June 2009.

[32] Chen, J., Zhou, J., and Ye, J., "Integrating low-rank and group-sparse structures for robust multi-task learning," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, (New York, NY, USA), pp. 42–50, ACM, 2011.

[33] Cortes, C., Mohri, M., and Rastogi, A., "Magnitude-preserving ranking algorithms," in *Proceedings of the 24th international conference on Machine learning*, ICML '07, (New York, NY, USA), pp. 169–176, ACM, 2007.

[34] Cortes, C., Mohri, M., and Rostamizadeh, A., "Ensembles of Kernel Predictors," Feb. 2012.

[35] Daley, D. J. and Vere-Jones, D., *An introduction to the theory of point processes.* Springer, 2008.

[36] Das, A. S., Datar, M., Garg, A., and Rajaram, S., "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web - WWW '07*, (New York, New York, USA), p. 271, ACM Press, 2007.

[37] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S., "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*, (New York, New York, USA), p. 253, ACM Press, 2004.

[38] Davis, J. and Dhillon, I., "Structured metric learning for high dimensional problems," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 195–203, ACM, 2008.

[39] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, Sept. 1990.

[40] Dinuzzo, F., Ong, C., Gehler, P., and Pillonetto., G., "Learning output kernels with block coordinate descent," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[41] Du, N., Song, L., Smola, A., and Yuan, M., "Learning Networks of Heterogeneous Inuence," *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[42] Eckart, C. and Young, G., "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, Sept. 1936.

[43] Eckstein, J. and Bertsekas, D. P., "On the Douglas Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, pp. 293–318, Apr. 1992.

[44] Eriksson, A. and Hengel, A. V. D., "Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm," *Computer Vision and Pattern*, no. 2, 2010.

[45] Fazel, M., Hindi, H., and Boyd, S., "A rank minimization heuristic with application to minimum order system approximation," in *Proceedings of the 2001 American Control Conference*, vol. 6, pp. 4734–4739, IEEE, 2002.

[46] Foster, J. G., Foster, D. V., Grassberger, P., and Paczuski, M., "Edge direction and the structure of networks.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 10815–20, June 2010.

[47] Fowlkes, C., Belongie, S., Chung, F., and Malik, J., "Spectral grouping using the Nyström method.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, pp. 214–25, Feb. 2004.

[48] Friedman, J., Hastie, T., and Tibshirani, R., "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[49] Gabay, D. and Mercier, B., "A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation," *Computers & Mathematics with Applications*, vol. 2, pp. 17–40, Jan. 1976.

[50] Golbandi, N., Koren, Y., and Lempel, R., "On bootstrapping recommender systems," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1805–1808, ACM, 2010.

[51] Golbandi, N., Koren, Y., and Lempel, R., "Adaptive Bootstrapping of Recommender Systems Using Decision Trees," *WSDM*, 2011.

[52] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C., "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[53] Gomez Rodriguez, M., Leskovec, J., and Krause, A., "Inferring networks of diffusion and influence," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, (New York, New York, USA), p. 1019, ACM Press, 2010.

[54] Goyal, A., Bonchi, F., and Lakshmanan, L. V., "Learning influence probabilities in social networks," in *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*, (New York, New York, USA), p. 241, ACM Press, 2010.

[55] Guan, Y., "A Composite Likelihood Approach in Fitting Spatial Point Process Models," *Journal of the American Statistical Association*, vol. 101, pp. 1502–1512, Dec. 2006.

[56] Gunawardana, A. and Meek, C., "Tied boltzmann machines for cold start recommendations," in *Proceedings of the 2008 ACM conference on Recommender systems*, (New York, New York, USA), pp. 19–26, ACM, 2008.

[57] Halko, N., Martinsson, P.-G., and Tropp, J. A., "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," pp. 1–81, Sept. 2009.

[58] Harpale, A. and Yang, Y., "Personalized active learning for collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 91–98, ACM, 2008.

[59] Hawkes, A. G., "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[60] He, Q., Pei, J., Kifer, D., Mitra, P., and Giles, L., "Context-aware citation recommendation," in *Proceedings of the 19th international conference on World wide web - WWW '10*, (New York, New York, USA), p. 421, ACM Press, 2010.

[61] Hofmann, T., "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, pp. 89–115, Jan. 2004.

[62] Hoi, S. C. H., Jin, R., and Lyu, M. R., "Learning nonparametric kernel matrices from pairwise constraints," in *Proceedings of the 24th international conference on Machine learning - ICML '07*, (New York, New York, USA), pp. 361–368, ACM Press, 2007.

[63] Huang, Y., Tresp, V., and Kriegel, H., "Multivariate Prediction for Learning in Relational Graphs," *NIPS 2009 Workshop: Analyzing Networks and Learning With Graphs*, pp. 1–8, 2009.

[64] Hunter, D. R. and Lange, K., "A tutorial on MM algorithms," *The American Statistician*, no. 58, 2004.

[65] Indyk, P., "Locality-sensitive hashing using stable distributions,"

[66] Indyk, P. and Motwani, R., "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, (New York, New York, USA), pp. 604–613, ACM Press, 1998.

[67] Jaggi, M. and Sulovsk\'y, M., "A Simple Algorithm for Nuclear Norm Regularized Problems," *Proceedings of the 27th International Conference on Machine Learning*, no. X, 2010.

[68] Jain, P., Kulis, B., Davis, J. V., and Dhillon, I. S., "Metric and Kernel Learning using a Linear Transformation," *Arxiv preprint arXiv:0910.5932*, Oct. 2009.

[69] Jégou, H., Furon, T., and Fuchs, J.-J., "Anti-sparse coding for approximate nearest neighbor search," *Arxiv preprint arXiv:1110.3767*, pp. 577–580, Oct. 2011.

[70] Jenatton, R., Obozinski, G., and Bach, F., "Structured Sparse Principal Component Analysis," *AISTATS*, vol. 9, 2010.

[71] Ji, S. and Ye, J., "An accelerated gradient method for trace norm minimization," *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[72] Jin, R. and Si, L., "A Bayesian approach toward active learning for collaborative filtering," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 278–285, AUAI Press, 2004.

[73] Johannes, M. and Polson, N., *Handbook of Financial Time Series.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[74] Joly, A. and Buisson, O., "Random maximum margin hashing," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 873–880, IEEE, 2011.

[75] Kekäläinen, J., "Binary and graded relevance in IR evaluations - Comparison of the effects on ranking of IR systems," *Information Processing and Management*, vol. 41, pp. 1019–1033, 2005.

[76] Keshavan, R. H. and Montanari, A., "Regularization for Matrix Completion," *Arxiv*, p. 5, Jan. 2009.

[77] Keshavan, R. H., Montanari, A., and Oh, S., "Matrix Completion from a Few Entries," *IEEE Transactions on Information Theory*, vol. 56, p. 30, Jan. 2009.

[78] Koren, Y., "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, pp. 1–24, 2010.

[79] Kulis, B. and Grauman, K., "Kernelized locality-sensitive hashing for scalable image search," in *Computer Vision, 2009 IEEE 12th International Conference on*, no. Iccv, pp. 2130–2137, IEEE, Sept. 2009.

[80] Kulis, B. and Darrell, T., "Learning to hash with binary reconstructive embeddings," *Proceedings of Advances in Neural Information Processing Systems*, 2009.

[81] Lawrence, N. and Urtasun, R., "Non-linear matrix factorization with gaussian processes," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 601–608, ACM, 2009.

[82] Lee, D. and Seung, H., "Algorithms for non-negative matrix factorization," *NIPS*, vol. 13, 2000.

[83] Leskovec, J., Backstrom, L., and Kleinberg, J., "Meme-tracking and the dynamics of the news cycle," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 497, 2009.

[84] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N., "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, (New York, New York, USA), p. 420, ACM Press, 2007.

[85] Lewis, E., Mohler, G., Brantingham, P. J., and Bertozzi, A. L., "Self-exciting point process models of civilian deaths in Iraq," *Security Journal*, vol. 25, pp. 244–264, Sept. 2011.

[86] Lewisa, E. and Mohlerb, G., "A Nonparametric EM algorithm for multiscale Hawkes processes," *Journal of Nonparametric Statistics*, no. 1, 2011.

[87] Li, P., Shrivastava, A., Moore, J., and Konig, A. C., "Hashing Algorithms for Large-Scale Learning," *Arxiv preprint arXiv:1106.0967*, pp. 1–9, June 2011.

[88] Li, W. and Yeung, D., "Relation regularized matrix factorization," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 1126–1131, 2009.

[89] Lin, Y.-R., Sun, J., Castro, P., Konuru, R., Sundaram, H., and Kelliher, A., "MetaFac: Community Discovery via Relational Hypergraph Factorization," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, (New York, New York, USA), p. 527, ACM Press, 2009.

[90] Liniger, T. J., *Multivariate Hawkes Processes.* PhD thesis, Swiss Federal Institute Of Technology Zurich, 2009.

[91] Liu, W., Wang, J., Kumar, S., and Chang, S., "Hashing with Graphs," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[92] Long, B., Wu, X., Zhang, Z. M., and Yu, P. S., "Unsupervised learning on k-partite graphs," in *KDD*, pp. 317–326, 2006.

[93] Long, B., Zhang, Z. M., U, W., X., Y., and P.s, "Spectral clustering for multi-type relational data," in *ICML*, vol. pp, pp. 585–592, 2006.

[94] Long, B., Zhang, Z. M., and Yu, P. S., "Probabilistic framework for relational clustering," in *KDD*, vol. pp, pp. 470–479, 2007.

[95] Long, B., Zhang, Z., Wu, X., and Yu, P., "Relational clustering by symmetric convex coding," in *Proceedings of the 24th international conference on Machine learning*, pp. 569–576, ACM, 2007.

[96] Mammen, E. and Thomas-agnan, C., "Smoothing Splines And Shape Restrictions," *Scandinavian Journal of Statistics*, vol. 26, pp. 239–251, 1998.

[97] Marsan, D. and Lengliné, O., "Extending earthquakes' reach through cascading," *Science (New York, N.Y.)*, vol. 319, pp. 1076–9, Feb. 2008.

[98] Martinsson, G., "Randomization : Making Very Large-Scale Linear Algebraic Computations Possible."

[99] Masuda, N., Takaguchi, T., Sato, N., and Yano, K., "Self-exciting point process modeling of conversation event sequences," May 2012.

[100] Mazumder, R., Hastie, T., and Tibshirani, R., "Spectral regularization algorithms for learning large incomplete matrices," *JMLR*, 2009.

[101] McPherson, M., Smith-Lovin, L., and Cook, J. M., "Birds of a Feather: Homophily in Social Networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, Aug. 2001.

[102] Meka, R., Jain, P., and Dhillon, I., "Matrix Completion from Power-Law Distributed," *NIPS*, 2009.

[103] Meyer, M. C., "Inference using shape-restricted regression splines," *The Annals of Applied Statistics*, vol. 2, pp. 1013–1033, Sept. 2008.

[104] Mitchell, L. and Cates, M. E., "Hawkes process as a model of social interactions: a view on video dynamics," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, p. 045101, Jan. 2010.

[105] Moallemi, C. C. and Roy, B. V., "Convergence of Min-Sum Message-Passing for Convex Optimization," *IEEE Trans. on Information Theory*, vol. 56, no. 4, pp. 2041—-2050, 2010.

[106] Mohler, G. O., Bertozzi, A. L., Goldstein, T. a., and Osher, S. J., "Fast TV Regularization for 2D Maximum Penalized Likelihood Estimation," *Journal of Computational and Graphical Statistics*, vol. 20, pp. 479–491, Jan. 2011.

[107] Myers, S. and Leskovec, J., "On the convexity of latent social network inference," *NIPS*, 2010.

[108] Newman, M. E. J. and Leicht, E. a., "Mixture models and exploratory analysis in networks.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, pp. 9564–9, June 2007.

[109] Norouzi, M. and Fleet, D., "Minimal Loss Hashing for Compact Binary Codes," in *Proceedings of the 28th International Conference on Machine Learning*, vol. 1, 2011.

[110] Okatani, T. and Deguchi, K., "On the Wiberg Algorithm for Matrix Factorization in the Presence of Missing Components," *International Journal of Computer Vision*, vol. 72, pp. 329–337, Sept. 2006.

[111] Pan, W., Cebrian, M., Dong, W., Kim, T., Fowler, J., and Pentland, A., "Modeling dynamical influence in human interaction patterns," *Arxiv preprint arXiv:1009.0240*, p. 19, Sept. 2010.

[112] Papangelou, F., "Integrability of Expected Increments of Point Processes and a Related Random Change of Scale," *Transactions of the American Mathematical Society*, vol. 165, p. 483, Mar. 1972.

[113] Park, S.-T. and Chu, W., "Pairwise preference regression for cold-start recommendation," *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, p. 21, 2009.

[114] Paterek, A., "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD Cup and Workshop*, vol. 2007, pp. 5–8, 2007.

[115] Pennock, D., Horvitz, E., Lawrence, S., and Giles, C., "Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach," in *Proceedings of the 16th conference on uncertainty in artificial intelligence*, pp. 473–480, 2000.

[116] Porteous, I., Bart, E., and Welling, M., "Multi-hdp: A non parametric bayesian model for tensor factorization," in *AAAI*, pp. 1487–1490, 2008.

[117] Pu, P. and Chen, L., "User-involved preference elicitation for product search and recommender systems," *AI Magazine*, vol. 29, no. 4, p. 93, 2009.

[118] Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 1994.

[119] Raginsky, M., "Locality-sensitive binary codes from shift-invariant kernels," *The Neural Information Processing*, 2009.

[120] Rakhlin, A., Abernethy, J., and Bartlett, P. L., "Online discovery of similarity mappings," *Proceedings of the 24th International Conference on Machine Learning (2007)*, pp. 767–774, 2007.

[121] Ramlau-Hansen, H., "Smoothing Counting Process Intensities by Means of Kernel Functions," *The Annals of Statistics*, vol. 11, pp. 453–466, June 1983.

[122] Rashid, A. M., Karypis, G., and Riedl, J., "Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach," *SIGKDD Workshop on Web Mining and Web Usage Analysis*, vol. 22, Oct. 2008.

[123] Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J., and Riedl, J., "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*, pp. 127–134, ACM, 2002.

[124] Reinsch, C. h., "Smoothing by Spline Functions," *Numerische Mathematik*, pp. 177–183, 1967.

[125] Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L., "Factorizing personalized Markov chains for next-basket recommendation," *Proceedings of the 19th international conference on World wide web - WWW '10*, p. 811, 2010.

[126] Rennie, J. and Srebro, N., "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*, pp. 713–719, ACM, 2005.

[127] Rettinger, A., Nickles, M., and Tresp, V., "Statistical Relational Learning with Formal Ontologies," *ECML PKDD*, no. 1, pp. 286–301, 2009.

[128] Richard, E., Savalle, P.-A., and Vayatis, N., "Estimation of simultaneously sparse and low rank matrices," *ICML*, June 2012.

[129] Rodriguez, M. G., Balduzzi, D., and Schölkopf, B., "Uncovering the temporal dynamics of diffusion networks," *Proceedings of the 28th International Conference on Machine Learning*, pp. 561–568, May 2011.

[130] Saeedi, A. and Bouchard-Côté, A., "Priors over recurrent continuous time processes," *NIPS*, pp. 1–9, 2011.

[131] Sain, S. R. and Scott, D. W., "Zero-bias locally adaptive density estimators," *Scandinavian Journal of Statistics*, vol. 29, pp. 441–460, Nov. 2002.

[132] Salakhutdinov, R. and Hinton, G., "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, pp. 969–978, July 2009.

[133] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M., "Methods and metrics for cold-start recommendations," *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, p. 253, 2002.

[134] Schmidt, M., *Graphical Model Structure Learning with L1-Regularization.* PhD thesis, 2010.

[135] Schubert, M. and Kriegel, H.-p., "Relation Prediction in Multi-Relational Domains using Matrix Factorization," *NIPS2008 Workshop on Structured Input, Structured Output*, no. Siso, 2008.

[136] Shakhnarovich, G., Viola, P., and Darrell, T., "Fast pose estimation with parameter-sensitive hashing," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 750–757 vol.2, IEEE, 2003.

[137] Shawe-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis.* New York, NY, USA: Cambridge University Press, 2004.

[138] Simma, A. and Jordan, M., "Modeling events with cascades of Poisson processes," *UAI*, 2012.

[139] Singh, A. and Gordon, G., "A unified view of matrix factorization models," *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pp. 358—-373, 2008.

[140] Singh, A. and Gordon, G., "Relational learning via collective matrix factorization," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 650–658, ACM, 2008.

[141] Sonnenburg, S., Raetsch, G., Schaefer, C., and Schölkopf, B., "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1531–1565, 2006.

[142] Srebro, N., Rennie, J., and Jaakkola, T., "Maximum-margin matrix factorization," *Advances in neural information processing systems*, vol. 17, pp. 1329–1336, 2005.

[143] Srebro, N., "Learning with matrix factorizations," *Thesis*, 2004.

[144] Stern, D., Herbrich, R., and Graepel, T., "Matchbox: large scale online bayesian recommendations," in *Proceedings of the 18th international conference on World wide web*, pp. 111–120, ACM, 2009.

[145] Stimberg, F., Opper, M., Ruttor, A., and Sanguinetti, G., "Inference in continuous-time change-point models," *NIPS*, pp. 1–9, 2011.

[146] Stomakhin, A., Short, M. B., and Bertozzi, A. L., "Reconstruction of missing data in social networks based on temporal patterns of interactions," *Inverse Problems*, vol. 27, p. 115013, Nov. 2011.

[147] Strecha, C., Bronstein, A., Bronstein, M., and Fua, P., "LDAHash: Improved Matching with Smaller Descriptors.," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1–14, May 2011.

[148] Su, X. and Khoshgoftaar, T. M., "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, no. Section 3, pp. 1–20, 2009.

[149] Sutskever, I., Salakhutdinov, R., and Tenenbaum, J., "Modelling relational data using Bayesian clustered tensor factorization," *Advances in Neural Information Processing Systems*, vol. 22, pp. 1–8, 2009.

[150] Taddy, M. A. and Kottas, A., "Mixture modeling for marked poisson processes," 2010.

[151] Toke, I. M., ""Market making" behaviour in an order book model and its impact on the bid-ask spread," *Arxiv*, p. 17, Mar. 2010.

[152] Toke, I. M., "An Introduction to Hawkes Processes with Applications to Finance," 2011.

[153] Torralba, A., Fergus, R., and Weiss, Y., "Small codes and large image databases for recognition," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008.

[154] Truccolo, W. and Donoghue, J. P., "Nonparametric modeling of neural point processes via stochastic gradient boosting regression.," *Neural computation*, vol. 19, pp. 672–705, Mar. 2007.

[155] Turlach, B. A., *Constrained Smoothing Splines Revisited.* PhD thesis, Australian National University, 1997.

[156] Vere-Jones, D., "Stochastic Models for Earthquake Occurrence," *Journal of the Royal Statistical Society*, vol. 32, no. 1, pp. 1–62, 1970.

[157] Vu, D. and Asuncion, A., "Dynamic egocentric models for citation networks," *Proc. 28th Intl. Conf. on . . .*, 2011.

[158] Wahba, G., *Spline models for observational data.* Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1990.

[159] Wang, H., Lu, Y., and Zhai, C., "Latent aspect rating analysis on review text data: a rating regression approach," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 783–792, ACM, 2010.

[160] Wang, Q., Xu, J., and Li, H., "Regularized Latent Semantic Indexing," *SIGIR*, 2011.

[161] Weiss, Y., Torralba, A., and Fergus, R., "Spectral hashing," in *Neural Information Processing Systems*, no. 1, pp. 1–8, 2008.

[162] Xu, Z., Kersting, K., and Tresp, V., "Multi-relational learning with gaussian processes," in *IJCAI*, vol. 9, pp. 1309–1314, 2009.

[163] Yu, K., "A Note on Inverted Wishart Distribution," Tech. Rep. 2, 1999.

[164] Yu, K., Lafferty, J., Zhu, S., and Gong, Y., "Large-scale collaborative prediction using a nonparametric random effects model," in *Proceedings of the 26th Annual International Conference on Machine Learning*, no. 1, pp. 1185–1192, ACM, 2009.

[165] Yu, K., Zhu, S., Lafferty, J., and Gong, Y., "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 211–218, ACM, 2009.

[166] Yu, S. X. and Shi, J., "Multiclass spectral clustering," in *Proceedings Ninth IEEE International Conference on Computer Vision*, no. 0, (Washington, DC, USA), pp. 313–319 vol.1, IEEE, 2003.

[167] Zha, H. and Zhang, Z., "On Matrices with Low-rank-plus-shift Structures: Partial SVD and Latent Semantic Indexing," *SIAM Journal Matrix Analysis and Applications*, vol. 21, pp. 522–536, 1999.

[168] Zhang, D., Wang, J., Cai, D., and Lu, J., "Self-taught hashing for fast similarity search," in *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 18–25, ACM, 2010.

[169] Zhang, D., Wang, J., Cai, D., and Lu, J., "Extensions to Self-Taught Hashing: Kernelisation and Supervision," *SIGIR Workshop*, vol. 29, p. 38, 2010.

[170] Zhang, D., Wang, J., Cai, D., and Lu, J., "Laplacian co-hashing of terms and documents," *Advances in Information Retrieval*, pp. 577–580, 2010.

[171] Zhao, K. and Zhang, Z., "Successively alternate least square for low-rank matrix factorization with bounded missing data," *Computer Vision and Image Understanding*, 2010.

[172] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, pp. 550–560, Dec. 1997.

[173] Zhu, X., Kandola, J. S., Ghahramani, Z., and Lafferty, J. D., "Non-parametric Transforms of Graph Kernels for Semi-Supervised Learning," *Advances in Neural Information Processing Systems (NIPS)*, 2004.

VITA

Ke Zhou was born in Wuhan, China in 1984. He received his MS and BS degree in computer science from Shanghai Jiao-Tong University. He received his PhD degree from School of Computational Science and Engineering in Georgia Tech in 2013.

His research interest includes machine learning and its applications in web search, collaborative filtering and social network. In particular, his PhD research focus on extending matrix factorization models to solve problems from real-world applications.

He has hold intern positions in Microsoft and Twitter in 2011 and 2012, working on web search relevance and online advertising, respectively.

PUBLICATIONS

[1] Ke Zhou, Hongyuan Zha and Le Song. Learning Triggering Kernels for Multi-dimensional Hawkes Processes. *ICML 2013.*

[2] Ke Zhou, Hongyuan Zha and Le Song. Learning Social Infectivity in Sparse Low-rank Networks Using Multi-dimensional Hawkes Processes. *AISTATS 2013.*

[3] Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon and Hongyuan Zha. Learning Multiple-Question Decision Trees for Cold-Start Recommendation. *ACM WSDM 2013.*

[4] Ke Zhou and Hongyuan Zha. Learning Binary Codes for Collaborative Filtering, *ACM SIGKDD 2012.*

[5] Ke Zhou, Xin Li and Hongyuan Zha. Collaborative Ranking: Improving the Relevance for Tail Queries, *CIKM 2012.*

[6] Ke Zhou, Shuang-Hong Yang and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation, *SIGIR 2011.*

[7] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Video Summarization via Transferrable Structured Learning, *WWW 2011.*

[8] Shihao Ji, Ke Zhou, Ciya Liao, Zhaohui Zheng, Gui-Rong Xue, Chapelle Olivier, Gordon Sun , and Hongyuan Zha. Global Ranking by Exploiting User Clicks, *ACM SIGIR 2009.*

[9] Jing Bai, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, Gordon Sun, Belle L. Tseng, Zhaohui Zheng, Yi Chang. Multi-task learning for learning to rank in web search, *ACM CIKM 2009.*

[10] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha and Yong Yu. Enhancing Diversity, Coverage and Balance for Summarization through Structure Learning, *ACM WWW 2009.*

[11] Ke Zhou, Gui-Rong Xue, Hongyuan Zha and Yong Yu. Learning to Rank with Ties, *ACM SIGIR 2008.*