A Unified Approach to Collaborative Filtering via Linear Models and Beyond

Suvash Sedhain

A thesis submitted for the degree of
Doctor of Philosophy of
The Australian National University

June 2017

Supervisor

Scott Sanner
Assistant Professor, University Of Toronto
Toronto, Canada.

Co-Supervisor

Aditya Krishna Menon
Senior Researcher, DATA61
Adjunct Assistant Professor, The Australian National University
Canberra ACT, Australia

Advisor

Lexing Xie
Associate Professor, The Australian National University
Contributed Researcher, DATA61
Canberra ACT, Australia

Declaration

I hereby declare that this thesis is my original work which has been done in collaboration with other researchers. This document has not been submitted for any other degree or award in any other university or educational institution. The following papers are accepted for publication in peer reviewed conference proceedings listed in a reverse chronological order:

- **(Chapter 4)** S. Sedhain, A. Menon, S. Sanner, L. Xie, LoCo: Social Cold-Start Recommendation via Low-Rank Regression. In Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI-16), San Francisco, USA.
- **(Chapter 5)** S. Sedhain, H. Bui, J. Kawale, N. Vlassis, B. Kveton, A. Menon, S. Sanner, T. Bui Practical Linear Models for Large-Scale One-Class Collaborative Filtering. In Proceedings of 25th International Joint Conference on Artificial Intelligence (IJCAI-16), NY, USA.
- **(Chapter 5)** S. Sedhain, A. Menon, S. Sanner, D. Braziunas, On the Effectiveness of Linear Models for One-Class Collaborative Filtering. In Proceedings of 30th AAAI Conference on Artificial Intelligence (AAAI-16), Phoenix, USA.
- **(Chapter 6)** S. Sedhain, A. Menon, S. Sanner, and L. Xie, AutoRec: Autoencoders Meet Collaborative Filtering. In Proceedings of the 24th International World Wide Web Conference (WWW-15), Florence, Italy.
- **(Chapter 4)** S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen, Social Collaborative Filtering for Cold-start Recommendations. In Proceedings of the ACM Conference on Recommender Systems (RecSys-14). Silicon Valley, USA.
- **(Chapter 3)** S. Sedhain, S. Sanner, L. Xie, R. Kidd, K.-N. Tran, and P. Christen Social Affinity Filtering: Recommendation through Fine-grained Analysis of User Interactions and Activities. In Proceedings of the ACM Conference on Online Social Networks (COSN-13). Boston, USA.

Suvash Sedhain
14 June 2017

To my aama and buwa.

Acknowledgments

I would like to express my sincere gratitude to everyone who made this thesis possible.

First of all, I would like to express my deepest gratitude to my supervisor, **Dr. Scott Sanner**, for all his guidance, patience and support throughout my doctoral study. Scott's immense knowledge, enthusiasm, and inspiration always helped me to stay motivated. I am grateful to him for advising not only as a doctoral supervisor but also as a good friend during the hard times. In short, I couldn't have imagined having a better supervisor for my Ph.D.

I would like to thank my co-supervisor, **Dr. Aditya Krishna Menon**, for helping me to grow as a researcher. His in-depth knowledge and remarkable ability to explain things in a simplest possible way made research fun for me. He has been a great mentor and a friend with whom I could discuss topics ranging from science to Game of Thrones. Further, I would like to thank my co-supervisor, **Dr. Lexing Xie**, for her immense support, encouragement, and helpful suggestions. I would like to express my sincere gratitude for her generosity in providing me the computational resources without which I couldn't have done my research.

I acknowledge the financial, academic and technical support provided by the National ICT of Australia (NICTA) and the Australian National University (ANU) for my doctoral program.

I am indebted to my colleagues at NICTA and ANU for providing a friendly environment that allowed me to grow as a person. Thank you for your stimulating discussions and support through the entire process. My special thanks to Trung Nguyen, Kar Wai Lim, Fazlul Hasan Siddiqui, Shamin Kinathil, Swapnil Mishra, Hadi Mohasel Afshar, Shahin Namin, Dana Pordel, Brendan Van Rooyen and Daniel McNamara. I would like to thank my friends Niroj Sapkota, Hira Babu Pradhan and Saroj Gautam for always being there for me.

I would like to thank Darius Braziunas for hosting my internship at Rakuten Kobo, and Jaya Kawale and Hung Bui for hosting my internship at Adobe Research.

I am grateful to my wife Komal Agarwal for her encouragement, unconditional love, and support throughout my PhD. I would like to thank my brother, Surendra Sedhain, and sisters, Bhagabati Sedhain, Suma Sedhain and Shova Sedhain for their love and support. I am also grateful to my niece Aatmiya Silwal and Samipya Mainali for always bringing a smile to my face.

Finally, and most importantly, I would like to thank my parents, Keshav Raj Sharma Sedhain and Chandra Kumari Sedhain. I couldn't have done this without their guidance, encouragement, unfathomable love and support. I dedicate this thesis to them.

Abstract

Recommending a personalised list of items to users is a core task for many online services such as Amazon, Netflix, and Youtube. Recommender systems are the algorithms that facilitate such personalised recommendation. Collaborative filtering (CF), the most popular class of recommendation algorithm, exploits the wisdom of the crowd by predicting users' preferences not only from her past actions but also from the preferences of other like-minded users. In general, it is desirable to have a CF framework that is (1) applicable to wide range of recommendation scenarios, (2) learning-based, (3) amenable to convex optimisation, and (4) scalable. However, all existing CF methods, such as neighbourhood and matrix factorisation, lack one or more of these desiderata.

In this dissertation, we investigate linear models, an under-appreciated but promising area for recommendations that addresses all the above desiderata. We formulate a unified framework based on linear models that yields CF algorithms for four prevalent scenarios. First, we investigate *Social CF*, which involves leveraging users' signals from online social networks. We propose social affinity filtering (SAF), that exploits fine-grained user interactions and activities in a social network. Second, we investigate *Cold-Start CF*, which refers to the scenario when we do not have any historical data about a user or an item. We formulate a large-scale linear model that leverages users social information. Third, we investigate *One-Class CF*, which concerns suggesting relevant items to users from the data that consists of only positive preferences such as item purchase.Noting the superior performance of linear models, we propose LRec, a user-focused linear CF model, and extend it to large-scale datasets via dimensionality reduction. Finally, we investigate *Explicit Feedback CF*, which concerns predicting user's actual preferences such as rating, or like/dislikes. We identify CF as an auto-encoding problem and propose AUTOREC, a generalized neural network architecture for CF. We demonstrate state-of-the-art performance of the proposed models through extensive experimentation on real world datasets.

In a nutshell, this dissertation elucidates the power of linear models for various CF tasks and paves the way for further research on applying deep learning models to CF.

Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{R}$	Real number
$\mathbf{R}$	Partially observed user-item preference matrix
$\lvert\mathbf{R}\rvert$	Number of observed entries in matrix $\mathbf{R}$
$\hat{\mathbf{R}}$	Recommendation matrix
U	Set of users
I	Set of items
$\mathbf{X}^U$	Users' side information
$\mathbf{X}^I$	Items' side information
I_u	Set of items purchased/rated by the user u
U_i	Set of users who purchased/rated the item i
$\mathbf{A}$	User latent factor matrix
$\mathbf{B}$	Item latent factor matrix
$\mathbf{S}$	Similarity matrix

Chapter 1

Introduction

1.1 Background: Personalised recommendation

Over the last decade, the Internet has evolved into a platform for large-scale online services such as Amazon, Netflix, Facebook, eBay, and Youtube. These services has transformed the way we communicate, buy products, watch movies, and listen to music. The number of items [1] offered by online services is unprecedented and ever increasing. As a result, automated systems for item filtering, suggestion and discovery have become very relevant.

Recommender systems are the algorithms that facilitate personalised recommendation by learning users' preferences from a database of their past actions. At a high level, recommender systems are the algorithmic counterpart of a smart personal assistant that finds relevant items on your behalf. On the one hand, recommender systems reduce individual effort in finding relevant items; on the other hand, they add immense business value to the service providers. Specifically, recommender systems help online services to increase their sales [Lee and Hosanagar, 2014] and user engagement. For instance, [Davidson et al., 2010] reported that recommendation accounts for about 60% of total video clicks from Youtube homepage. Hence, recommender systems are at the core of the online services, such as:

- **Entertainment:** Online services, such as Netflix [Gomez-Uribe and Hunt, 2015], Youtube [Davidson et al., 2010], Spotify, etc. extensively use personalised recommendation to deliver relevant content to the users. Netflix reported that recommendations account for 80% of the hours streamed [Gomez-Uribe and Hunt, 2015].

- **E-commerce:** Many online stores, such as Amazon [Linden et al., 2003], eBay [Zhang and Pennacchiotti, 2013], use recommender systems to help customers to find relevant items. By exposing users to relevant items, recommender systems not only attract potential buyers but also convert site surfers to buyers.

- **Social networks:** Recommender systems are widely used in online social networks to help users in exploring new social connections and relevant contents. Facebook [Backstrom and Leskovec, 2011] and Twitter [Gupta et al., 2013] use recommender systems to suggest friends, interesting posts and to serve relevant advertisements. Similarly, Linkedin, a popular professional social network, uses recommender systems [Amin et al., 2012] in suggesting relevant jobs, companies, and candidates for recruiters.

There are two main approaches to personalised recommendation: content-based filtering (CBF) [Pazzani and Billsus, 2007] and collaborative filtering (CF) [Resnick and Varian, 1997, Sarwar et al., 2001, Linden et al., 2003, Koren et al., 2009, Koren, 2010b]. CBF makes a recommendation leveraging user and item

[1] We use the term item to refer products, books, movies, music, videos, etc.

attributes. On the other hand, CF algorithms make a personalised recommendation by directly learning users' preferences from user-item interaction data. CF algorithms are widely popular mainly due to their superior performance over CBF [Linden et al., 2003]. In this dissertation, we primarily focus on developing CF algorithms for the personalised recommendation.

In a real-world recommendation task, there arise various scenarios based on data sources and problem setting. CF can be categorised into two different problem classes, *explicit feedback CF* and *one-class CF*, depending on the nature of the preference data. *Explicit feedback CF* concerns predicting user's actual preferences from rating, or like/dislikes data. *One-class CF (OC-CF)* concerns suggesting relevant items to users from the data that consists of only positive preferences such as item purchase. On the other hand, different CF scenarios arise based on the recommendation settings. For instance, *Cold-start* CF refers to the scenario when we do not have any historical data about users or items. Similarly, *Social-CF* involves leveraging users' signals from social networks when such data is available. In this dissertation, we focus on formulating algorithms for the above CF scenarios.

1.2 Motivation: What is lacking in existing CF algorithms?

Recommender systems are a very active area of research in both industry and academia. Over the years, many CF algorithms have been proposed. In general, it is desirable to have a CF framework that is (1) applicable to wide range of recommendation scenarios, (2) learning-based, (3) amenable to convex optimisation, and (4) scalable[2]. However, there are several limitations to existing approaches which limit their application.

Most existing work on CF focuses on Neighbourhood (KNN) and Matrix factorisation (MF) methods. KNN algorithms [Herlocker et al., 1999, Bell and Koren, 2007, Sarwar et al., 2001, Linden et al., 2003] make recommendations by finding similar users or items. As a key limitation, KNN uses predefined similarity metric instead of learning directly from the data by optimising some objective function, and hence is unable to adapt to the characteristics of the data at hand. On the other hand, MF algorithms [Salakhutdinov and Mnih, 2008b, Koren et al., 2009] incorporate learning by factorising the user-item preference matrix in an optimisation framework. However, MF models are non-convex, and hence are susceptible to sub-optimal local minima. Recently, [Ning and Karypis, 2011] proposed SLIM, a constrained linear model for OC-CF with a convex objective. However, SLIM is not user-focused and involves constrained optimisation limiting its applicability on large scale datasets.

In addition to the above limitations, a key problem with the existing approaches is that there is no definite answer as to which CF model class works best for broad range of recommendation scenarios. For instance, CF algorithms, such as MF, are unable to make recommendations in cold-start scenarios. Furthermore, most of the work on social-CF extends existing CF algorithms to incorporate social signals [Ma et al., 2011b, Noel et al., 2012, Ma et al., 2009b, 2008b]. However, these models are not expressive enough to incorporate fine-grained social signals limiting the full exploitation of the available data.

In summary, the existing approaches have several limitations for a real world recommendation scenario. Hence, there is a need for a different approach to CF that addresses all key desiderata.

[2]In this thesis, we refer these properties as key CF desiderata.

1.3 Our approach to CF: Linear models and beyond

As discussed in Section 1.2, the existing approaches to CF have several limitations. In this section, we discuss our approach to CF addressing all the above desiderata.

1.3.1 Linear models for recommendation

In this dissertation, we focus on investigating linear models, an underappreciated but promising area for recommendations that addresses all CF desiderata. We formulate a unified framework based on linear models for different CF scenarios.

Linear models [Nelder and Baker, 2004] are the most widely used method for various practical machine learning problems. They have several desirable properties, as we will discuss shortly, which make them attractive for a broad range of problems. First, unlike neighbourhood methods, linear methods learn from data by minimising some objective function. Second, linear models facilitate the design of *convex* objectives which ensure globally optimal solution. Third, they are *simple* and *scalable*, which is very desirable while building a large-scale system. For instance, linear regression has a closed-form solution, which can be exploited to scale recommendation to large datasets, as we will show in this thesis. Further, we will show that the formulation of CF problems as a linear model yields an embarrassingly parallel model i.e. one that can be learned without distributed communication during the optimisation. Fourth, linear models are theoretically well understood, and various efficient off-the-shelf optimisation tools and algorithms are available. Fifth, Linear models are highly *interpretable*. Interpretability of the model not only helps to understand and improve the model but also to explain the recommendations generated by the model. Especially, by explaining the recommendations, the system becomes more transparent, builds users' trust in the system and convinces them to consume the recommended items [Vig et al., 2009].

1.3.2 Beyond linear models

In the last couple of years, nonlinear methods have gained a lot of attention due to impressive performance. In particular, deep learning has revolutionised many areas of machine learning, namely computer vision [Krizhevsky et al., 2012], natural language processing [Mikolov et al., 2013] and speech recognition [Hinton et al., 2012] . However, there has been very limited research in the application of deep learning for CF [Salakhutdinov et al., 2007]. In this dissertation, we identify CF as an auto-encoding problem and propose a general nonlinear neural network architecture for CF, bridging the core of the thesis to the deep learning literature.

1.4 Key contributions of dissertation

Before outlining the contributions of this dissertation, we first define the commonly used symbols. Let $\mathbf{R}$ denote the user-item preference matrix, $\mathbf{X}$ denote the user features, and $\mathbf{X}^{(\text{te})}$ denote cold-start users features. Further, let $\mathbf{R} \approx \mathbf{P}_k\mathbf{\Sigma}_k\mathbf{Q}_k^T$ and $\mathbf{X} \approx \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ be a singular value decomposition (SVD) of the preference and feature matrix respectively. Finally, let $f(\cdot), g(\cdot)$ denote the activation function. The contributions of this dissertation are in formulating linear models for different CF scenarios and a neural architecture that generalises to various CF models. In this dissertation, we investigate various CF models as summarised in Table 1.1.

1. **Linear models for social collaborative filtering**
   In our first contribution, we focus on Social-CF problem. The majority of work on Social-CF

	Model	Recommendation	Model parameters
Linear	SAF	$\mathbf{X}diag(\mathbf{w})\mathbf{X}^T\mathbf{R}$	$\mathbf{w}$
	LoCo	$\mathbf{X}^{(\text{te})}\mathbf{V}_k\mathbf{Z}$	$\mathbf{Z}$
	LREC	$\mathbf{W}\mathbf{R}$	$\mathbf{W}$
	LINEAR-FLOW	$\mathbf{Z}\mathbf{P}_k^T\mathbf{R}$	$\mathbf{Z}$
Non Linear	AUTOREC	$f\left(\mathbf{U}\cdot g(\mathbf{V}\mathbf{R})\right)$	$\mathbf{U}$, $\mathbf{V}$

Table 1.1: Summary of the proposed models.

aggregates rich social information into a single measure of user-to-user interaction [Cui et al., 2011, Li and Yeung, 2009b, Noel et al., 2012, Ma et al., 2008b]. But in aggregating all of these interactions and activities into a single strength of interaction we discard valuable fine-grained information. Hence, the existing approaches to Social-CF are not capable of exploiting rich social data. To address this limitation, we propose a scalable linear Social-CF algorithm, *Social Affinity Filtering (SAF)*, which learns to weight fine-grained user interaction and activities in a social network. We show that only a small subset of user interactions and activities are valuable for the recommendation, hence learning which of these are most informative is of critical importance. Furthermore, we show that Facebook page likes are highly predictive of users' preferences. The insights from this work provide a foundation for the follow-up work on cold-start recommendation work.

2. **Linear models for social cold-start recommendation**
   As a second contribution, we investigate linear models for cold-start recommendation. Without loss of generality, we focus on user cold-start [3], for OC-CF setting. The user cold-start problem concerns the task of recommending items to users who have not previously purchased or otherwise expressed preferences towards any item under consideration. We show how several popular cold-start models can be seen as special case of a linear content-based model. Leveraging this insight and the predictive power of social information, we propose a class of large-scale linear models that leverages high dimensional social information for the cold-start recommendation. We present a comprehensive experimental evaluation to demonstrate the superior performance of proposed method and the predictive power of social network content in addressing the cold-start problem.

3. **Linear models for one class collaborative filtering**
   As a third contribution, motivated by the superior performance of the linear models on Social-CF and Cold-Start recommendation, we investigate linear models in a general OC-CF setting. We propose, LREC, a user-focused linear model. LREC produces user-personalised recommendations by training a convex, unconstrained objective that is embarrassingly parallelisable (i.e., without distributed communication) across users. A comprehensive set of experiments on a range of real-world datasets reveals LRecs superior performance compared to the state-of-the-art methods.

4. **Large scale linear models via dimensionality reduction**
   Linear methods, LRec and SLIM, yield good performance on OC-CF problem. However, they involve solving a large number of regression subproblems, which can be impractical to large scale problems. Furthermore, LRec is also expensive regarding the memory requirements. We address these limitations by proposing LINEAR-FLOW, which formulates OC-CF as a regularised linear regression problem that uses randomised SVD for fast dimensionality reduction. Through

[3]the proposed approach is general and also applicable to the item cold-start

extensive experiments on real-world datasets, we demonstrate that Linear-Flow achieves state-of-the-art performance as compared to other methods, with a significant reduction in computational cost.

5. **Neural network architecture for collaborative filtering**
As a final contribution, we take a departure from linear models by exploring deep learning models. We show that a particular neural architecture, an Autoencoder, can represent a broad range of CF models. We propose AUTOREC, a neural architecture, that generalises various CF methods. A comprehensive set of experiments on a range of real-world datasets reveals that AutoRec yields state-of-the-art results on the rating prediction problem.

To summarise, the core of this dissertation focus on formulating a unified approach to CF using linear models that performs well on various CF scenarios. We conclude by bridging the core of the dissertation to deep learning architecture, and leading the way for future research on applying deep learning models to CF.

1.5 Outline of dissertation

In this thesis, starting from a concrete, practically relevant scenario where linear models are useful, we progressively use the insights derived in the design of these models to apply them to a broad spectrum of CF tasks.

Chapter 2 introduces the necessary background and terminology for recommender systems. In particular, we introduce the terminologies and basic concepts. We discuss related work and seminal CF algorithms for different CF scenarios.

In Chapter 3, we introduce our novel Social collaborative filtering algorithm, Social Affinity Filtering (SAF). We provide a detailed experimental analysis and demonstrate the superiority of the proposed algorithm compared to state-of-the-art social collaborative filtering algorithms. Furthermore, we present feature analysis and demonstrate the substantial predictive power of social features.

In Chapter 4, we introduce our cold-start algorithm, LoCo, that leverages high dimensional social features. We present the experimental evaluation to demonstrate the superiority of proposed model and predictive power of social network content in addressing the cold-start problem.

In Chapter 5, we present LRec, a novel user focused linear model for one-class collaborative filtering. We perform a thorough experimentation on four real-world datasets to demonstrate the superiority of the LRec. Next, we address the computational and space limitation of LRec by proposing LINEAR-FLOW, a linear algorithm that leverages fast dimensionality reduction via randomised algorithms. In a comprehensive set of experiments, we show the proposed method is computationally efficient and yields comparable results.

Chapter 6 introduces AUTOREC, a general neural network architecture for CF. We report our evaluations on three different datasets and demonstrate that AutoRec yields the state-of-the-art results.

Finally, in Chapter 7, we summarize our work presented in this thesis, and present a number of interesting future directions.

Chapter 2

Overview of Recommender Systems

In this Chapter, we will formally define the recommendation problem, discuss approaches to the recommendation and various Collaborative Filtering (CF) algorithms as outlined in Figure 2.1. We then discuss the four prevalent CF scenarios that we investigate in this dissertation, as mentioned in Chapter 1. We conclude by discussing evaluation metrics used in this dissertation to evaluate all recommendation algorithms.

2.1 Recommendation problem: Formal definition

Recommendation is essentially a task of predicting users' preferences on items they have not consumed. Let U denote a set of users, and I a set of items, with $m = |U|$ and $n = |I|$. Let $\mathbf{R}$ denote the observed user-item preference matrix; for *explicit feedback* $\mathbf{R} \in \mathbb{R}^{m \times n}$ where higher scores refer to higher preference e.g. as in ratings, whereas for *one-class feedback* data $\mathbf{R} \in \{0, 1\}^{m \times n}$ where $\mathbf{R}_{ui}$ indicates whether u purchased item i or not. For simplicity, we set the unobserved entries, R_{ui}, to 0 . Let I_u denote the set of the items for which user u has expressed preference, and U_i be the set of the users who have expressed preference for the item i.

The goal of recommender systems is to predict the preferences of user u on unobserved items, i.e. $I - I_u$. In other words, recommendation can be defined as learning $\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$, the *recommendation matrix* which approximates users preference on items. We refer to the symbol table at the beginning of this dissertation for commonly used symbols. Formally, the recommendation problem is defined as:

Definition 1. *Given a set of users U and items I, a recommender system learns a function $\mathcal{F} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$, where $\mathcal{F}(\mathbf{R}) = \hat{\mathbf{R}}$, which predicts users preference on items. Optionally, it can take any additional information, such as user features $\mathbf{X}^U \in \mathbb{R}^{m \times d}$ or item features $\mathbf{X}^I \in \mathbb{R}^{n \times d}$*

Based on the problem formulation and its evaluation, which we will discuss later, recommendation can be grouped into two different tasks:

- ***Rating Prediction task*** concerns with predicting the *exact* rating for user u on item i. Rating prediction is exemplified by the Netflix challenge [Bennett et al., 2007b] and widely used in the explicit feedback setting.

- ***Ranking task*** concerns with recommending a ranked list of relevant items to a user u. Since the end goal of a recommender system is to suggest a list of relevant items, recommendation as a ranking task is preferred in real world setting.

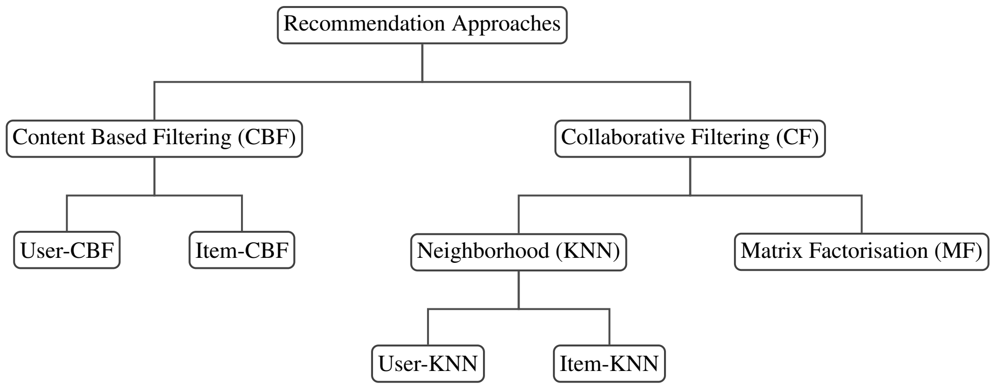

Recommendation Approaches
Content Based Filtering (CBF)
Collaborative Filtering (CF)
User-CBF
Item-CBF
Neighborhood (KNN)
Matrix Factorisation (MF)
User-KNN
Item-KNN

Figure 2.1: Overview of recommender system approaches and models.

2.2 Approaches to recommendation

There are two main approaches to personalised recommendation, namely *Content based filtering* and *Collaborative filtering*.

2.2.1 Content based filtering (CBF)

Content-based filtering (CBF) [Pazzani and Billsus, 2007] uses item or user features [1] to predict the users' preferences. CBF treats the recommendation as regression or classification problem and learns a user or item specific recommendation model. There are two categories of CBF:

- ***User-CBF*** makes a recommendation by learning an item-specific model from the user features and observed preference for the corresponding item. Principally, User-CBF makes a recommendation by finding the items liked by the users with similar features. User demographics (age, sex, location, etc.) are widely used features for User-CBF.
- ***Item-CBF*** learns a user-specific recommendation model from the item features and observed preference for the corresponding user. Item descriptions, tags, and reviews are widely used item features for Item-CBF.

CBF has some advantages, especially for *cold-start* recommendation i.e. they can provide meaningful recommendation to the users without any preference data. However, the key limitation of CBF is that they suffer from over-specialization [Lops et al., 2011], which means the recommended items lack novelty. For instance, the recommendation for Item-CBF is restricted to the items that are similar to the previously liked items. Similarly, the recommendations for User-CBF are limited to the items liked by similar users based on a coarse approximation from the users' features. Hence, CBF lack serendipitous discovery of items.

2.2.2 Collaborative filtering (CF)

Collaborative Filtering (CF) [Goldberg et al., 1992, Resnick and Varian, 1997, Sarwar et al., 2001, Linden et al., 2003, Koren et al., 2009, Koren, 2010b] is the most popular recommendation algorithm and has become the de facto choice of model for recommendation. CF algorithms are based on the general idea that users' preferences may be correlated and one can detect and exploit these correlations across

[1]We use the term feature to refer user attributes and item content/descriptions.

the user population to make recommendations for a user. In a broad sense, CF algorithms leverages three different kinds of correlation within the user-item preference data:

- *inter-item correlation*, which refers to the fact that the items liked by similar set of users are correlated
- *inter-user correlation*, which refers to the fact that the user who liked similar set of items are correlated
- *intra user-item correlation*, which refers to the fact that the users who agreed in the past will tend to agree in the future. Similarly, users tend to prefer the items that are similar to the ones they already liked.

Based on these key assumptions, CF algorithms exploit the wisdom of the crowd by predicting users' preferences not only from her past actions but also from the preferences of like-minded users. In other words, CF algorithms allows users to collaborate with each other in predicting user's preferences.

Over the years, many CF algorithms have been proposed. Tapestry [Goldberg et al., 1992] was the first system to introduce the term CF and applied it in the context of email filtering. Later, [Resnick et al., 1994] proposed a rating based CF system for recommending news articles. Although CF had been an active area research in industry, it received lots of attention after the "Netflix challenge" [Bennett et al., 2007b].

CF algorithms are attractive for recommendation due to various reasons. First, CF learns to make personalised recommendation from preference data. Second, unlike CBF, CF algorithms do not over-specialize and are capable of making *serendipitous* [Herlocker et al., 2004] recommendations. However, most of the CF algorithms are not capable of making recommendations in cold-start scenarios. In the following section, we will discuss various CF algorithms in detail.

2.3 Collaborative filtering models

In this section, we discuss the various types of CF algorithms.

2.3.1 Neighbourhood models (KNN)

Neighbourhood-based CF models [Herlocker et al., 1999, Bell and Koren, 2007, Sarwar et al., 2001, Linden et al., 2003] are based on the general idea that users share their taste with similar users or similar items. Neighbourhood-based CF makes recommendations by computing k-nearest neighbours for each user or item. Hence, it is referred as k-Nearest Neighbour (KNN) algorithm in the literature. KNN based CF defines neighbourhood from observed user-item preference data using various predefined similarity metrics, which we will discuss shortly. There are two approaches to neighbourhood-based models:

1. **User-Based Neighbourhood Model (U-KNN)** [Herlocker et al., 1999, Bell and Koren, 2007] leverages *inter-user* and *intra user-item* correlation from the observed user-item preference data. It is based on the key assumption that the similar users like similar items and predicts users' preferences using the user-user similarity matrix.

2. **Item-Based Neighbourhood Model (I-KNN)** [Sarwar et al., 2001, Linden et al., 2003] leverages *inter-item* and *intra user-item* correlation from the observed user-item preference data. It assumes that the users tend to like the items that are similar to the previously liked items. While U-KNN uses user-user similarity for recommendation, I-KNN uses item-item similarity for recommendation.

Various similarity metrics are used to define the neighbourhood. The most common similarity metrics are

Pearson Correlation, which measures the strength and direction of the linear relationship between entities. For rating data, Pearson correlation between users u and v is computed as

$$\mathbf{S}_{uv} = \frac{\sum_{i \in I_{uv}} (\mathbf{R}_{ui} - \bar{\mathbf{R}}_u)(\mathbf{R}_{vi} - \bar{\mathbf{R}}_v)}{\sqrt{\sum_{i \in I_{uv}} (\mathbf{R}_{ui} - \bar{\mathbf{R}}_u)^2} \sqrt{\sum_{i \in I_{uv}} (\mathbf{R}_{vi} - \bar{\mathbf{R}}_v)^2}}$$

where, $I_{uv} = I_u \cap I_v$, and $\bar{\mathbf{R}}_u$ and $\bar{\mathbf{R}}_v$ are the mean of the observed ratings for user u and v respectively.

Cosine Similarity, which measures the angle between two vectors. It is widely used in both rating prediction and item recommendation problem. The cosine similarity between user u and v is computed as

$$S_{uv} = \frac{\langle \mathbf{R}_{u:}, \mathbf{R}_{v:} \rangle}{\|\mathbf{R}_{u:}\|_2 \, \|\mathbf{R}_{v:}\|_2},$$

where $\langle \cdot, \cdot \rangle$ denotes inner product.

Jaccard Similarity, which measures the similarity between two sets. The Jaccard similarity between user u and v is computed as

$$\mathbf{S}_{uv} = \frac{|I_u \cap I_i|}{|I_u \cup I_i|}$$

The similarities between items $\mathbf{S}_{ij}$ can be computed in a same way.

Neighbourhood methods are attractive for several reasons. They are simple to implement, efficient, and interpretable. However, they define $\mathbf{S}$ using a fixed similarity metrics instead of learning them by optimising some principled loss function [Koren, 2008]. Further, recommendation performance can be quite sensitive to the choice of $\mathbf{S}$, and the choice depends on the problem domain(as shall be shown in later experiments). Hence, neighbourhood methods are unable to adapt to the characteristics of the data at hand.

2.3.2 Matrix factorisation models (MF)

MF based models [Srebro and Jaakkola, 2003, Salakhutdinov and Mnih, 2008b, Koren et al., 2009] embed users and items into some shared latent space, with the aim of inferring complex preference profiles for both users and items. MF models are based on the key idea that the user-item preference data is correlated, and hence can be expressed in terms of low-rank user and item latent factors.

Formally, let $\mathbf{J} \in \mathbb{R}_+^{m \times n}$ be some pre-defined weighting matrix to be defined shortly. Let $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be some loss function, typically squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. Then, the general matrix factorisation framework optimises

$$\min_{\theta} \sum_{u \in U, i \in I} \mathbf{J}_{ui} \cdot \ell(\mathbf{R}_{ui}, \hat{\mathbf{R}}_{ui}(\theta)) + \Omega(\theta), \tag{2.1}$$

$$\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{m-2} & \multicolumn{1}{c}{m-1} & \multicolumn{1}{c}{m} \\ \hline 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \hline \end{array}}_{\text{one-hot encoding for user } u_4}\underbrace{\begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{m+1} & \multicolumn{1}{c}{m+1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{m+d_u-1} & \multicolumn{1}{c}{m+d_u} \\ \hline 1 & 0 & \cdots & 0 & 1 \\ \hline \end{array}}_{u_4 \text{ features}}$$

Figure 2.2: $\mathbf{X}^U$ for user u_4 in matchbox model

where the recommendation matrix is[2]

$$\hat{\mathbf{R}}(\theta) = \mathbf{A}^T\mathbf{B} \tag{2.2}$$

for $\theta = \{\mathbf{A}, \mathbf{B}\}$, and $\Omega(\theta)$ is the *regulariser*, which is typically

$$\Omega(\theta) = \frac{\lambda}{2} \cdot (||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2)$$

for some $\lambda > 0$. The matrices $\mathbf{A} \in \mathbb{R}^{k \times m}, \mathbf{B} \in \mathbb{R}^{k \times n}$ are the *latent representations* of users and items respectively, with $k \in \mathbb{N}_+$ being the *latent dimension* of the factorisation.

MF is one of the most extensively studied model in the field of CF. [Salakhutdinov and Mnih, 2008a] proposed a fully Bayesian model for MF. Similarly, [Lawrence and Urtasun, 2009] proposed a nonlinear MF algorithm using Gaussian process. However, these models are computationally expensive and are not suitable for large scale recommendation.

Further, there have been extensions of MF to incorporate user and item side information such as [Stern et al., 2009] , which we refer as *Matchbox* model. Matchbox defines the recommendation matrix as

$$\hat{\mathbf{R}}(\theta) = (\mathbf{A}\mathbf{X}^U)^T(\mathbf{B}\mathbf{X}^I) \tag{2.3}$$

where, $\mathbf{A} \in \mathbb{R}^{k \times (m+d_u)}$, $\mathbf{B} \in \mathbb{R}^{k \times (n+d_i)}$, $\mathbf{X}^U \in \mathbb{R}^{(m+d_u) \times m}$, and $\mathbf{X}^I \in \mathbb{R}^{(n+d_i) \times n}$. Here, $\mathbf{X}^U$ composed of users features and one-hot encoding of the user index as shown in Figure 2.2. Similarly, $\mathbf{X}^I$ composed of item features and one-hot encoding of the item index. Intuitively, Matchbox embeds the user and item side information to the k dimensional latent space. Matchbox corresponds to MF model if only one-hot encoding of user and item indices are used as features.

One of the fundamental limitation of MF based methods is that the problem is non-convex, hence susceptible to local minima. Furthermore, the recommendations are based on latent factors and are not easily interpretable [Zhang et al., 2014] as in neighbourhood methods.

2.4 Collaborative filtering scenarios

In a real-world recommendation task, there arise various scenarios based on data sources and problem setting. In this dissertation, we discuss four prevalent scenarios, as discussed in Section 1.1:

2.4.1 Explicit feedback CF

Explicit feedback CF concerns with the prediction of users' *actual* preferences such as rating, like/dislikes. From here onwards, we use the term ***"rating"*** to refer explicit feedback. We discuss various CF models for explicit feedback scenarios :

[2]Typically, one also includes user- and item- bias terms in the recommendation matrix. We omit these for brevity.

KNN models

User-based KNN [Herlocker et al., 1999, Bell and Koren, 2007] defines the rating prediction as weighted sum of the ratings given by users' neighbours.

$$\hat{\mathbf{R}}_{ui} = \frac{\sum_{v \in \mathcal{N}^k(u,i)} \mathbf{S}_{uv}(\mathbf{R}_{vi} - b_{vi})}{\sum_{v \in \mathcal{N}^k(u,i)} \mathbf{S}_{uv}} \tag{2.4}$$

where $\mathcal{N}^k(u,i)$ is the set of k-nearest neighbours, as defined by $\mathbf{S}$, of user u who have rated the item i.

Item-based KNN [Sarwar et al., 2001] defines the rating prediction as weighted sum of the rating of the similar items.

$$\hat{\mathbf{R}}_{ui} = \frac{\sum_{i \in \mathcal{N}^k(u,i)} \mathbf{S}_{ij}(\mathbf{R}_{u,j} - b_{uj})}{\sum_{j \in \mathcal{N}^k(u,i)} \mathbf{S}_{ij}} \tag{2.5}$$

where $\mathcal{N}^k(u,i)$ is a set of k-nearest neighbour of item i which is rated by the user u.

MF models

For rating prediction [Salakhutdinov and Mnih, 2008b, Koren et al., 2009], one typically sets $\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} > 0]\!]$ in equation 2.1 , so that one only considers (user, item) pairs with known preference information. The MF model [Salakhutdinov and Mnih, 2008b] for rating prediction optimises

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u \in U_i, i \in I_u} (\mathbf{R}_{ui} - \mathbf{A}_u^T \mathbf{B}_i)^2 + \frac{\lambda}{2}(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2) \tag{2.6}$$

Additionally, biased matrix factorisation (*Biased-MF*) [Koren et al., 2009] incorporates a global, user, and item bias by optimising

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u \in U_i, i \in I_u} (\mathbf{R}_{ui} - b - b_u - b_i - \mathbf{A}_u^T \mathbf{B}_i)^2 + \frac{\lambda}{2}(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2), \tag{2.7}$$

where $b,\ b_u,\ b_i$ are global, user and item bias respectively.

Most recently, [Lee et al., 2013] proposed Local Low Rank Matrix Factorisation (LLORMA), an ensemble MF algorithm that minimises the squared error weighted by the proximity of user-item pair to a predefined point called anchor point. Formally, it minimises

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(u^*,i^*) \in \mathcal{Q}} K((u,i),(u^*,i^*))(\mathbf{R}_{ui} - \mathbf{A}_u^T \mathbf{B}_i)^2 + \frac{\lambda}{2}(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2), \tag{2.8}$$

where $K((u,i),(u^*,i^*))$ is a two-dimensional smoothing kernel that measures the proximity of target point (u,i) to the anchor point (u^*,i^*); $\mathcal{Q}$ is a set of anchor points. In general, it learns local matrix factorisation model with respect to each anchor point. Finally, given q different anchor points, the rating is estimated as a linear combination of predictions from each model.

2.4.2 One-class CF

One-Class Collaborative Filtering (OC-CF) [Pan et al., 2008] concerns suggesting relevant items to users from the data that consists of positive only preferences such as item purchase. In this dissertation, we use the term "***purchase***" to refer one-class feedback in general. We discuss various CF models for OC-CF scenarios :

KNN models

User-based KNN predicts the users' preferences on items as:

$$\hat{\mathbf{R}}_{ui} = \sum_{v \in \mathcal{N}^k(u,i)} \mathbf{S}_{uv} \tag{2.9}$$

where $\mathcal{N}^k(u,i)$ is a set of k-nearest neighbour of user u who also purchased the item i.

Item-based KNN [Linden et al., 2003] predicts the users' preferences on items as:

$$\hat{\mathbf{R}}_{ui} = \sum_{j \in \mathcal{N}^k(u,i)} \mathbf{S}_{ij} \tag{2.10}$$

where $\mathcal{N}^k(u,i)$ a set of k-nearest neighbour of item i that are purchased by user u.

Note that unlike rating prediction(as in Equation 2.4 and 2.5), we do not normalise the score. This is mainly because we are concerned with the relative scores in OC-CF setting.

Sparse Linear Methods (SLIM)

As an alternative to neighbourhood methods for OC-CF, Sparse linear Methods (SLIM) [Ning and Karypis, 2011] directly learns an item-similarity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ via

$$\begin{gathered} \min_{\mathbf{W} \in \mathcal{C}} ||\mathbf{R} - \mathbf{RW}||_F^2 + \frac{\lambda}{2}||\mathbf{W}||_F^2 + \mu||\mathbf{W}||_1 \\ \text{where } \mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{n \times n} \colon \operatorname{diag}(\mathbf{W}) = 0, \mathbf{W} \geq 0\}, \end{gathered} \tag{2.11}$$

where $\lambda, \mu > 0$ are appropriate constants. Here, $|| \cdot ||_1$ denotes the elementwise ℓ_1 norm of $\mathbf{W}$ so as to encourage sparsity, and the constraint $\operatorname{diag}(\mathbf{W}) = 0$ prevents a trivial solution of $\mathbf{W} = \mathbf{I}_{n \times n}$. The nonnegativity constraint encourages interpretability, but Levy and Jack [2013] demonstrated that good performance can be achieved without it. Given a learned $\mathbf{W}$, SLIM produces a recommendation matrix

$$\hat{\mathbf{R}}(\theta) = \mathbf{RW}.$$

Thus, SLIM is equivalent to an item-based neighbourhood approach where the similarity matrix $\mathbf{S} = \mathbf{W}$ is *learned* from data. Although SLIM has a convex learning objective, it is item-focused which hampers it predictive performance. Similarly, it involves constrained optimisation, which limits fast training.

MF models

For OC-CF, we cannot set the weighting matrix $\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} > 0]\!]$ for MF objective shown in equation 2.1, as one will simply learn on the known positive preferences, and thus predict $\mathbf{R}_{ui} = 1$. An alternative is to set $\mathbf{J}_{ui} = 1$ uniformly. This treats all absent purchases as indications of a negative preference. The resulting approach is termed *PureSVD* in [Cremonesi et al., 2010], and has been shown to perform surprisingly well in top-N recommendation task for both explicit and implicit feedback datasets [Cremonesi et al., 2011].

As an intermediate between the two extreme weighting schemes above, the *WRMF* method [Pan et al., 2008, Hu et al., 2008] sets $\mathbf{J}_{ui}$ to be

$$\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} = 0]\!] + \alpha \cdot [\![\mathbf{R}_{ui} > 0]\!] \tag{2.12}$$

where α assigns an importance weight to the observed preferences. *WRMF* uses Alternating Least Squares (ALS) method for optimization. The solution for $\mathbf{A}$ and $\mathbf{B}$ in each step of ALS is given by (2.13)

$$\begin{aligned} \mathbf{A}_{:u} &= (\mathbf{B}\mathbf{J}^{U}\mathbf{B}^{T} + \lambda\mathbf{I})^{-1}\mathbf{B}\mathbf{J}^{U}\mathbf{R}^{T}_{u:} \\ \mathbf{B}_{:i} &= (\mathbf{A}\mathbf{J}^{I}\mathbf{A}^{T} + \lambda\mathbf{I})^{-1}\mathbf{A}\mathbf{J}^{I}\mathbf{R}_{:i} \end{aligned} \tag{2.13}$$

where $\mathbf{J}^{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{J}^{I} \in \mathbb{R}^{m\times m}$ are diagonal matrices such that $\mathbf{J}^{U}_{ij} = \mathbf{J}_{ui}$ and $\mathbf{J}^{I}_{uu} = \mathbf{J}_{ui}$. In each iteration of ALS, due to the weighting, we need to compute the inverse for each user and item as shown in (2.13). This makes WRMF computationally expensive compared to using uniform weights.

[Tang and Harrington, 2013] proposed a randomised SVD based method to scale uniformly weighted MF for OC-CF on large-scale datasets. The proposed method involves computation of rank-k randomised SVD of the matrix $\mathbf{R}$ (as discussed in Appendix A).

$$\mathbf{R} \approx \mathbf{P}_k\mathbf{\Sigma}_k\mathbf{Q}_k^T \tag{2.14}$$

where $\mathbf{P}_k \in \mathbb{R}^{m\times k}$, $\mathbf{Q}_k \in \mathbb{R}^{n\times k}$ and $\mathbf{\Sigma}_k \in \mathbb{R}^{k\times k}$. Given the truncated SVD solution, they initialize the item latent factor with the SVD solution and solve

$$\underset{\mathbf{A}}{\operatorname{argmin}} \left\|\mathbf{R} - \mathbf{A}^T\mathbf{B}\right\|_F^2 + \lambda \left\|\mathbf{A}\right\|_F^2 \text{ s.t. } \mathbf{B} = \mathbf{\Sigma}_k^{\frac{1}{2}}\mathbf{Q}_k^T \tag{2.15}$$

Similarly, if the matrix $\mathbf{A}$ is fixed instead of the matrix $\mathbf{B}$, the objective becomes

$$\underset{\mathbf{B}}{\operatorname{argmin}} \left\|\mathbf{R} - \mathbf{A}^T\mathbf{B}\right\|_F^2 + \lambda \left\|\mathbf{B}\right\|_F^2 \text{ s.t. } \mathbf{A} = \mathbf{P}_k\mathbf{\Sigma}_k^{\frac{1}{2}} \tag{2.16}$$

We refer to (2.16) and (2.15) as U-MF-RSVD and I-MF-RSVD respectively.

Bayesian Personalised Ranking (BPR)

An alternate strategy to adapt matrix factorisation techniques to the OC-CF is the *Bayesian Personalised Ranking (BPR)* Rendle et al. [2009]. BPR optimises a loss over (user, item) pairs, so as to ensure that the known positive preferences score at least as high as the unknown preferences:

$$\min_{\theta} \sum_{u\in\mathcal{U}, i\in\mathcal{R}(u), i'\notin\mathcal{R}(u)} \ell(1, \hat{\mathbf{R}}_{ui}(\theta) - \hat{\mathbf{R}}_{ui'}(\theta)) + \Omega(\theta), \tag{2.17}$$

where $\ell(1, v) = \log(1 + e^{-v})$ is the logistic loss, and $\hat{\mathbf{R}}$ is as per Equation 2.2. Intuitively, this forces the scores for items with unknown preferences below that of an item with a positive preference (which are anyway not relevant for the purposes of future recommendation). For each user, the objective can be seen as maximisation of a surrogate to (a scaled version of) the area under the ROC curve (AUC) over items. The AUC is the probability of a random relevant item scoring higher than a random positive item, and is a popular measure of performance given binary relevance judgments [Fürnkranz and Hüllermeier, 2010, pg. 6]. While the objective has a quadratic complexity in the number of items, stochastic gradient optimisation is feasible; nonetheless computational complexity remains a concern with such pairwise ranking approaches.

Gantner et al. [2012] proposed an extension of BPR that normalises the loss for each user,

$$\min_{\theta} \sum_{u \in U, i \in \mathcal{R}(u), i' \notin \mathcal{R}(u)} \frac{1}{|\mathcal{R}(u)|} \cdot \ell(1, \hat{\mathbf{R}}_{ui}(\theta) - \hat{\mathbf{R}}_{ui'}(\theta)) + \Omega(\theta).$$

This can be seen to be more appropriate than Equation 2.17 in scenarios where we want to ensure good recommendations for the average user, i.e. the model is user-focused.

2.4.3 Cold-start CF

Cold-start problem [Schein et al., 2002] refers to the scenario when we do not have any historical preference data about the users or items under consideration for recommendation. This proves a challenge for CF algorithms that explicitly rely on such information to make personalised recommendations. In such scenarios, user and item side information [Zhang, Zi-Ke et al., 2010, Sahebi and Cohen, 2011, Ma et al., 2008a, Cao et al., 2010, Jamali and Ester, 2010, Krohn-Grimberghe et al., 2012] are leveraged to make personalised recommendation.

Formally, in a standard cold-start setting, we have set of users with historical preference data, $U^{(\mathrm{tr})}$, referred as *warm start users*, and target *cold-start users* $U^{(\mathrm{te})}$. Let $\mathbf{X} \in \mathbb{R}^{m \times d}$ refer to user metadata, and $\mathbf{X}^{(\mathrm{tr})}, \mathbf{X}^{(\mathrm{te})}$ be the metadata for the warm- and cold-start users respectively. There are two classes of model that leverages side-information for cold-start recommendation:

Neighbourhood based cold-start CF

Neighbourhood based cold-start CF [Zhang, Zi-Ke et al., 2010, Sahebi and Cohen, 2011] uses metadata to compute user-user or item-item similarity, and makes recommendation as discussed in 2.3.1. We will discuss the neighborhood based cold-start CF in detail in Chapter 4.

MF based cold-start CF

MF based cold-start CF [Krohn-Grimberghe et al., 2012] borrows the idea from collective matrix factorisation (CMF) [Singh and Gordon, 2008] for cold-start recommendation and optimises

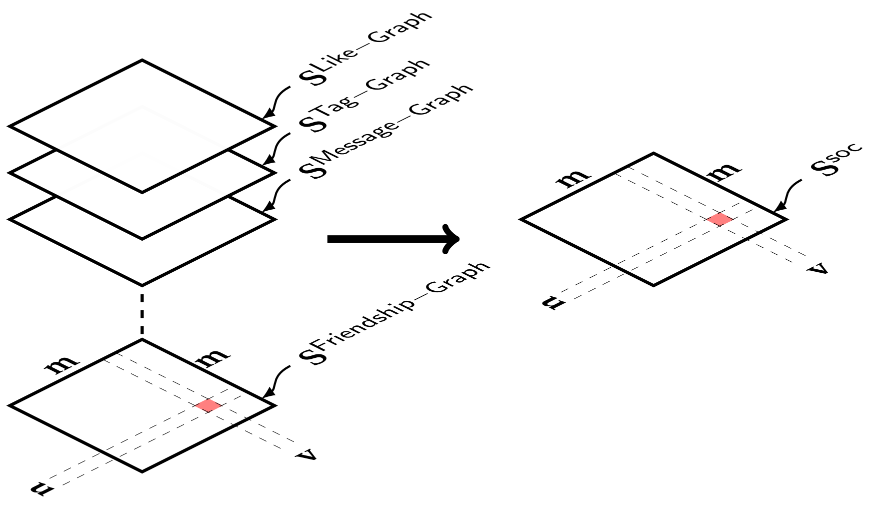
$$\min_{\mathbf{A},\mathbf{B},\mathbf{Z}} ||\mathbf{R} - \mathbf{A}\mathbf{B}||_F^2 + \mu||\mathbf{X} - \mathbf{A}\mathbf{Z}||_F^2 + \lambda_A||\mathbf{A}||_F^2 + \lambda_B||\mathbf{B}||_F^2 + \lambda_Z||\mathbf{Z}||_F^2 \tag{2.18}$$

where $\mathbf{A} \in \mathbb{R}^{m \times k}, \mathbf{V} \in \mathbb{R}^{k \times n}$, and $\mathbf{Z} \in \mathbb{R}^{k \times d}$ for some *latent dimensionality* $k \ll \min(m, n)$. The intuition for this approach is that it finds a latent subspace $\mathbf{A}$ for users that is jointly predictive of both their preferences and social characteristics. We then predict

$$\hat{\mathbf{R}}^{(\mathrm{te})} = \mathbf{A}^{(\mathrm{te})}\mathbf{B}. \tag{2.19}$$

Another popular cold-start approach is the two-step model *BPR-LinMap* [Gantner et al., 2010]. Here, the first step is to model the warm-start users by $\mathbf{R}^{(\mathrm{tr})} \approx \mathbf{A}^{(\mathrm{tr})}\mathbf{B}$, with latent features $\mathbf{A}^{(\mathrm{tr})}, \mathbf{B}$ as before. The second step is to learn a mapping between the metadata $\mathbf{X}^{(\mathrm{tr})}$ and latent features $\mathbf{A}^{(\mathrm{tr})}$ using e.g. linear regression,

$$\mathbf{A}^{(\mathrm{tr})} \approx \mathbf{X}^{(\mathrm{tr})}\mathbf{T} \tag{2.20}$$



$\mathbf{S}^{Like-Graph}$
$\mathbf{S}^{Tag-Graph}$
$\mathbf{S}^{Message-Graph}$
$\mathbf{S}^{Friendship-Graph}$
$\mathbf{S}^{soc}$
m
m
u
v

Figure 2.3: Aggregation of the social features.

for $\mathbf{T} \in \mathbb{R}^{d \times k}$. We then estimate the cold-start latent features $\mathbf{A}^{(\text{te})} = \mathbf{X}^{(\text{te})}\mathbf{T}$, and use Equation 2.19 for prediction.

2.4.4 Social CF

Social-CF is an emerging field in recommender systems research. Social-CF concerns about leveraging social signals from online social networks for a recommendation. The fundamental assumption in Social-CF is that users' choices are not only influenced by personal preferences but also by interpersonal factors, such as friend groups. Recent findings from social science research also reinforce the importance of social influence in user preferences. For instance, [Brandtzg and Nov, 2011] found that the real-world interactions correlate with the strength of ties while virtual interactions reveal interest. Similarly, people who interact frequently tend to share similar interests [Singla and Richardson, 2008] and level of user interactions correlate with the positive ratings that they give each other [Anderson et al., 2012].

The signals from online social networks range from simple user-user interaction to complex relations and interactions. In general, social networks are composed of two components, the *Social graph*, which consists of user-user relationship, such as friendship, and the *Interest graph*, which consists of user preferences and interests such as page likes, group membership. In general, Social-CF extends MF model and leverages user-user social relation, $\mathbf{S}^{soc} \in \mathbb{R}^{m \times m}$, defined by aggregating different social signals. In Figure 2.3, we show aggregation of social signals into $\mathbf{S}^{soc}$ from various user-user social signals, such as $\mathbf{S}^{Friendship-Graph}$, which refers to user-user friendship relationship; $\mathbf{S}^{Message-Graph}$, which refers to the communication graph; $\mathbf{S}^{Tag-Graph}$, which indicates whether users are tagged in same photo; and $\mathbf{S}^{Like-Graph}$, which indicates whether users have shared likes, i.e

$$\mathbf{S}^{soc} = \mathcal{F}(\mathbf{S}^{Like-Graph}, \mathbf{S}^{Tag-Graph}, \mathbf{S}^{Message-Graph}, ..., \mathbf{S}^{Friendship-Graph}) \tag{2.21}$$

where $\mathcal{F}$ is some aggregation function. In general, $\mathbf{S}^{soc}$ is incorporated in MF framework by adding a social loss component, $\mathcal{L}_{social}$ [Cui et al., 2011, Yang et al., 2011a, Ma et al., 2011a, Li and Yeung, 2009a] to MF objective. In other words, $\mathcal{L}_{social}$ acts as a regulariser to MF objective.

$$\mathcal{L}_{social} = \ell(\mathbf{S}^{\text{soc}}_{uv}, \mathbf{A}) \tag{2.22}$$

For instance, [Ma et al., 2011a, Li and Yeung, 2009a] used friendship relation and defined $\mathcal{L}_{\text{soc}}$

$$\mathcal{L}_{social} = \sum_{u} \sum_{v \in \mathcal{N}(u)} \mathbf{S}^{soc}_{uv} \, ||\mathbf{A}_u - \mathbf{A}_v||^2$$

where $\mathcal{N}(u)$ is a set of friends of user u.

As an alternative to social regularisation, [Ma et al., 2008a] formulated social recommendation as a co-factorisation problem

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u,i} (\mathbf{R}_{ui} - \mathbf{A}_u^T \mathbf{B}_i)^2 + \sum_{u,v} (\mathbf{S}^{\text{soc}}_{uv} - \mathbf{A}_u^T \mathbf{Z}_i)^2 + \frac{\lambda}{2}(||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2 + ||\mathbf{Z}||_F^2) \tag{2.23}$$

Similarly, [Ma et al., 2009a] proposed a model that predicts rating as a weighted average of users' and her friends rating, and minimises

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u,i} (\mathbf{R}_{ui} - (\alpha \mathbf{A}_u^T \mathbf{B}_i + (1-\alpha) \sum_{v \in \mathcal{N}(u)} \mathbf{A}_v^T \mathbf{B}_i)^2 \tag{2.24}$$

Most recently, [Noel et al., 2012] proposed a social extension to Matchbox, as defined in Equation 4.5. We refer this model as *Social Matchbox (SMB)*. It optimises

$$\min_{\mathbf{A},\mathbf{B}} \sum_{u \in U_i, i \in I_u} (\mathbf{R}_{ui} - (\mathbf{A}\mathbf{x}_u^U)^T (\mathbf{B}\mathbf{x}_i^I))^2 + \frac{\lambda_S}{2} \sum_{u,v} (\mathbf{S}^{\text{soc}}_{uv} - (\mathbf{A}\mathbf{x}_u^U)^T (\mathbf{A}\mathbf{x}_v^U))^2 + \frac{\lambda_A}{2} ||\mathbf{A}||_F^2 + \frac{\lambda_B}{2} ||\mathbf{B}||_F^2 \tag{2.25}$$

where $\mathbf{x}_u^U$ and $\mathbf{x}_i^I$ are the corresponding side-information, as shown in Figure 2.2, for the user u and item i respectively.

In general, the key limitation of existing Social-CF methods is that they do not leverage fine-grained social interactions. Instead, they use $\mathbf{S}^{\text{soc}}$, an aggregation of all social signals into a single numeric value, limiting the full exploitation of the available social data.

2.5 Evaluation of recommender systems

The evaluation of a model is of critical importance while developing machine learning systems. Evaluation allows to quantify the performance of the model. In general, there are two broad categories of model evaluation, namely *online* and *offline* evaluation. Online evaluation concerns with comparing different models in a production environment by directly collecting feedback from the users such as A/B testing. On the other hand, offline evaluation concerns with evaluating the models used the gathered data by dividing it into train and test set. Although online testing is desirable, it is expensive and most importantly requires access to the production system. Hence, in this dissertation, we use offline evaluation to evaluate the models.

In this section, we introduce various rating and ranking metrics used throughout this dissertation to evaluate the models.

2.5.1 Error based metrics

Error based metrics evaluate the prediction accuracy, and are widely used to access the performance of recommender system for rating prediction. In this dissertation, we use the two most widely used metrics:

Mean Absolute Error (MAE) measures how close the predicted value is from the ground truth in magnitude

$$\mathrm{MAE}(\mathbf{R}, \hat{\mathbf{R}}) = \frac{1}{|\mathbf{R}|} \sum_{u,i} |\mathbf{R}_{ui} - \hat{\mathbf{R}}_{ui}|$$

where $|\mathbf{R}|$ is the number of observed entries in $\mathbf{R}$.

Root Mean Square Error (RMSE) penalises large errors more compared to MAE. RMSE was popularised in recommender systems evaluation by the Netflix competition [Bennett et al., 2007b].

$$\mathrm{RMSE}(\mathbf{R}, \hat{\mathbf{R}}) = \sqrt{\frac{1}{|\mathbf{R}|} \sum_{u,i} (\mathbf{R}_{ui} - \hat{\mathbf{R}}_{ui})^2}$$

2.5.2 Ranking metrics

Ranking metrics evaluates the quality of the recommended list, and are widely used to evaluate the performance of recommendation models. In this dissertation, we use the three most widely used ranking metrics: *precision@k, recall@k*, and *mean Average Precision@k*. Let, I_u^{rec} be the sorted list of recommended items for a user u.

Precision@k measures the fraction of relevant items in the top-k recommended list, and is defined as

$$\text{precision@k(u)} = \frac{|I_u \cap I_u^{rec}[:k]|}{k}$$
$$\text{precision@k} = \sum_{u=1}^{m} \frac{precision@k(u)}{m}$$

Recall@k measures the fraction of the relevant items that are present in the top-k recommended list, and is defined as

$$\text{recall@k(u)} = \frac{|I_u \cap I_u^{rec}[:k]|}{|I_u|}$$
$$\text{recall@k} = \sum_{u=1}^{m} \frac{recall@k(u)}{m}$$

mean Average Precision@k (mAP@k) [3] is another widely used metric to evaluate recommender systems. Unlike *precision@k* and *recall@k*, *mAP@k* considers ordering of the items in the recommended list by giving more importance to the item at the top of the list. Formally, it is defined

[3]As defined in `https://www.kddcup2012.org/c/kddcup2012-track1/details/Evaluation`.

as

$$\text{ap@k(u)} = \sum_{i=1}^{k} \frac{precision@i(u)}{min(|I_u|, |I_u^{rec}[: i]|)}$$

$$\text{mAP@k} = \sum_{u=1}^{m} \frac{ap@k(u)}{m}.$$

Mean average precision approximates the area under the precision-recall curve for each user [Manning et al., 2008].

2.6 Summary

This chapter reviewed the foundations of recommender systems research. We formally defined recommendation problem and discussed various recommendation algorithms and their limitations. From here onwards, each chapter focuses on formulating new algorithms for various CF scenarios, addressing the limitations of the existing algorithms.

Chapter 3

Linear Models for Social Recommendation

In this Chapter, we delve into Social Collaborative Filtering (Social-CF), one of the prevalent CF scenarios, as discussed in Chapter 2. We highlight the limitations of the existing methods and formulate a novel Social-CF algorithm using linear models addressing the key limitations. The insights from this chapter will serve as a foundation for cold-start recommendation, another CF scenario, which we will discuss in the following chapter.

3.1 Problem statement

In this Chapter, we are concerned with predicting whether user u likes an item i or not. The historical preference data consist of explicit likes and dislikes i.e. we have partially observed user item preference data $\mathbf{R} \in \{0, 1\}^{m \times n}$, where 1 indicates like and 0 indicates dislike. Further, we have rich social network data of the corresponding users and their friends. Unlike the classical recommendation setting, such as movie recommendation, in many real world scenarios the items can be highly heterogeneous. For instance, Facebook recommends us the links liked by friends, where the link can be a hyper-link to videos, blog posts, and news articles. In such heterogeneous settings, the number of items for recommendation can be large, leading to the *data sparsity* problem. Hence, it is critical to leverage user side information. In this Chapter, we focus on formulating models that leverage users' social network data.

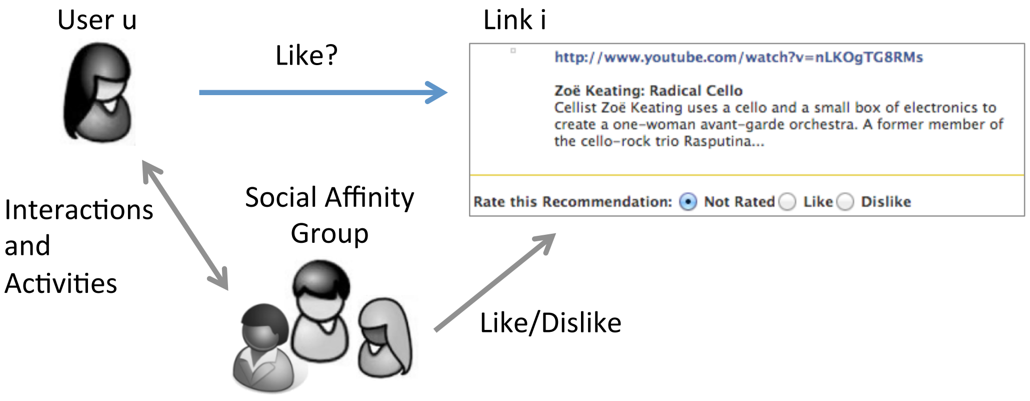
3.2 Background

In this Chapter, we investigate Social-CF algorithms in the context of the Facebook social network. Online social networks such as Facebook record a rich set of user preferences (likes of links, posts, photos, videos), user traits, interactions and activities (conversation streams, tagging, group memberships, interests, personal history, and demographic data). The availability of rich labelled graph of social interactions and contents presents a myriad of new dimensions to the recommendation problem. The users' signals from a social network range from simple user-user signals to complex information that spans across the network. However, most existing Social-CF methods, as discussed in section 2.4.4, aggregate the rich fine-grained social signals into a simple measure of user-to-user interaction [Cui et al., 2011, Yang et al., 2011a, Ma et al., 2011a, Li and Yeung, 2009a, Noel et al., 2012]. But by aggregating all of these signals into a *single* strength of interaction, we discard rich fine-grained social signals, as we will show in this chapter. Further, the existing methods extend Matrix Factorisation (MF) models, which are inherently non-convex and hard to interpret, as discussed in Section 2.3.2.

Acronyms	Meaning
SAF	Social Affinity Filtering Algorithm
$ISAF$	Interaction Social Affinity Features
$ASAF$	Activity Social Affinity Features
Symbols	**Meaning**
$\mathbf{R}$	User-Item preference data
$\mathbf{X}$	Users' social signals (interactions/activities)
ϕ	Social Affinity Features
H	Conditional Entropy

Table 3.1: Symbols and Acronyms used for Social Affinity Filtering.

In this work, we take a departure from the existing approaches and formulate a Social-CF algorithm, *Social Affinity Filtering (SAF)*, using convex linear models that directly leverages fine-grained social signals for the recommendation. Further, we will also show how leveraging social signals can mitigate the cold-start problem.



http://www.youtube.com/watch?v=nLKOgTG8RMs

Zoë Keating: Radical Cello
Cellist Zoë Keating uses a cello and a small box of electronics to create a one-woman avant-garde orchestra. A former member of the cello-rock trio Rasputina...

Rate this Recommendation: [✓ Not Rated] Like Dislike

Figure 3.1: A typical recommender system uses only "like/dislike" data for recommendation, whereas, social recommender systems seek to leverage users' social network data to predict their preferences.

3.3 Social affinity filtering

In this section, we define various social signals and social features, namely *social affinity features*. We then formulate Social-CF as a linear classification problem, *Social Affinity Filtering (SAF)*, that leverages fine-grained *social affinity features*. The key idea behind SAF is that the fine-grained social signals are predictive of users' preferences. For instance, user u who has liked *Justin Bieber Facebook page* might have similar taste as other *Justin Bieber* fans. Now, we proceed to formalising *Social Affinity Filtering (SAF)*. In Table 3.1, we summarise the additional symbols and acronyms defined in this chapter.

3.3.1 Social signals

In this work, we define two different types of social signals in a social network. We use the term *interactions* and *activities* to refer to the range of user-user and user-community action, respectively.

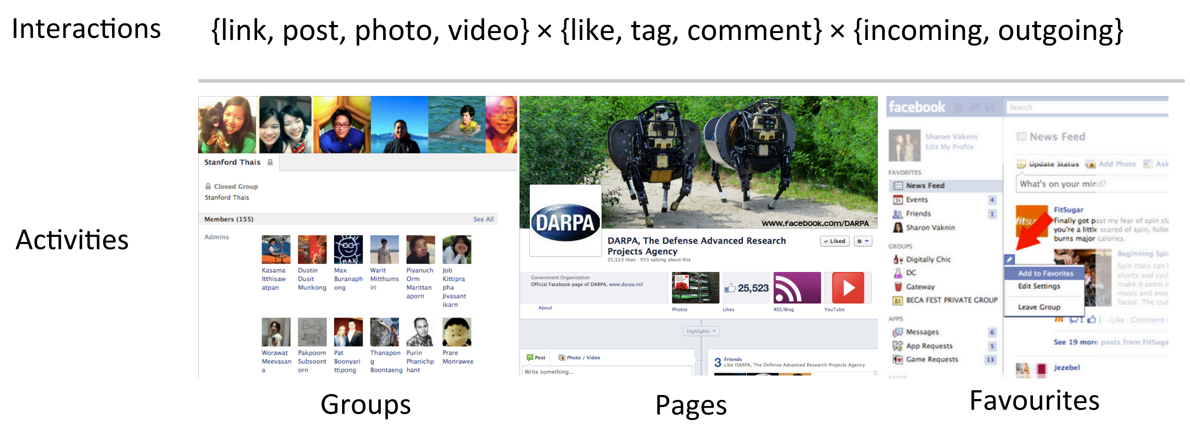

Interactions
{link, post, photo, video} × {like, tag, comment} × {incoming, outgoing}
Activities
Groups
Pages
Favourites

Figure 3.2: Overview of *Interactions* and *Activities* in Facebook social network.

Interactions

Interactions describes communication between users u and v in a social network. In the context of Facebook, it can be broken down into three dimensions:

- **Modality:** (4 possibilities) User u can interact with another user v via *links, posts, photos* and *videos* that appear in either user's timeline. We define a modality set as

$$\mathcal{M} = \{links, posts, photos, videos\}.$$

- **Action type:** (3 possibilities) A user u can *comment* on or *like* user v's item. He/she can also *tag* user v on an item, often indicating that user v is present when the content is created (for photo/video/post), or to explicitly raise user v's attention for a post — with one exception in Facebook that u cannot tag a link with users. We define an action type set as

$$\mathcal{A} = \{comment, like, tag\}.$$

- **Directionality:** (2 possibilities) We look at *incoming* and *outgoing* interactions, i.e., if user u comments on, tags, or likes user v's item, then this is an *outgoing* interaction for u, and an *incoming* interaction for v. Hence, we have directionality set as

$$\mathcal{D} = \{incoming, outgoing\}.$$

  Although high correlation between *incoming* and *outgoing* interactions has been observed [Saez-Trumper et al., 2011], whether interaction direction affects user preferences differently is still an open question we wish to answer in this work.

Based on above defined interaction types, we define *Interaction-Classes* as:

$$Interaction\text{-}Classes := \mathcal{M} \times \mathcal{A} \times \mathcal{D}.$$

Overall there are 22 possible interaction types, namely the cross-product of modalities, actions and directions, minus the special cases of *link-tag-*$\{$*incoming, outgoing*$\}$ since links cannot be tagged.

For each $k \in$ *Interaction-Classes*, we define *Interaction-Features* $\mathbf{X}^k \in \mathbb{R}^{m \times m}$ as:

$$\mathbf{X}^k_{uv} = \begin{cases} 1 & \text{user } v \text{ has had interaction } k \text{ with } u \\ 0 & \text{otherwise.} \end{cases}$$

Activities

Activities are user interactions with Facebook communities like groups, pages, and favourites. We discuss each in turn

- **Groups** on Facebook [1] are analogous to real-world community organisations. They allow users to declare membership and support people to organise activities, to post related content, and to have recurring discussions about them. Examples of groups include *Stanford Thai* (Fig 3.2 left), or *Harvard Debate Club*. We denote set of all groups with $\mathcal{E}^{(g)}$.

- **Pages** on Facebook [2] are analogous to the homepages of people, organisations and events on the world-wide-web. They are publicly visible, and users can subscribe to the updates on the page, and also engage in discussions. Example pages include *DARPA* (an organisation, Fig 3.2 middle), or *Beyonce* (a singer). We denote set of all pages with $\mathcal{E}^{(p)}$.

- **Favourites** are analogous to bookmarks (on physical books or on the web browser). They are a user-created list containing various items such as Facebook apps, books, music, and many other types of items (even pages) to indicate their interest. Example favourites include *Big Bang Theory* (TV series), or *FC Barcelona* (soccer club). Fig 3.2 right shows a Facebook screenshot when a user adds a favourite. [3]. We denote set of all favourites with $\mathcal{E}^{(f)}$.

Based on the above acitivity types, we define

$$\textit{Activity-Groups} = \{groups, pages, favourites\}.$$

For each $k \in$ *Activity-Groups*, we define *Activity-Features* $\mathbf{X}^k \in \mathbb{R}^{m \times d}$ as:

$$\mathbf{X}^k_{ua} = \begin{cases} 1 & \text{user } u \text{ has taken part in activity } a \text{ of the } k \textit{ Activity-Group} \\ 0 & \text{otherwise} \end{cases}$$

where $a \in \mathcal{E}^{(k)}$.

[1]From Facebook Blog: http://www.facebook.com/blog/blog.php?post=324706977130, “Groups are the place for small group communication and for people to share their common interests and express their opinion. Groups allow people to come together around a common cause, issue or activity to organise, express objectives, discuss issues, post photos and share related content.”

[2]From Facebook Blog: (http://www.facebook.com/blog/blog.php?post=324706977130 “Facebook Pages enable public figures, businesses, organisations and other entities to create an authentic and public presence on Facebook. Facebook Pages are visible to everyone on the Internet by default. Facebook users can connect with these Pages by becoming a fan and then receive their updates and interact with them.”

[3]According to Facebook Blog, (https://www.facebook.com/help/232262810142682 “Facebook facilitates a wide variety of user selected favourites (Activities, Favorite Athletes, Books, Interests, Movies, Music, Sports, Favorite Teams, Television). These favourites allow a user to associate themselves with other people who share their same favourite tendencies.”

3.3.2 Social affinity features from social signals

With *Interaction-* and *Activity-features* now defined, we use them to build features for a classification-based approach to social recommendation that we term *Social Affinity Filtering (SAF)*. In SAF, our goal is to predict $\mathbf{R}_{ui}$ for user u and item i. For this task, we use interaction and activity features to define corresponding *Social Affinity Features* as proxies for $\mathbf{R}_{ui}$. Formally, we define such features as follows:

- **Interaction social affinity features (ISAFs)**: We define ISAF, ϕ^k_{ui}, for user u, item i and interaction $k \in$ *Interaction-Classes* as :

  $$\phi^k_{ui} = \sum_{v \in \{U-u\}} \mathbf{R}_{vi} \cdot \mathbf{X}^k_{uv} \tag{3.1}$$

  In this work, we binarise ϕ^k via $[\![\phi^k_{ui} > 0]\!]$. In short, ϕ^k_{ui} is 1 if any user sharing interaction k with u liked i. Here, v is implicitly limited to u's Facebook friends (with whom u may interact).

- **Activity social affinity features (ASAFs)**: We define ASAF, ϕ^a_{ui}, for user u, item i and $k \in$ *Activity-Group* as :

  $$\phi^a_{ui} = \mathbf{X}^k_{ua} \cdot \sum_{v \in \{U-u\}} \mathbf{R}_{vi} \cdot \mathbf{X}^k_{va} \tag{3.2}$$

  where $a \in \mathcal{E}^{(k)}$.

  Like *ISAF*, we binarize ϕ^a_{ui} via $[\![\phi^a_{ui} > 0]\!]$. In short, ϕ^a_{ui} is 1 if both u and some other v are a member of the activity a and v has liked i. Here, v may range over all Facebook users, i.e., v need not be a friend of u to share the same public activity k.

The choice of binary ISAFs and ASAFs is mainly due to the superior performance compared to non-binary features in our experiments.

3.3.3 Recommendation model

Given ISAFs and ASAFs, we combine both features to construct feature vector ϕ_{ui}. *Social Affinity filtering (SAF)* is then simply a linear classification model

$$\mathcal{F} : \phi_{ui} \rightarrow \mathbf{R}_{ui}$$

Given a dataset of historical observations $D = \{\phi_{ui}, \mathbf{R}_{ui}\}$, we can *train* $\mathcal{F}$ using any existing classification method; in this work we consider linear classifiers trained by an SVM, logistic regression, or naïve Bayes. For *prediction*, given user u and item i, we build the feature vector $\mathbf{x_{ui}}$ and predict $\mathbf{R}_{ui} = \mathcal{F}(\mathbf{x_{ui}})$ using the trained classifier $\mathcal{F}$.[4] In Fig 3.3, we overview the SAF training data.

3.3.4 Model analysis

- SAF is equivalent to weighted nearest centroid algorithm, where weights are learned via linear models.

[4]Since most classification methods provide a score (or probability) of a classification, we can also generate the top-n item recommendations for a user u by sorting items by their score.

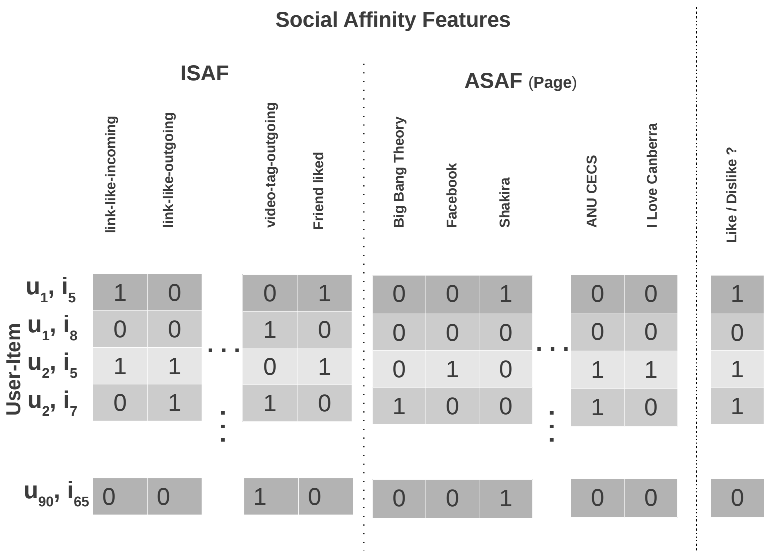

Social Affinity Features
ISAF
ASAF (Page)
link-like-incoming
link-like-outgoing
video-tag-outgoing
Friend liked
Big Bang Theory
Facebook
Shakira
ANU CECS
I Love Canberra
Like / Dislike ?
User-Item
u1, i5
u1, i8
u2, i5
u2, i7
u90, i65

Figure 3.3: SAF training data example: each row corresponds to a training data sample for a specific user-item pair (u, i) for which the prediction target ϕ_{ui} is observed (last column). All other columns represent the value of ISAF or ASAF features evaluated relative to the (u, i) label of each row. All columns are binary-valued $(0, 1)$.

To better understand the SAF recommendation model, we analyse SAF using activity features, ϕ^a_{ui}, for an activity k, as defined in Equation 3.2. The linear model learns to predict

$$\begin{aligned}\hat{\mathbf{R}}_{ui} &= \sum_a \phi^a_{ui} \cdot \mathbf{w_a} \\ &= \sum_a \left(\mathbf{X}^k_{ua} \cdot \sum_{v \in \{U-u\}} \mathbf{R}_{vi} \cdot \mathbf{X}^k_{va} \right) \mathbf{w_a} \\ &= \sum_a \left(\mathbf{X}^k_{ua} \cdot \mathbf{w_a} \cdot \sum_{v \in \{U-u\}} \mathbf{R}_{vi} \cdot \mathbf{X}^k_{va} \right)\end{aligned} \tag{3.3}$$

where $\mathbf{w_a}$ represent learned weights on activity, say pages. Intuitively, the model can be understood as performing linear regression on a feature representation, ϕ^a_{ui} that is joint for users and items.

Further, Equation 3.3 can be written in a matrix form as

$$\hat{\mathbf{R}} = \mathbf{X} diag(\mathbf{w_a}) \mathbf{X}^T \mathbf{R} \tag{3.4}$$

Equation 3.4 corresponds to metadata based *weighted nearest centroid classifier* [Manning et al., 2008], where the weights are learned via linear regression. However, as a downside, the proposed model shares the weights, $\mathbf{w_a}$, for all users. Hence, the proposed method might be lacking on personalisation.

3.4 Experiment and evaluation

In this section, we discuss the dataset used for experiments, experimentation methodology, and report a detailed analysis of the proposed model.

	App Users	**Ego network of App Users**
Male	85	20,840
Female	34	17,032
Total	119	37,872

Table 3.2: App user demographics. The *ego network* is the friend network of the App users.

App Users	**Tags**	**Comments**	**Likes**
Post	7,711	22,388	15,999
Link	—	7,483	6,566
Photo	28,341	10,976	8,612
Video	2,525	1,970	843
Ego network of App Users	**Tags**	**Comments**	**Likes**
Post	1,215,382	3,122,019	1,887,497
Link	—	891,986	995,214
Photo	9,620,708	3,431,321	2,469,859
Video	904,604	486,677	332,619

Table 3.3: Statistics on user *interactions*.

	App Users	**Ego Network of App Users**
Groups	3,469	373,608
Page Likes	10,771	825,452
Favourites	4,284	892,820

Table 3.4: Statistics on user *actions*, counted for *Groups, Pages* and *Favourites* over the App users and their ego network.

3.4.1 Data description

We built a Facebook App that recommend links to the users everyday, where the users may give their feedback on the links indicating whether they *liked* or *disliked* it. For the app users, we also collected the detailed user interaction and activity history available through the Facebook Graph API. The data collection was performed with full permission from the user and in accordance with an approved Ethics Protocol #2011/142 from the Australian National University.

Our App requested to collect information on profiles (including activity memberships) and timelines (interactions) for the App users *and* their friends as required by Sec 3.3.2. With such expressive permissions, many potential users were hesitant to install the App — after an intensive one month user drive at our University, we were able to attract 119 App users allowing us to collect activity and interaction data for a combined 37,872 users.[5]

We summarise basic statistics of the data in Tables 3.2–3.5. Table 3.2 presents user and friend demographics. Table 3.3 summarises the number of records for each item modality (row) and action (column) combination. Table 3.4 shows the group membership, page like and favourite counts for users.

Our App recommends three links to App users each day, which the users may optionally like or dislike. Recommended links are harvested from *both* friends' and non-friends' timelines. We display

[5]The issue of low App user uptake with such expressive App permissions underscores the importance of identifying the *minimal* set of permissions to obtain good recommendation performance — a question we address in our subsequent analysis.

	Friend recommendation	**Non-Friend recommendation**
Like	1392	1127
Dislike	895	2111

Table 3.5: Dataset breakdown of prediction target $like(u, i)$ by the source of the link (Friend/Non-friend) and rating.

only three links per day in order to avoid rank-bias with preferences; each link could be independently rated. Table 3.5 shows App user link preference statistics.

All subsequent experiments use offline batch data stored and analysed *after* a four month data collection period.

3.4.2 Model comparision

In this section, we compare novel SAF-based methods with a variety of (social) collaborative filtering baselines:

1. Most Likely Class Constant Predictor (Const)
2. Nearest Neighbor (NN), as defined in Equation 2.5
3. Matrix Factorisation (MF) [Salakhutdinov and Mnih, 2008b], as defined in Equation 2.8
4. Social Matchbox (SMB) [Noel et al., 2012], as defined in Equation 2.25

Here, Const serves as a lower bound on performance, NN and MF are two well-known state-of-the-art *non-social* collaborative filtering algorithms, and SMB is a state-of-the-art *social* collaborative filtering algorithm employing matrix factorisation with social regularisation.

Among the novel SAF methods, we analyse four different sets of social affinity features:

1. Interaction Social Affinity Features (ISAF)
2. Activity-based Social Affinity Features (ASAF) for
   (a) Group Memberships
   (b) Page Likes
   (c) Favourites

Furthermore, for these four classes of features, we train one of three classifier types, leading to the following classes of SAF recommenders evaluated in our experiments:

1. Naïve Bayes (NB-ISAF, NB-ASAF)
2. Support Vector Machines (SVM-ISAF, SVM-ASAF)
3. Logistic Regression (LR-ISAF, LR-ASAF)

NB uses a standard Naïve Bayes implementation, SVM and LR are both implemented using *LIBLINEAR* [Fan et al., 2008].

In all experiments, we report average classification accuracy (fraction of correct classifications in held-out test data) using 10-fold cross validation and provide standard error bars corresponding to 95% confidence intervals.

3.4.3 Performance analysis

Fig 3.4 compares the above baselines and SAF algorithms. In all of these experiments, SAF variants performed statistically significantly better than the best baseline (SMB), except for NB-ASAF which we conjecture is due to violation of feature independence assumptions that become more pronounced as the number of features increases (n.b., NB-ISAF uses 22 features while NB-ASAF uses 1000's of features).

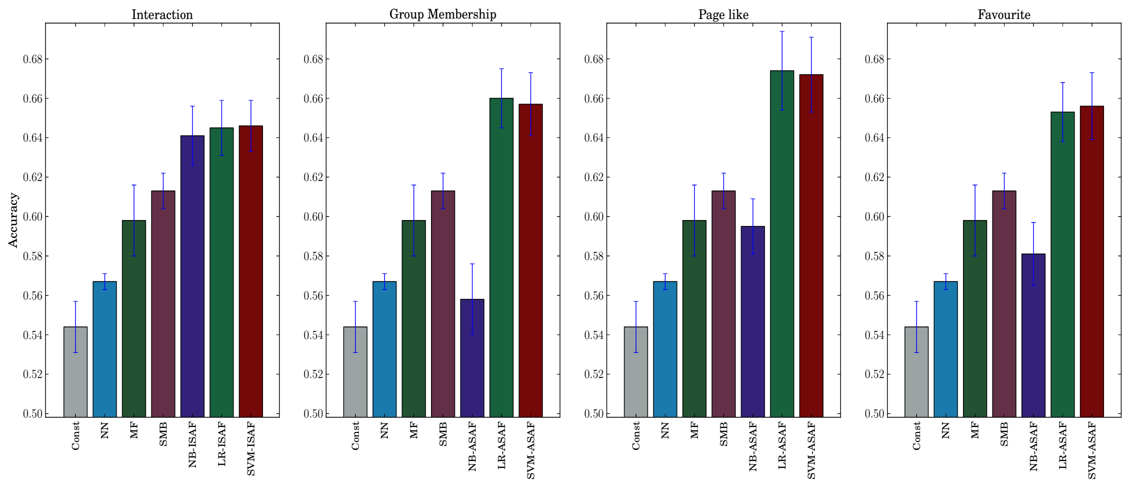


Figure 3.4: Comparision of a simple baseline (Const), two collaborative filtering baselines (NN and MF), a social collaborative filtering baseline (SMB) and novel SAF recommenders using different feature sets (one ISAF and three ASAF sets) and classifiers (NB,LR,SVM). The best SAF-based model (LR-ASAF) — for Page likes — significantly outperforms all baselines by at least 6%. Combining all four feature sets (not shown) does not lead to improvement over Page likes features alone.

In terms of the best recommenders, we observe that LR-ASAF and SVM-ASAF perform comparably to each other and learn quite well despite the large size of this ASAF feature set. Overall, *LR-ASAF performs 6% better than the best baseline for page likes*. We combined all four features sets in a fifth experiment (not shown) and remark that none of NB, LR, or SVM with all features outperformed LR-ASAF with just page likes. We also note that all ISAF variants statistically significantly outperform all (social) collaborative filtering baselines. Hence, w.r.t. this Facebook dataset, we conclude that (a) *SAF with any available feature set* is sufficient to outperform existing (social) collaborative filtering baselines and that (b) if one wanted to minimise the permissions an App requests then it seems *SAF with page likes* alone is sufficient to outperform all other feature sets (alone or combined).

It is important to consider why ASAFs outperform ISAFs. We conjecture the reasons for this are quite simple: ISAFs span across *friends* of user u whereas ASAFs span across the all users, independent of u's friends. Hence, given the relative sparsity of friend-only data in Facebook compared to the greater Facebook population (at least the subset of the population the App collected) and also the relative number of ISAFs compared to ASAFs, Among ASAFs, page likes are the most predictive followed by group membership and favourites. This reinforces our conjecture that data sparsity can hurt SAF since we note from Table 3.4 that page likes are more prevalent than groups and favourites.

Comparing SAF to the state-of-the-art in social collaborative filtering as represented by Social Matchbox (SMB) [Noel et al., 2012], we observe that SAF consistently outperforms it. We note that the key difference of SAF vs. SMB is that SAF exploits the predictiveness of fine-grained interactions and activities, whereas most social collaborative filtering approaches [Cui et al., 2011, Li and Yeung, 2009b, Ma et al., 2009b, 2008b, 2011b, Noel et al., 2012, Yang et al., 2011b] instead collapse the diverse set of interactions into a single aggregate statistic for each pair of users, as discussed in Section 2.4.4. The performance of SAF-based recommenders suggests that the aggregation of all social information into aggregate statistics (without learning which interactions or activities are most informative) may not distinguish informative parts of the social signal from the noise.

On the computational side, we remark that SAF is implemented as a simple linear classifier that can be used in conjunction with a variety of classification methods (e.g., naive Bayes, logistic regression, SVM) and online training algorithms amenable to real-time, big data settings. Furthermore, the linear classification methods used in SAF admit global convex optimisation w.r.t. a variety of training objectives (e.g., log loss in logistic regression, or hinge loss in SVMs) unlike (social) collaborative filtering approaches based on matrix factorisation that use non-convex objectives and lack training optimality guarantees.

3.4.4 Cold-Start evaluation

Many collaborative filtering algorithms (e.g., NN and MF) suffer from the user *cold-start* problem, as discussed in Section 2.4.3 .These algorithms cannot perform better than the constant (most likely class) predictor since they have no way of generalising to a new unseen user. Since SAF trains a single model for all users and does *not* require a user's preferences in order to recommend for them, SAF can be used in a *cold-start* setting to recommend for users *without* expressed item preferences as long as those users have interactions or shared activities with users who have expressed item preferences.

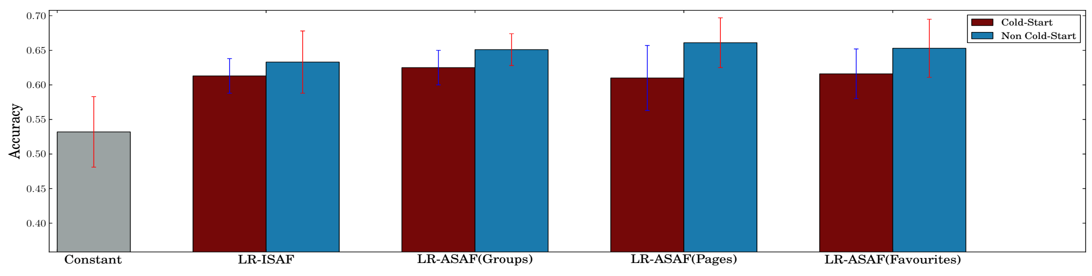


Figure 3.5: Cold-start evaluation of SAF: accuracy evaluated on cold-start users outperforms the most likely class (Constant) predictor baseline and is somewhat comparable to the non cold-start case when all test user data is *not* withheld from training.

To quantitatively evaluate the cold-start performance of SAF, we run 10-fold cross validation with specially constructed folds. For *cold-start* evaluation, in each fold, we hold out a random 10% subset of the users for testing and train on the remaining 90% of users. In the test set, we further hold out 30% of the test user data. In the *non cold-start* evaluation, we test on the same data as in the *cold-start* evaluation, but we add in the 30% of held-out test user data to the *cold-start* training set thus allowing the *non cold-start* setting to train on some of the test user data. In Fig 3.5, we clearly see that the accuracy[6] of the SAF predictor for cold-start is significantly better than the baseline Constant predictor. Furthermore, the accuracy of the cold-start predictor is actually comparable to the non cold-start predictor, indicating that SAF exhibits strong cold-start performance.

Table 3.6: Conditional entropy of various interactions (lower conditional entropies are more informative). We observe that interactions on videos are more informative than other modalities (link, post, photo), tagging is marginally more informative than commenting and liking, and outgoing interactions are slightly more informative than incoming ones. Breaking down the analysis by modality-direction and action-direction reveals finer-grained distinctions.

Modality (ϕ)	$H(R\|\phi = 1)$
video	0.850
link	0.915
post	0.918
photo	0.926

Action Type (ϕ)	$H(R\|\phi = 1)$
tags	0.920
comments	0.921
likes	0.924

Direction (ϕ)	$H(R\|\phi = 1)$
outgoing	0.928
incoming	0.935

Modality-Direction (ϕ)	$H(R\|\phi = 1)$
tags-outgoing	0.885
likes-outgoing	0.885
tags-incoming	0.900
likes-incoming	0.902
comments-outgoing	0.908
comments-incoming	0.912

Action-Direction (ϕ)	$H(R\|\phi = 1)$
photo-outgoing	0.857
video-outgoing	0.863
link-outgoing	0.895
link-incoming	0.896
post-incoming	0.902
post-outgoing	0.906
video-incoming	0.915
photo-incoming	0.921

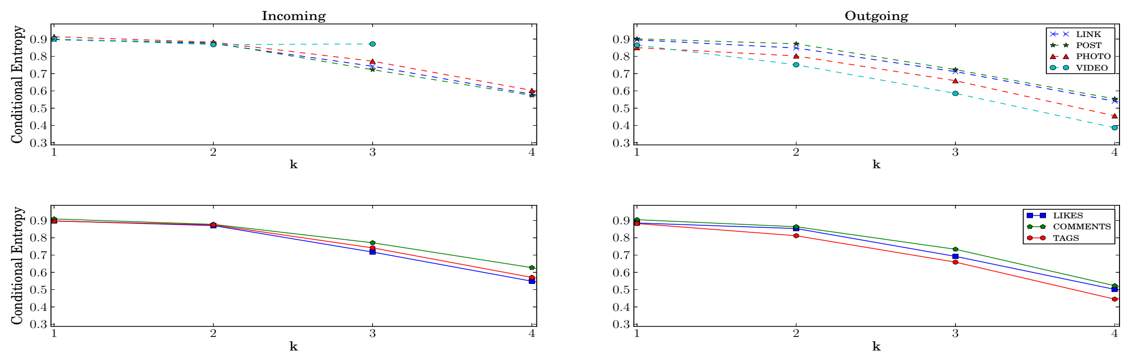


Figure 3.6: Conditional Entropy of modalities/activities for incoming/outgoing interactions vs. item liked by at least k friends. Increasing k generally has a stronger influence on informativeness than other features of interactions.

3.4.5 Interaction analysis

In this section we analyse the informativeness of Interaction Social Affinity Features (ISAFs), namely user interactions according to their modality, type, and direction, as described in Section 3.3.1.

A general method for measuring the amount of information that a feature ϕ^k_{ui} provides w.r.t. predicting a user preference $\mathbf{R}_{ui}$ (in this case, just 1 or 0) is to calculate its conditional entropy:

$$H(\mathbf{R}_{ui} = 1|\phi^k_{ui} = 1) = -\sum_{y \in (0,1)} p(\mathbf{R}_{ui} = y|\phi^k_{ui} = 1) \cdot \ln(p(\mathbf{R}_{ui} = y|\phi^k_{ui} = 1)) + \\ p(\mathbf{R}_{ui} = y|\phi^k_{ui} = 0) \cdot \ln(p(\mathbf{R}_{ui} = y|\phi^k_{ui} = 0))$$

Lower conditional entropies generally indicate more informative features.

First we analyse various interactions to better understand what interactions have a high affinity with a user's preferences. To this end, we make a few observations from the conditional entropy analysis of Table 3.6:

- Interaction on *videos* indicates a stronger preferential affinity between users than other modalities (links, posts and photos). We conjecture this is because videos are time-consuming to view and hence users mainly watch the videos of those users whose preferences they share.
- Tagging has a slightly better conditional entropy than commenting and liking, potentially since tagging often results from direct social interaction (appearing in a photo or video together) indicating common interests.
- A user is more likely to share preferences with someone who she initiates the interaction with (outgoing) vs. with someone who initiates the interaction with her (incoming). E.g., we note that while outgoing photo and video interactions are *most* informative in the last table of Table 3.6, it appears that incoming photo and video interactions are *least* informative.

In Fig 3.6 we plot the conditional entropy of modality and action for incoming/outgoing interactions constrained to links liked by at least k friends with the corresponding interaction We note that preference affinity with any interaction increases as more people with the corresponding interaction like the item. E.g., while incoming interactions were not as predictive as outgoing interactions for the same k, we note that higher k on incoming *can be more predictive* than lower k for outgoing. Similar principles hold for modality and action vs. k — a larger k is generally more predictive than the individual variation among modality and action at a fixed k, an exception being the modality-outgoing analysis.

3.4.6 Activity analysis

Now we analyse the informativeness of Activity Social Affinity Features (ASAFs) by looking at the correlation between the size and type of groups, pages and favourites.

Fig 3.7 provides scatter plots of conditional entropies and logistic regression weights vs. activity group size.[7] Both plots show that small activity groups *can* be highly predictive (low conditional entropy or weights that deviate extremely from zero) whereas large groups are *rarely* predictive.

In Fig 3.8 we plot the average conditional entropy of the top 10% of features cumulative up to the size of the activity group given on the x-axis. This graph distinctly shows that the small sizes of groups,

[6]The slight decrease in accuracy for non cold-start case compared to Fig 3.4 is due to the decreased amount of test user data present in the training set for this set of experiments.

[7]Here the size of a *group*, *page* and *favourite* is the number of total users in the activity group. For *pages* and *favourites* this is the total number of Facebook users, whether or not they are in the App users' ego network, while for *groups* only the number of users in the App users' ego network is visible to our App.

Median Informative Favourites by Category				
Books	**Movies**	**Music**	**Television**	**Interests**
Harry Potter series	Forrest Gump	John Lennon	Futurama	Travel
A Song of Ice and Fire	Pretty Woman	U2	Star Trek	Music
Discworld	Napoleon Dynamite	AC/DC	The Trap Door	Literature
Hitchhiker's Guide To The Galaxy	Harry Potter	The Smashing Pumpkins	Drawn Together	Painting
The Hobbit	Toy Story 3	Gotye	Sherlock(Official)	Running
The Magician's Guild	The Godfather	The Rolling Stones	Hitchhiker's Guide to the Galaxy	Sports
Ranger's Apprentice	Mulan	All Axess	Buffy The Vampire Slayer	Films
Cosmos	How to Train Your Dragon	Steve Aoki	South Park	Genetics

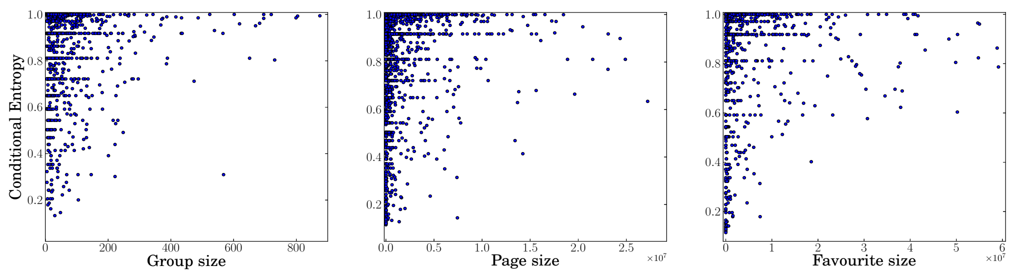
Most Informative Favourites by Category				
Books	**Movies**	**Music**	**Television**	**Interests**
Calvin and Hobbes	Billy Madison	Avascular Necrosis	Metalocalypse	Computers
Tomorrow when the War Began	Team America: World Police	Tortured	Beast Wars	Texas HoldEm
I really like ceilings	Pan's Labyrinth	Elysian	Hey Arnold!	Programming
Angels and demons	Pirates of the Caribbean	Anno Domini	Sherlock	Economics
Magician	Aladdin	Darker Half	Hey Hey It's Saturday	Martial arts
Digital Fortress	Starship Troopers	Hellbringer	Neil Buchanan and Art Attack!	Graphic design
The Bible	Happy Gilmore	Johnny Roadkill	Breaking Bad	Cooking
Interview with the Vampire	Timon and Pumbaa	Aeon of Horus	Red vs. Blue	Klingon language

Table 3.7: (top) Examples of 8 items per Favourite category near the *median* conditional entropy (*median informativeness*). (bottom) Examples of top 8 items with the lowest conditional entropy (*most informative*). A general trend is that more informative favourite category activity tend to be more specialised in appeal, e.g. "Avascular Necrosis" is an informative music group favourite — its members tend to share common preferences — while "John Lennon" and "U2" have a broader audience with more diverse preferences. Interestingly, "Sherlock" appears in both most and median informative table but the median informative is an official page with wide range of fans, whereas the most informative is a duplicate fan page with few fans.

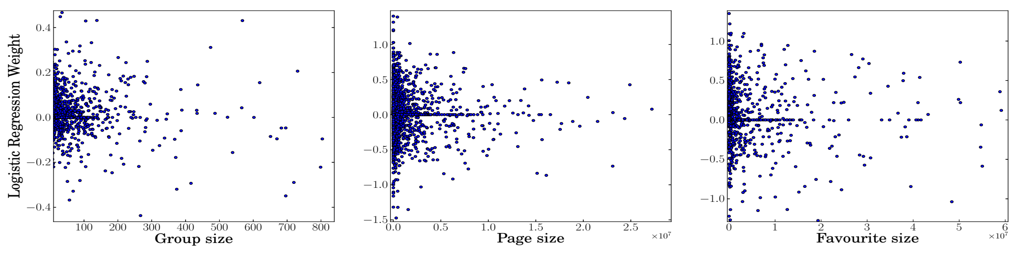
pages and favourites have low average conditional entropy that transitions sharply to a higher average at about 50 for groups and 10^5 for pages/favourites.

We also analyse predictiveness of favourites by categories obtained from the Facebook API in Fig 3.9. While half of category instances in movie, books, or movies with "long-tail" (less popular, specialised) content may not be highly predictive (judging by median informativess), these categories do contain some highly predictive instances (as evidenced by the top two quartiles). On the contrary, highly generic categories (e.g. interests) and those with fewer choices (e.g. sports or fav-teams) tend to be less predictive overall. These observations of Fig 3.9 are also reiterated by the examples provided in Table 3.7 where uninformative favourites tend to have a broad appeal whereas informative favourites generally appear much more specialised. This reinforces the insight of SAF that it is important to learn which activities are predictive.

One might ask how the number of activities a user joins affects recommendation performance. In Fig 3.10, we see that on average, more user activities generally leads to higher accuracy. As an alternative analysis, Fig 3.3 shows performance vs. the number of *active features*, i.e., for any (user,item) recommendation in a row of Fig 3.3, the active features are those that are true (1). Here we see that excessive item popularity among activities hurts the discriminative power of SAF to make good recommendations.



(a)



(b)

Figure 3.7: Conditional entropy vs. size (a); logistic regression feature weights vs size (b). In (a) we observe that the activities with large membership are rarely informative while the most informative activities tend to have low memberships. Similarly in (b) we see that the most predictive features with the largest weights (positive or negative) are concentrated toward activities with small memberships.

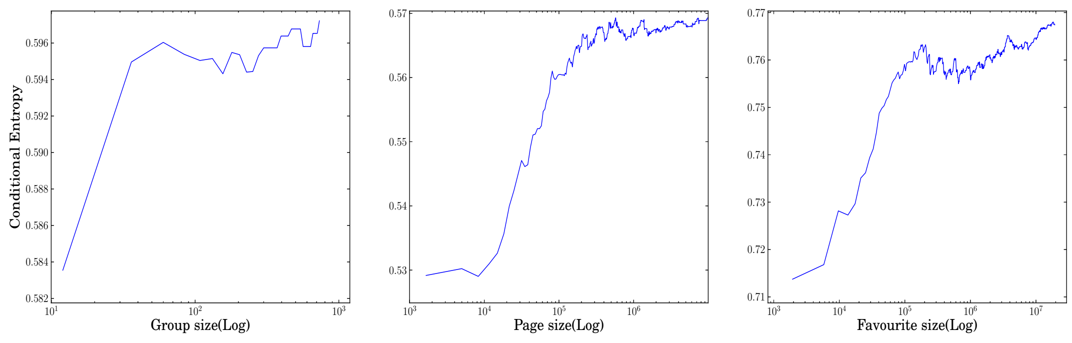


Figure 3.8: Average conditional entropy of top 10% groups, pages and favourite features *cumulative* over the size. Here we see that as we add in larger membership activities, the average informativeness decreases substantially (entropy increases).

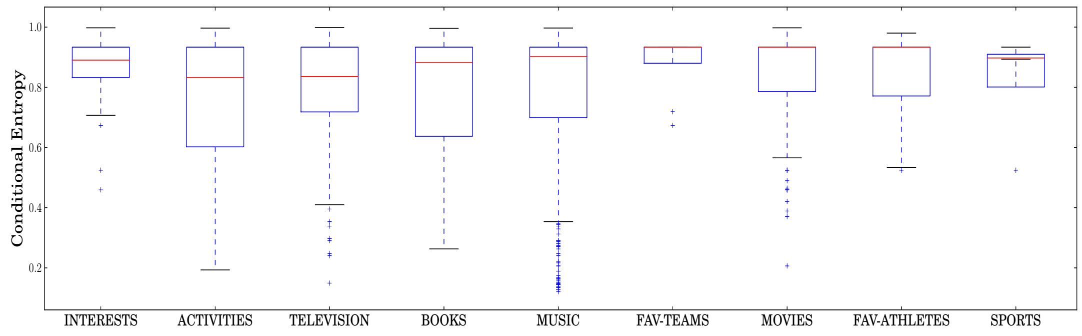


Figure 3.9: Conditional entropy for top 1000 favourites breakdown by categories. While at least half of activity categories with many options like music are not informative (judging by median values), some of the most informative activities are music. This reiterates the point that it is crucial to *learn* which activities are informative rather than aggregating average information.

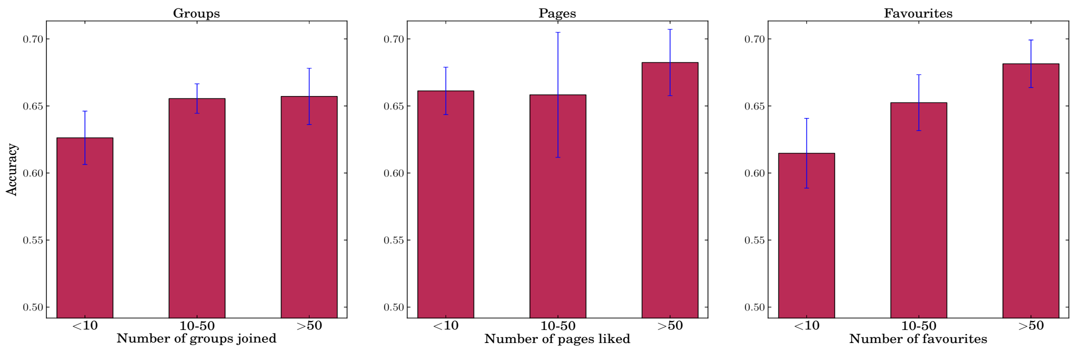


Figure 3.10: Accuracy of the SAF increases as users become more active in social network by joining more groups/pages/favourites. It does not appear too many activities hurts — SAF learns to discriminate when activities are predctive.

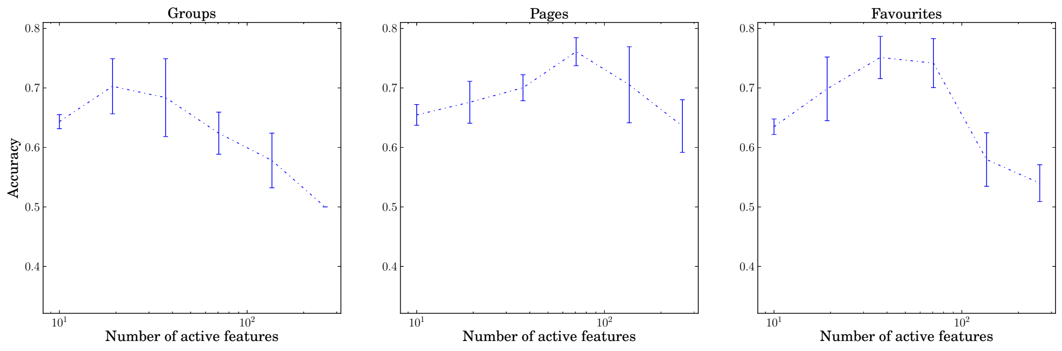


Figure 3.11: Accuracy increases as the number of active features increases, but then, after reaching a certain limit, it starts to decrease, i.e., excessive item popularity among activities hurts the discriminative power of SAF to make good recommendations.

3.5 Related work and discussion

This work relates to many others in inferring user preferences on social and information networks. We structure the discussion into three parts: the first is concerned with the nature and observations on user traits, interactions and diffusion mechanisms; the second is concerned with correlating these user traits and interactions to user preferences and interests; the third is concerned with methods that use these observations for predicting user interest or recommending content on social networks.

The first group of related work studies the nature of user profile, interactions, and diffusion. Profile information and demographics is correlated with user behaviour patterns. Chang *et al* [Chang et al., 2010] showed that the tendency to initiate a Facebook friendship differs quite widely across ethnic groups, while Backstorm *et al* [Backstrom et al., 2011] have additionally showed that female and male users have opposite tendencies for dispersing attention for within-gender and across-gender communication. Two particular measurement studies on Facebook attention [Backstrom et al., 2011, Wilson et al., 2009] have inspired our work. Although the average number of friends for a Facebook user is close to the human psychological limit, known as the Dunbar number [Hill and Dunbar, 2003], the findings concur that a user's attention (i.e., interactions) are divided among a much smaller subset of Facebook friends. [Backstrom et al., 2011] studied two types of attention: communication interaction and viewing attention (e.g. looking at profiles or photos). Users' communication attention is focused on small numbers of friends, but viewing attention is dispersed across all friends. This finding supports our approach of looking at many types of user interactions across all of a user's contact network, as a user's interest is driven by where he/she focuses attention.

The mechanisms of diffusion invite interesting mathematical and empirical investigations. The Galton-Watson epidemics model suits the basic setup of social message diffusion, and can explain real-world information cascade such as email chain-letters when adjusted with selection bias [Golub and Jackson, 2010]. For social diffusions in a one-to-many setting, however, the epidemics model has been less accurate. Ver Steeg *et al* [Ver Steeg et al., 2011] found that online message cascades (on Digg social reader) are often smaller than prescribed by the epidemics model, seemingly due to the diminishing returns of repeated exposure. Romero *et al* [Romero et al., 2011], in an independent study, confirmed the effect of diminishing returns with Twitter hashtag cascades, and further found that cascade dynamics differ across broad topic categories such as politics, culture, or sports. Our observations on user preference on items liked by a number of Facebook friends suggest a large cumulative number of friend preferences is more predictive, although further investigation is needed to pinpoint the effect of diminishing returns on repeated exposures.

The nature of social diffusion seem to be not only democratic [Asur et al., 2011, Bakshy et al., 2011], but also broadening for users [Bakshy et al., 2012]. While influential users are important for cascade generation [Bakshy et al., 2011], large active groups of users are needed to contribute for the cascade to sustain [Asur et al., 2011]. Moreover, word-of-mouth diffusion can only be harnessed reliably by targeting large numbers of potential influencers, confirmed by observations on Twitter [Bakshy et al., 2011] and online ads [Watts and Dodds, 2007]. In a study facilitated by A/B testing on Facebook links, [Bakshy et al., 2012] found that while people are more likely to share the information they were exposed to by their strong ties rather than their weak ties, the bulk of information we consume and share comes from people with different perspectives (weak ties). SAF aims to leverage many of these insights for social recommendation by viewing affinity groups as diffusion channels. Yet, information diffusion and recommendation are distinct problems — while we observed best *recommendation* performance using activities ranging across all Facebook users, the vast majority of *information diffusion* happens within one step from the source node [Goel et al., 2012].

The second group of related work tries to correlate from user interactions to preferences and tie strength. Saez-Trumper et al [Saez-Trumper et al., 2011] found that incoming and outgoing activities are highly correlated on broadcast platforms such as Facebook and Twitter, and such correlation does not hold in one-to-one mode of communication such as email. Multiple studies have found that online interactions tend to correlate more with interests than with user profile. Singla *et al* [Singla and Richardson, 2008] found that users who frequently interact (via MSN chat) tend to share (web search) interests. Anderson *et al* [Anderson et al., 2012] concluded that the level of user activities correlate with the positive ratings that they give each other, i.e., it is less about what they say (content of posts) but more about who they interacted with. Such findings echo those by Brandtzag [Brandtzg and Nov, 2011] that real-world interactions (e.g., appearing in the same photo evidenced by tagging) further strengthens friendship on Facebook, while virtual interactions reveal interests. Furthermore, ratings of real-world friendship strength and trust [Gilbert and Karahalios, 2009] seems to be better predicted from the intimacy, intensity, and duration of interactions, than from social distance and network structure. Our work is not only inspired by these observations, we also quantify the strength of correlations of user interest with a large variety of user affinities – namely, activities, and group preferences in different categories.

The third group of related work is concerned with using social network and behaviour information for recommendation. Matrix factorisation, as discussed in Section 2.3.2, is one of the prevailing approaches for recommender systems [Koren et al., 2009, Ma et al., 2008b]. Recent advances include extending matrix factorisation to user social relation [Cui et al., 2011, Yang et al., 2011a, Ma et al., 2011a, Li and Yeung, 2009a, Ma et al., 2008a, Noel et al., 2012], as discussed in Section 2.4.4. These systems have shown very promising performance across a range of problems, but they all collapse social affinity (fine-grained interactions and group affinity) into one or a very low-dimensional representation. The point of departure of this work is to explore the predictive power of fine-grained social information.

3.6 Conclusion

In this Chapter, we formulated a novel Social-CF algorithm, *Social Affinity Filtering (SAF)*, using linear models that directly leverages fine-grained signals from a social network. Further, we demonstrated the effectiveness of linear models for Social-Cf and substantial predictive power of fine-grained social features in predicting users' preferences.

However, as a fundamental limitation, *Social Affinity Filtering* learns a global model for the users, as discussed in Section 3.3.4, and might be lacking personalisation. Further, the proposed method requires feature engineering, which requires human supervision. In the following Chapter, building upon the key insights i.e. superiority of the linear models and social features for the recommendation, we formulate a personalised linear model that is capable of leveraging social signals for cold-start recommendation.

Chapter 4

Linear Models for Cold-Start Recommendation

In Chapter 3, we demonstrated the efficacy of linear models for Social-CF, and showed that social information, such as page likes, are highly predictive of users' preferences. In this Chapter, we investigate *cold-start recommendation*. The user cold-start problem concerns the task of recommending items to users who have not previously purchased or otherwise expressed preferences towards any item under consideration. Similarly, item cold-start problem of recommending new items to the users In this Chapter, we focus on formulating a personalised linear model for cold-start users[1] in OC-CF setting that leverages users' metadata, such as page likes.

4.1 Problem setting

In this Chapter, we are concerned with recommending a ranked list of items to the users. We have historical preference data $\mathbf{R} \in \{0,1\}^{m \times n}$ that consists of users' purchase history, where $\mathbf{R}_{ui} = 1$ means that user u purchased item i as in one-class collaborative filtering setting. We wish to train a model using warm-start users' data and test on cold-start users. Formally, let $U^{(\text{tr})}$ denote *warm-start* users with at least one purchase, and the $U^{(\text{te})}$ *cold-start* users without any preference data. Our interest is in producing $\hat{\mathbf{R}}^{(\text{te})} \in \mathbb{R}^{|U^{(\text{te})}| \times n}$, the recommendation matrix for the cold-start users.

Personalised recommendation for cold-start users is impossible from $\mathbf{R}$ alone. But given a metadata matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$, with $\mathbf{X}^{(\text{tr})}, \mathbf{X}^{(\text{te})}$ being the metadata for the warm- and cold-start users respectively, we can leverage such data to learn users' preferences. For concreteness, we will think of $\mathbf{X}_{up}$ being whether or not user u "likes" a page p, though $\mathbf{X}$ could equally reflect e.g. users' group memberships, friend circles, *et cetera*. The key idea that enables cold-start prediction is that users metadata such as page likes are predictive of users' preferences, as observed in Chapter 3.

4.2 Background

In Section 2.4.3, we briefly discussed algorithms for cold-start recommendation. Here, we will discuss cold-start algorithms in detail. There have been two broad strands of such work extending neighbourhood and matrix factorisation respectively.

[1]The proposed methods directly translate to item cold-start problem.

4.2.1 Neighbourhood based cold-start CF

As discussed in Section 2.4.3, neighbourhood based cold-start CF [Zhang, Zi-Ke et al., 2010, Sahebi and Cohen, 2011] uses metadata to compute user-user or item-item similarity to make recommendations. For user cold-start, the similarity between cold-start and warm-start users, $\mathbf{S}_{uv}$, where $u \in U^{(\text{te})}$ and $v \in U^{(\text{tr})}$, is computed from the metadata $\mathbf{X}$ using predefined similarity metrics such as *cosine similarity*. The recommendation matrix is defined as

$$\hat{\mathbf{R}}^{(\text{te})} = \mathbf{S} \cdot \mathbf{R}^{(\text{tr})}. \tag{4.1}$$

4.2.2 MF based cold-start CF

There are two popular approaches to MF based CF for cold-start recommendation, as discussed in 2.4.3,

Co-Factorisation

Co-Factorisation based cold-start CF (CMF) [Krohn-Grimberghe et al., 2012] finds a latent subspace for users that is jointly predictive of both user preferences and side-information. It minimises

$$\min_{\mathbf{A},\mathbf{B},\mathbf{Z}} ||\mathbf{R} - \mathbf{A}\mathbf{B}||_F^2 + \mu||\mathbf{X} - \mathbf{A}\mathbf{Z}||_F^2 + \lambda_A||\mathbf{A}||_F^2 + \lambda_B||\mathbf{B}||_F^2 + \lambda_Z||\mathbf{Z}||_F^2 \tag{4.2}$$

where $\mathbf{A} \in \mathbb{R}^{m\times k}, \mathbf{V} \in \mathbb{R}^{k\times n}$, and $\mathbf{Z} \in \mathbb{R}^{k\times d}$ for some *latent dimensionality* $k \ll \min(m, n)$. We then predict

$$\hat{\mathbf{R}}^{(\text{te})} = \mathbf{A}^{(\text{te})}\mathbf{B}. \tag{4.3}$$

The intuition is that $\mathbf{A}$ is jointly predictive of rating and metadata, and generalise for cold-start recommendation.

Feature mapping

Feature mapping based cold-start recommendation is a two-step model. Here, the first step is to model the warm-start users by $\mathbf{R}^{(\text{tr})} \approx \mathbf{A}^{(\text{tr})}\mathbf{B}$, with latent features $\mathbf{A}^{(\text{tr})}, \mathbf{B}$ as before. The second step is to learn a mapping between the metadata $\mathbf{X}^{(\text{tr})}$ and latent features $\mathbf{A}^{(\text{tr})}$ using e.g. linear regression,

$$\mathbf{A}^{(\text{tr})} \approx \mathbf{X}^{(\text{tr})}\mathbf{T} \tag{4.4}$$

for $\mathbf{T} \in \mathbb{R}^{d\times k}$. We then estimate the cold-start latent features $\mathbf{A}^{(\text{te})} = \mathbf{X}^{(\text{te})}\mathbf{T}$, and use Equation 2.19 for prediction, and we refer this model as *BPR-LinMap* [Gantner et al., 2010]. Similarly, [van den Oord et al., 2013] used convolutional neural network to map item features to latent factors for item cold-start.

Other models

There has been a significant amount of research in leveraging side-information for MF models, such as:

- **Regression Based Latent Factor Model (RLFM)** [Agarwal and Chen, 2009] leverages users' and items' metadata by combining regression and latent factor model for predicting users' preferences:

$$\hat{\mathbf{R}}_{ui} = b_u + b_i + \phi(\mathbf{X}_u, \mathbf{X}_i) \cdot \mathbf{w} + \mathbf{A}_u^T\mathbf{B}_i$$

where ϕ is a feature generating function; b_u and b_i are user and item bias respectively. For the cold start user u, preference is predicted as:

$$\hat{\mathbf{R}}_{ui} = b_u + b_i + \phi(\mathbf{X}_u, \mathbf{X}_i) \cdot \mathbf{w}$$

- **Matchbox** incorporates users' and items' metadata in MF model, as discussed in Equation 4.5, by predicting recommendation matrix as

$$\hat{\mathbf{R}}(\theta) = (\mathbf{A}\mathbf{X}^U)^T(\mathbf{B}\mathbf{X}^I) \tag{4.5}$$

In principle the above methods can produce cold-start recommendations. However, the key limitation is that they are formulated for rating prediction problem and don't scale in one-class collaborative filtering (OC-CF) setting. In particular, we cannot use Alternate Least Squares(ALS), due to lack of analytical solution, which is a key for large-scale OC-CF. Naively treating unobserved as 0 results into a large number of matrix entries making gradient based optimisation impractical for real world scenarios.

4.2.3 Limitations of existing approaches

The neighbourhood based models are simple and intuitive. However, they don't account for correlation amongst the features. These models risk *underfitting*, as the features can be highly correlated, such as pages liked by the user.

As discussed in Section 2.3.2, MF based methods are non-convex, and hence susceptible to local minima. The feature mapping algorithm is a two step algorithm, and hence highly susceptible to error propagation. Further, while it accounts for feature correlations, it risks *overfitting* for high dimensional $\mathbf{X}$ while learning the mapping function. This was indeed observed in the high-dimensional experiments in [Gantner et al., 2010]. Similarly, CMF, has limited scalability by virtue of requiring tuning of at least four hyperparameters $(\mu, \lambda_A, \lambda_B, \lambda_Z)$. In Table 4.1, we summarise the properties of various cold-start models.

Method	**Feature Correlation**	**Convex**	**Scalability**
NEIGHBORHOOD	×	-	✓
CMF	✓	×	×
BPR-LINMAP	✓	×	×
GEN-NEIGHBORHOOD	×	-	✓
LoCo	✓	✓	✓

Table 4.1: Comparison of various cold-start methods whether they handle correlated features, yield optimal solution and are scalable.

4.3 Linear models for cold-start recommendation

In this section, we formulate a cold-start recommendation model that exploits users' social side information $\mathbf{X}$. Perhaps the most natural form of cold-start model that leverages $\mathbf{X}$ is the *linear content-based model* as shown below:

$$\hat{\mathbf{R}}^{(\text{te})} = \mathbf{X}^{(\text{te})}\mathbf{W}, \tag{4.6}$$

where $\mathbf{W} \in \mathbb{R}^{d \times n}$. The most apparent approach to learn $\mathbf{W}$ is by performing multivariate linear regression:

$$\min_{\mathbf{W}} ||\mathbf{R}^{(\text{tr})} - \mathbf{X}^{(\text{tr})}\mathbf{W}||_F^2 + \lambda ||\mathbf{W}||_F^2 \tag{4.7}$$

for which

$$\mathbf{W} = ((\mathbf{X}^{(\text{tr})})^T \mathbf{X}^{(\text{tr})} + \lambda \mathbf{I})^{-1} (\mathbf{X}^{(\text{tr})})^T \mathbf{R}^{(\text{tr})}. \tag{4.8}$$

The fundamental limitation of Equation 4.7 is that $\mathbf{X}$ is generally very high dimensional, often much more than the number of training instances i.e $d \gg m$. Hence, the model can easily overfit. Further, we cannot use the analytical solution as shown in Equation 4.8 due to the computational cost and memory requirement in inverting $((\mathbf{X}^{(\text{tr})})^T \mathbf{X}^{(\text{tr})} + \lambda \mathbf{I})$.

In the following section, we first compare Equations 4.1 and 4.6, and formulate a generalised neighbourhood method for cold-start recommendation. Second, we address the limitation, as discussed above, and formulate a large-scale linear model for cold start recommendation.

4.3.1 Generalised neighbourhood (Gen-Neighbourhood) based cold-start model

For the task of user cold-start recommendation, while Equation 4.1 uses user-user similarity, Equation 4.6 is essentially learning feature-item similarity. Based on this observation, we first formulate a *generalised neighbourhood model* for OC-CF setting:

$$\hat{\mathbf{R}} = \mathbf{R} \odot \mathbf{R}^T \star \mathbf{R}, \tag{4.9}$$

where $\odot$ and $\star$ refer to generalised matrix operations such as inner-product, and cosine product (normalised inner-product). By permitting various ordering of operations and definitions for the $\odot$ and $\star$ operators other than standard matrix multiplication, we define a generalised matrix algebra framework for recommendation that can recover many existing frameworks when the operators are defined appropriately. For instance, if we compute $\hat{\mathbf{R}}$ as :

$$\hat{\mathbf{R}} = (\mathbf{R} \odot \mathbf{R}^T) \star \mathbf{R},$$

where $\odot$ is cosine product and $\star$ is an inner-product, then it corresponds to cosine similarity based User-KNN model. The cosine product , say $\mathbf{C} = \mathbf{A} \odot_{cosine} \mathbf{B}$, can be defined as a normalised matrix multiplication

$$\begin{aligned} \mathbf{C}_{ij} &= \mathbf{A}_{i:} \odot_{cosine} \mathbf{B}_{:j} \\ &= \frac{1}{||\mathbf{A}_{i:}|| \, ||\mathbf{B}_{:j}||} \sum_k \mathbf{A}_{ik} \cdot \mathbf{B}_{kj} \end{aligned}$$

Similarly, if we compute $\hat{\mathbf{R}}$ as :

$$\hat{\mathbf{R}} = \mathbf{R} \odot (\mathbf{R}^T \star \mathbf{R}).$$

where $\star$ is an inner-product and $\odot$ is cosine product, then it corresponds to cosine similarity based Item-KNN model.

Now we define a generalised users feature based model as:

$$\hat{\mathbf{R}} = \mathbf{X}^{(\text{te})} \odot \mathbf{X}^{(\text{tr})} \star \mathbf{R}^{(\text{tr})}. \tag{4.10}$$

As per the definition of linear model in Equation 4.6, we define the cold-start recommendation as

$$\begin{aligned}\hat{\mathbf{R}} &= \mathbf{X}^{(\mathrm{te})} \odot \mathbf{W} \\ \hat{\mathbf{R}} &= \mathbf{X}^{(\mathrm{te})} \odot (\mathbf{X}^{(\mathrm{tr})} \star \mathbf{R}^{(\mathrm{tr})}).\end{aligned} \tag{4.11}$$

The proposed model can be applied to any users' side-information. Like the neighbourhood model, as discussed in Section 2.3.1, the key limitation of the proposed model is that it uses fixed operator for $\odot$ and $\star$ instead of learning directly by optimising some loss function. To address this limitation, we formulate a learning based model in the following section.

4.3.2 Low linear cold-start (LoCo) model

We now formulate a cold-start model overcoming the limitations of the linear model as discussed earlier. Our basic strategy is to learn a linear model where the weight matrix $\mathbf{W}$ is of *low rank*:

$$\min_{\mathbf{W}\colon \operatorname{rank}(\mathbf{W}) \leq k} ||\mathbf{R}^{(\mathrm{tr})} - \mathbf{X}^{(\mathrm{tr})}\mathbf{W}||_F^2 + \lambda ||\mathbf{W}||_F^2. \tag{4.12}$$

for some k latent dimension. We will optimise over a subset of low-rank matrices as follows. Let $\mathbf{X}_k^{(\mathrm{tr})} = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$ be the rank-k SVD approximation of $\mathbf{X}^{(\mathrm{tr})}$. Then, we will let $\mathbf{W} = \mathbf{V}_k \mathbf{Z}$ and solve

$$\min_{\mathbf{Z}} ||\mathbf{R}^{(\mathrm{tr})} - \mathbf{X}^{(\mathrm{tr})}\mathbf{V}_k\mathbf{Z}||_F^2 + \lambda ||\mathbf{V}_k\mathbf{Z}||_F^2. \tag{4.13}$$

Since $\mathbf{V}_k$ is orthonormal, the regulariser is equivalently $||\mathbf{Z}||_F^2$, and so we have the closed-form solution

$$\mathbf{Z} = (\mathbf{V}_k^T (\mathbf{X}^{(\mathrm{tr})})^T \mathbf{X}^{(\mathrm{tr})} \mathbf{V}_k + \lambda \mathbf{I})^{-1} \mathbf{V}_k^T (\mathbf{X}^{(\mathrm{tr})})^T \mathbf{R}^{(\mathrm{tr})}. \tag{4.14}$$

We call this model LoCo, for Low-rank Cold-start recommendation. An approximation of the projection $\mathbf{X}^{(\mathrm{tr})}\mathbf{V}_k$ can be computed efficiently using *randomised SVD* which we discuss in Appendix A. Further, the matrix inverse in Equation 4.14 is a $k \times k$ matrix and thus it can be computed efficiently for modest k. In fact, for the exact V_k,

$$\begin{aligned}\mathbf{X}\mathbf{V}_k &= \mathbf{U}_k \mathbf{\Sigma}_k, \\ (\mathbf{X}\mathbf{V}_k)^T \mathbf{X}\mathbf{V}_k &= \mathbf{\Sigma}_k^T \mathbf{U}_k^T \mathbf{U}_k \mathbf{\Sigma}_k = \mathbf{\Sigma}_k^2,\end{aligned}$$

and Equation 4.13 corresponds to whitened linear regression. In such case

$$(\mathbf{V}_k^T (\mathbf{X}^{(\mathrm{tr})})^T \mathbf{X}^{(\mathrm{tr})} \mathbf{V}_k + \lambda \mathbf{I})$$

is a diagonal matrix and its inverse can be computed trivially. However, in randomised SVD, V_k is an approximate, but we expect this to hold approximately.

4.4 Relation to existing models

In this section, we discuss how all of the discussed cold-start models can be viewed as a special case of 4.6. In Table 4.2, we summarise recommendation for various cold-start approaches discussed in this Chapter.

Method	Recommendation Matrix ($\hat{\mathbf{R}}^{(\text{te})}$)
NEIGHBOURHOOD	$(\mathbf{X}^{(\text{te})} \odot \mathbf{X}^{(\text{tr})}) \cdot \mathbf{R}^{(\text{tr})}$
CMF	$\mathbf{X}^{(\text{te})}\mathbf{Z}^T(\mathbf{B}\mathbf{B}^T + \mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{B}$
BPR-LINMAP	$\mathbf{X}^{(\text{te})}((\mathbf{X}^{(\text{tr})})^T\mathbf{X}^{(\text{tr})} + \lambda\mathbf{I})^{-1}(\mathbf{X}^{(\text{tr})})^T\mathbf{A}^{(\text{tr})}\mathbf{B}$
GEN-NEIGHBOURHOOD	$\mathbf{X}^{(\text{te})} \odot (\mathbf{X}^{(\text{tr})} \star \mathbf{R}^{(\text{tr})})$
LOCO	$\mathbf{X}^{(\text{te})}\mathbf{V}_k(\mathbf{V}_k^T(\mathbf{X}^{(\text{tr})})^T\mathbf{X}^{(\text{tr})}\mathbf{V}_k + \lambda\mathbf{I})^{-1}\mathbf{V}_k^T(\mathbf{X}^{(\text{tr})})^T\mathbf{R}^{(\text{tr})}$

Table 4.2: Summary of cold-start recommendation for various approaches.

4.4.1 Neighbourhood model

Consider the neighbourhood model of in Equation 4.11, When $\odot, \star$ are standard inner product operations,

$$\hat{\mathbf{R}}^{(\text{te})} = \mathbf{X}^{(\text{te})}((\mathbf{X}^{(\text{tr})})^T\mathbf{R}^{(\text{tr})}), \tag{4.15}$$

corresponding exactly to the linear model of Equation 4.6 with a weight matrix $\mathbf{W} = \mathbf{X}^T\mathbf{R}$. Recalling that $\mathbf{R}^{(\text{te})} \equiv \mathbf{0}$, the predicted rating for user u and item i is thus

$$\begin{aligned}\hat{\mathbf{R}}_{ui} &= \mathbf{X}_{u:}\mathbf{X}^T\mathbf{R}_{:i} \\ &= \frac{1}{2}\cdot\mathbf{X}_{u:}\mathbf{X}^T(2\cdot\mathbf{R}_{:i} - \mathbf{1} + \mathbf{1}) \\ &= \frac{1}{2}\cdot\left(\sum_{\mathbf{R}_{u'i}=1}\mathbf{X}_{u:}\mathbf{X}_{u':}^T - \sum_{\mathbf{R}_{u'i}=0}\mathbf{X}_{u:}\mathbf{X}_{u':}^T\right) + \frac{1}{2}\mathbf{X}_{u:}\mathbf{X}^T\mathbf{1}.\end{aligned}$$

The second term above is independent of i, and thus plays the role of a per-user bias that does not affect ranking. The first term above corresponds to a *nearest unnormalised centroid* classifier: we measure whether the social metadata $\mathbf{X}_{u:}$ for the given user is more similar to that of the unnormalised metadata centroid of the users that like item i, or those that dislike item i. The normalised centroid classifier is also known as the Rocchio classifier in information retrieval [Manning et al., 2008], and is attained if we normalise the columns of $\mathbf{R}$ above to sum to 1, i.e. use an asymmetric cosine similarity for $\star$.

4.4.2 CMF model

The parameters of the CMF model (Equation 4.2) can be learned by iteratively optimising with respect to each individual parameter, keeping all others fixed. Each such individual optimisation is a regression problem, and thus admits a closed form solution. The unregularised optimal solution for $\mathbf{A}$ is

$$\mathbf{A} = (\mathbf{R}\mathbf{B}^T + \mathbf{X}\mathbf{Z}^T)(\mathbf{B}\mathbf{B}^T + \mathbf{Z}\mathbf{Z}^T)^{-1} \tag{4.16}$$

for the optimal choices of $\mathbf{Z}, \mathbf{B}$, which will depend in some non-trivial way on $\mathbf{R}, \mathbf{X}$. Thus, the cold-start prediction is $\hat{\mathbf{R}}^{(\text{te})} = \mathbf{A}^{(\text{te})}\mathbf{V} = \mathbf{X}^{(\text{te})}\mathbf{Z}^T(\mathbf{B}\mathbf{B}^T + \mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{B}$. This is a special case of Equation 4.11 for low-rank weight matrix $\mathbf{W} = \mathbf{Z}^T(\mathbf{B}\mathbf{B}^T + \mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{B}$.

4.4.3 BPR-LinMap model

For the BPR-LinMap model (Equations 4.4), the optimal linear regression weights $\mathbf{T}$ to map the metadata to latent features are

$$\mathbf{T} = ((\mathbf{X}^{(\text{tr})})^T\mathbf{X}^{(\text{tr})} + \lambda\mathbf{I})^{-1}(\mathbf{X}^{(\text{tr})})^T\mathbf{A}^{(\text{tr})}, \tag{4.17}$$

meaning that

$$\hat{\mathbf{R}}^{(\mathrm{te})} = \mathbf{X}^{(\mathrm{te})}((\mathbf{X}^{(\mathrm{tr})})^T\mathbf{X}^{(\mathrm{tr})} + \lambda\mathbf{I})^{-1}(\mathbf{X}^{(\mathrm{tr})})^T\mathbf{A}^{(\mathrm{tr})}\mathbf{B},$$

which is an instance of Equation 4.6 with low-rank weight matrix $\mathbf{W} = ((\mathbf{X}^{(\mathrm{tr})})^T\mathbf{X}^{(\mathrm{tr})} + \lambda\mathbf{I})^{-1}(\mathbf{X}^{(\mathrm{tr})})^T\hat{\mathbf{R}}^{(\mathrm{tr})}$, recalling that $\hat{\mathbf{R}}^{(\mathrm{tr})} = \mathbf{A}^{(\mathrm{tr})}\mathbf{B}$. Indeed, the model is identical to that of multivariate linear regression (Equation 4.8), with one crucial difference: one uses $\hat{\mathbf{R}}^{(\mathrm{tr})}$ in place of $\mathbf{R}^{(\mathrm{tr})}$ i.e. we replace the regression target matrix by a low-rank approximation.

4.5 Experiment and evaluation

In this section, we discuss the dataset used for experiments, experimentation methodology, and report detailed experimental results.

4.5.1 Data description

In this Chapter, we evaluate the performance of the models on two real-world datasets.

- ***Kobo Dataset*** comes from Kobo Inc., a major online ebook retailer with more than 20 million readers. It contains an anonymised dataset of ebook purchases and Facebook friends and page likes for a random subset of 32,027 Kobo users; these users purchased more than 88,810 books, had over 9 million friends and liked about 6 million pages. We subsample pages by including only those pages liked by at least 5 people and not more than 5,000 people, which reduces the number of unique pages to about 606,780. The dataset also includes item purchase timestamps, basic user demographics, namely age group, gender, location and friendship network.

  We split the dataset into 10 temporally divided train and test folds. To prepare the training dataset, we include all the user-item data prior to a specific date, starting from June 2012 and incremented by a month in each fold. The test set includes the first purchase of all new users in the following month when the user had at least 10 page likes.

- ***Flickr Dataset*** comes from [Tang and Liu, 2009]. It consists of 80,513 users, 195 groups joined by the users, and a social network with 5,899,882 friendship relationships as summarised in Table 4.3. The goal is to recommend relevant groups to the users leveraging their social network data i.e friendship network.

  We create 10 train-test folds by including random 10% of the users in the test set and remaining 90% users in the training set.

Kobo Data	
# Users	32,027
# Items	88,810
# Pages	6,218,698

Flickr Data	
# Users	80,513
# Items (groups)	195
# Friendship relationships	5,899,882

Table 4.3: Dataset description for Kobo and Flickr datasets.

4.5.2 Model comparison

In this section, we discuss various baselines we used to compare with the proposed methods:

- MOST POPULAR baseline recommends the most popular items in the dataset in order of popularity.
- CBF-KNN-LOW is a neighbourhood recommender , as discussed in Section 4.2.1, where user-user similarities are computed from the low dimensional projection of user-attributes.
- BPR-KNN-LOW of [Gantner et al., 2010], as discussed in Section 4.2.2, but using *k-nearest-neighbor* [2] rather than linear regression to map user attributes to the latent factors.
- CMF of [Krohn-Grimberghe et al., 2012] as defined in Equation 4.2.

For *generalised cold-start neighbourhood model*, we explore the following operators for $\odot$ and $\star$ as per Equation 4.11, for any row vector $\boldsymbol{r}$ and column vector $\boldsymbol{c}$

- IP : Standard matrix multiply inner product given by $\mathbf{S}_{r,c} = \langle \boldsymbol{r}, \boldsymbol{c} \rangle$.
- BINIP : For $\tau \in \mathbb{R}$, a binary thresholded version of the inner product given by

$$\mathbf{S}_{r,c} = \begin{cases} 1 & if \ \langle \boldsymbol{r}, \boldsymbol{c} \rangle > \tau \\ 0 & \text{otherwise.} \end{cases}$$

- LOGIP : A logarithm of the inner product $\mathbf{S}_{r,c} = \log\langle \boldsymbol{r}, \boldsymbol{c} \rangle$.
- COS : Cosine similarity which is obtained by an inner product of two L_2 normalized vectors equivalent to $\mathbf{S}_{r,c} = \frac{\langle \boldsymbol{r}, \boldsymbol{c} \rangle}{\|\boldsymbol{r}\| \|\boldsymbol{c}\|}$.

We can obtain different social cold-start recommenders by choosing the similarity metrics (IP, LOGIP, BINIP, COS) used for both $\odot$ and $\star$ in Equation 4.11. We express the choices in the order of $\star$-$\odot$ and show evaluations for the following eight possibilities (the remaining possibilities did not approach the best result reported among these eight):

- $\odot$ = IP, $\star \in \{$IP, LOGIP, BINIP, COS$\}$
- $\odot$ = COS, $\star \in \{$IP, LOGIP, BINIP, COS$\}$

In our experiments, we used the threshold $\tau = 2$ for BINIP.

We used cosine similarity for the methods relying on similarity except for generalised neighborhood method where we used the above operators. For the latent factor methods, we tuned the latent dimension from $\{5, 10, 50, 100, 500, 1000\}$. For the methods relying on ℓ_2 regularisation, we tuned all regularisation strengths from $\{1000, 100, 10, 1, 0.1, 0.01, 0.001, 0.0001\}$.

In all of the experiments, we report *Precision@k*, *Recall@k* and *mean average precision* ($mAP@100$), as discussed in Chapter 2, using 10-fold cross validation and provide standard error bars corresponding to 95% confidence intervals.

4.5.3 Results and analysis

In this section, we report and discuss the experimental results for various cold-start algorithms

[2] We experimented on various mapping strategies, as reported in [Gantner et al., 2010], on full and low rank projection of user attributes. We found kNN with the low-dimensional projection of user attributes to work best.

Evaluation of generalised neighbourhood cold-start model

In Table 4.4 and 4.5, we evaluate the performance of generalised neighbourhood based cold-start recommender for various choice of $\star$ and $\odot$ on *Kobo* dataset. Here, we see that Cos-Cos model yields superior results. From here onwards, we will choose Cos-Cos models for rest of the experiments.

	IP-IP		LogIP-IP		BinIP-IP		Cos-IP	
k	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
@1	0.027±0.013	0.027±0.013	**0.030±0.012**	**0.030±0.012**	0.028±0.012	0.028±0.012	**0.030±0.012**	**0.030±0.012**
@3	0.018±0.007	0.053±0.020	0.020±0.008	0.061±0.023	**0.022±0.009**	**0.066±0.026**	**0.022±0.008**	**0.066±0.024**
@5	0.017±0.006	0.086±0.032	0.019±0.008	0.097±0.038	**0.020±0.008**	**0.102±0.040**	**0.020±0.008**	**0.102±0.038**
@10	0.014±0.005	0.136±0.047	**0.015±0.005**	0.148±0.050	**0.015±0.005**	0.153±0.051	**0.015±0.005**	**0.154±0.052**
@20	0.009±0.003	0.181±0.060	0.009±0.003	0.185±0.061	0.009±0.003	0.186±0.060	**0.010±0.003**	**0.191±0.060**
mAP	0.057±0.019		0.062±0.021		0.063±0.022		**0.065±0.021**	

Table 4.4: Performance of Page Likes Recommender for $\star \in \{\text{IP, LogIP, BinIP, Cos}\}$, $\odot = IP$ on *Kobo* dataset.

	IP-Cos		LogIP-Cos		BinIP-Cos		Cos-Cos	
k	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
@1	0.031±0.014	0.031±0.014	0.031±0.013	0.031±0.013	0.030±0.014	0.030±0.014	**0.038±0.015**	**0.038±0.015**
@3	0.021±0.008	0.064±0.023	0.022±0.008	0.065±0.023	0.021±0.008	0.063±0.024	**0.025±0.010**	**0.076±0.029**
@5	0.020±0.008	0.100±0.039	0.020±0.008	0.101±0.038	0.020±0.007	0.100±0.037	**0.023±0.009**	**0.117±0.044**
@10	0.015±0.005	0.151±0.052	0.015±0.005	0.153±0.052	0.016±0.005	0.159±0.055	**0.017±0.006**	**0.173±0.059**
@20	0.010±0.003	0.193±0.061	0.010±0.003	0.193±0.061	0.010±0.003	0.202±0.063	0.010±0.003	**0.205±0.063**
mAP	0.065±0.022		0.065±0.022		0.065±0.021		**0.075±0.025**	

Table 4.5: Performance of Page Likes Recommender for $\star \in \{\text{IP, LogIP, BinIP, Cos}\}$, $\odot = Cos$ on *Kobo* dataset.

Evaluation of predictive power of various side-information

In Table 4.6, we compare the predictiveness of various side-information, namely (a) Demographics, (b) Friend network, and (c) Page likes, using Cos-Cos model on *Kobo* dataset. Here, we observe page likes based recommender significantly outperforms other models. This illustrates the predictive power of page likes, as we observed in Chapter 3. For the rest of the experiments, we will use page-like features to compare various cold-start recommendation algorithms.

	Demographics		Friend Network		Page Likes	
k	Precision	Recall	Precision	Recall	Precision	Recall
@1	0.014±0.004	0.014±0.004	0.014±0.006	0.014±0.006	**0.038±0.015**	**0.038±0.015**
@3	0.010±0.003	0.031±0.008	0.011±0.004	0.032±0.012	**0.025±0.010**	**0.076±0.029**
@5	0.010±0.004	0.050±0.018	0.009±0.003	0.044±0.016	**0.023±0.009**	**0.117±0.044**
@10	0.008±0.003	0.083±0.029	0.006±0.002	0.062±0.022	**0.017±0.006**	**0.173±0.059**
@20	0.006±0.003	0.125±0.051	0.004±0.001	0.078±0.025	**0.010±0.003**	**0.205±0.063**
mAP	0.035±0.010		0.029±0.010		**0.075±0.025**	

Table 4.6: Performance comparison of Cos-Cos model for various users' side-information on *Kobo* dataset.

Evaluation of various cold-start recommenders

In Table 4.7 and Table 4.8, we compare the performance of LoCo with various cold-start recommenders on the *Kobo* and *Flickr* datasets. LoCo yields a $\sim$25% improvement on the *Kobo* dataset and 10% improvement on the *Flickr* dataset over the most competitive baseline, viz CMF on *Kobo*, Cos-Cos

	MOST POPULAR		BPR-KNN-LOW		CMF		CBF-KNN-LOW		COS-COS		LoCo	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
@1	0.006±0.003	0.006±0.003	0.008±0.005	0.008±0.005	0.049±0.028	0.049±0.028	0.038±0.022	0.038±0.022	0.038±0.015	0.038±0.015	**0.064±0.033**	**0.064±0.033**
@3	0.008±0.003	0.023±0.010	0.011±0.007	0.033±0.021	0.033±0.019	0.101±0.056	0.023±0.010	0.069±0.031	0.025±0.010	0.076±0.029	**0.041±0.016**	**0.123±0.047**
@5	0.008±0.004	0.039±0.019	0.014±0.009	0.071±0.045	0.027±0.013	0.140±0.067	0.022±0.008	0.108±0.040	0.023±0.009	0.117±0.044	**0.036±0.012**	**0.178±0.059**
@10	0.007±0.003	0.074±0.034	0.012±0.006	0.123±0.062	0.019±0.008	0.187±0.080	0.019±0.006	0.191±0.059	0.017±0.006	0.173±0.059	**0.025±0.007**	**0.252±0.070**
@20	0.006±0.003	0.120±0.067	0.010±0.004	0.203±0.089	0.011±0.004	0.221±0.079	0.013±0.004	0.270±0.072	0.010±0.003	0.205±0.063	**0.015±0.004**	**0.300±0.077**
mAP	0.026±0.011		0.042±0.017		0.093±0.044		0.079±0.027		0.075±0.025		**0.117±0.042**	

Table 4.7: Comparison of cold-start recommenders on *Kobo* dataset.

	MOST POPULAR		BPR-KNN-LOW		CMF		CBF-KNN-LOW		COS-COS		LoCo	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
@1	0.170±0.003	0.135±0.003	0.126±0.002	0.095±0.002	0.159±0.003	0.123±0.003	0.206±0.002	0.158±0.002	0.223±0.005	0.176±0.005	**0.260±0.003**	**0.202±0.003**
@3	0.097±0.001	0.221±0.003	0.074±0.002	0.163±0.003	0.099±0.001	0.217±0.003	0.132±0.002	0.292±0.004	0.136±0.002	0.304±0.005	**0.154±0.002**	**0.341±0.004**
@5	0.073±0.001	0.282±0.005	0.056±0.001	0.205±0.006	0.076±0.001	0.272±0.004	0.101±0.001	0.367±0.004	0.103±0.001	0.376±0.005	**0.113±0.001**	**0.409±0.006**
@10	0.050±0.001	0.377±0.005	0.038±0.001	0.279±0.005	0.051±0.001	0.366±0.005	0.066±0.001	0.471±0.005	0.067±0.001	0.478±0.004	**0.070±0.001**	**0.497±0.005**
@20	0.033±0.000	0.502±0.004	0.026±0.000	0.382±0.004	0.033±0.000	0.467±0.005	0.040±0.000	0.572±0.004	0.041±0.000	0.582±0.004	**0.042±0.000**	**0.582±0.004**
mAP	0.224±0.003		0.168±0.002		0.215±0.002		0.277±0.002		0.294±0.004		**0.321 ± 0.003**	

Table 4.8: Comparison of cold-start recommenders on *Flickr* dataset.

on *Flickr* dataset. Among the runners-up (existing methods), CMF was superior on the *Kobo* dataset, but COS-COS performed better on the *Flickr* dataset. Furthermore, BPR-KNN-LOW performed poorly on both datasets, indicating the lack of robustness of two step algorithm. The high variances for all methods on the *Kobo* dataset indicate that with temporal splits, there is a potentially strong concept drift in the data that affects all methods since they do not leverage temporal information.

Further, tuning the large number of hyperparameters for CMF and BPR-KNN-Low took ∼2 hours and 20 hours respectively on the *Kobo* dataset. On the other hand, tuning for LoCo and other methods took the order of minutes. Hence, LoCo is computationally efficient and suitable for large-scale recommendation.

Evaluation of cold-start vs near cold-start recommender

In Table 4.9, we compare the performance of cold-start recommender with near cold start recommender on *Kobo* dataset. We prepare the test set by including users who have purchased only one item in the training set and used Item-Knn, as discussed in Section 2.4.2, for recommendation. Table 4.9 shows that item purchase is a strong signal. However, cold-start recommender based on page likes yields superior results for higher k for *precision@k* and *recall@k* metrics.

	ITEM-KNN		LoCo	
k	Precision	Recall	Precision	Recall
@1	**0.115±0.028**	**0.115±0.028**	0.064±0.033	0.064±0.033
@3	**0.051±0.012**	**0.153±0.035**	0.041±0.016	0.123±0.047
@5	**0.034±0.007**	**0.171±0.034**	**0.036±0.012**	**0.178±0.059**
@10	0.019±0.003	0.186±0.033	**0.025±0.007**	**0.252±0.070**
@20	0.009±0.002	0.190±0.034	**0.015±0.004**	**0.300±0.077**

Table 4.9: Comparision of cold-start recommender (*LoCo*) with near cold-start Item-Knn recommender on *Kobo* dataset.

Evaluation of runtime of various cold-start recommenders

In table 4.10, we compare the hyperparameter validation time needed for all methods on *Ebook*. LoCo is seen to be orders of magnitude faster to tune than the learning-based methods CMF and BPR-KNN-Low.

Method	BPR-KNN-Low	CMF	CBF-KNN-Low	Cos-Cos	Loco
Validation time	20 hour 30 mins	2 hour 2 mins	5 mins 10 secs	2 secs	5 mins 3 secs

Table 4.10: Comparison of validation times on *Kobo* dataset.

While the CBF and Cos-Cos methods are faster to tune than LoCo, the latter is more accurate. Hence, LoCo attains a suitable balance between accuracy and runtime.

Analysis

We conclude our experiments by evaluating the performance of the proposed methods as a function of different key quantities:

- **How does performance vary with the number of page likes for a target user?**

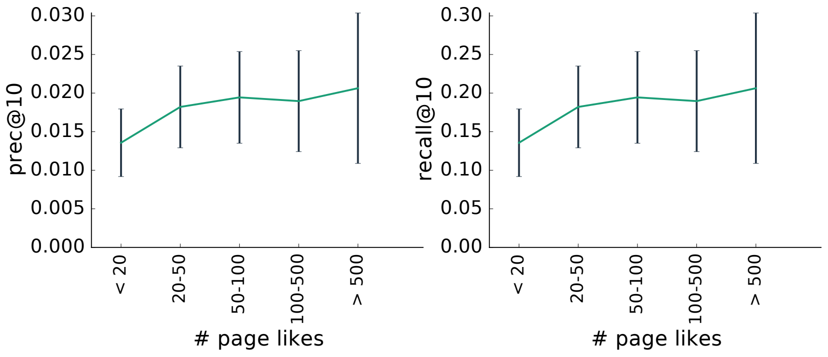


Figure 4.1: Variation of the performance of the page like based Cos-Cos model with the numbers of page likes.

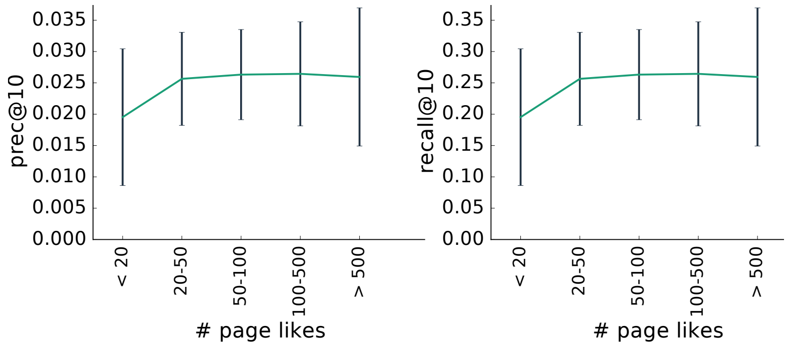


Figure 4.2: Variation of the performance of the page like based LoCo model with the numbers of page likes on Kobo dataset.

The number of page likes varies from user to user. Thus we divide users into six categories based on the number of pages they have liked and evaluate the performance (mAP@100) on each user category. In Figure 4.1 and 4.2, we evaluate the performance of Cos-Cos and LoCoon *Kobo* dataset respectively. It shows that performance increases as the number of user page likes increases, but with diminishing return. The high variance for users with few page likes indicate a lack of sufficient information, whereas the high variance for users with many page likes indicate a lack of selectiveness with page likes. Most importantly, we see that LoCo makes good item recommendations to the users with a moderate number of page likes.

- **How does performance of LoCo varies with the dimension of the projection?**

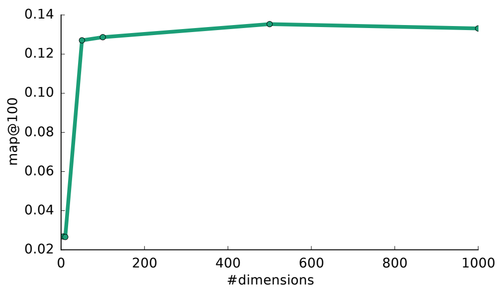


Figure 4.3: LoCo retrieval performance versus dimension of projection on the Kobo dataset.

In Figure 4.3, we evaluate the performance (mAP@100) with the projection dimension on the *Kobo* dataset. We observe that performance increases sharply with the number of dimensions, but with diminishing returns beyond 100 dimensions. Hence, LoCo is applicable to large scale problems with high dimensional user features.

4.6 Conclusion

In this Chapter, we formulated novel linear cold-start algorithms. We showed how several popular social cold-start models can be seen as special cases of a linear content-based model, with different constraints on the learned weights. Based on this insight, we proposed two large-scale cold-start models: (a) Generalised neighbourhood based model,and (b) LoCo. Further, we demonstrated the effectiveness of linear models for the cold-start recommendation and substantial predictive power Facebook page likes in predicting users' preferences.

Despite its superior performance, it is unclear how the proposed framework can be extended to operate seamlessly in non-cold-start settings as well. In practical settings, we expect a mix of both cold and warm start users, and it's undesirable to build separate models for each set of users. In next Chapter, we address this by formulating a CF algorithm for OC-CF using the same linear framework.

Chapter 5

Linear Models for One-Class Collaborative Filtering

In the last two Chapters, we demonstrated the efficacy of linear models, and predictiveness of social signals for specific CF scenarios. In this Chapter, we investigate linear models for the one-class collaborative filtering (OC-CF) setting. OC-CF is concerned with predicting users' preferences from the data that consists of positive only preferences. OC-CF scenario is one of the most important settings for recommendation mostly due to its prevalence in many practical recommendation settings such as item purchase in e-commerce, as discussed atatcin Chapter 2. In this Chapter, we formulate recommendation as a classification problem and propose a user-focused linear model for OC-CF. Further, leveraging the classification view, we formulate a dimensionality reduction based model to scale linear models on large scale datasets

5.1 Problem setting

In this Chapter, we are concerned with recommending a ranked list of items to the users in OC-CF setting. Formally,w e have historical preference data $\mathbf{R} \in \{0, 1\}^{m \times n}$ that consists of users' purchase history, where $\mathbf{R}_{ui} = 1$ means that the user u purchased item i, and $\mathbf{R}_{ui} = 0$ refers to not purchased; however, it doesn't imply that user would not purchase the item in the future. Hence, $\mathbf{R}_{ui} = 0$ indicates missing as opposed to negative feedback. Based on existing purchase information, a recommendation algorithm will produce a matrix $\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$ of estimated preference scores. As the entries in $\hat{\mathbf{R}}$ are real-valued, for each user u, we can sort the entries of $\hat{\mathbf{R}}_{u:}$ to obtain a predicted ranking over items. As discussed in Chapter 2, none of the existing methods meet all CF desiderata, namely (1) applicable to wide range of recommendation scenarios, (2) learning-based, (3) amenable to convex optimisation, and (4) scalable. In this Chapter, we propose a simple linear model that addresses all CF desiderata.

5.2 LRec: Linear model for OC-CF

First, we focus on learning a linear model for each user in OC-CF setting defined above. Given the user-item interaction matrix $\mathbf{R}$, for each $u \in U$, we define a vector $\mathbf{y}^{(u)} \in \{\pm 1\}^n$ that is the same as $\mathbf{R}_{u:}$ except with zeros in $\mathbf{R}_{u:}$ replaced by -1:

$$\mathbf{y}^{(u)} = 2\mathbf{R}_{u:} - 1.$$

Now, we formulate LRec as

$$\min_{\mathbf{w}} \sum_{u \in U} \sum_{i \in I} \ell(\mathbf{y}_i^{(u)}, \mathbf{R}_{:i}^T \mathbf{w}^{(u)}) + \Omega(\mathbf{w}), \tag{5.1}$$

where ℓ is some convex loss function, and $\mathbf{W} = \{\mathbf{w}^{(u)}\}_{u \in U}$. Each $\mathbf{w}^{(u)} \in \mathbb{R}^m$, and so we can equivalently think of $\mathbf{W} = \{\mathbf{w}\}$ for some $\mathbf{W} \in \mathbb{R}^{m \times m}$, with $\mathbf{w}^{(u)} = \mathbf{W}_{:u}$.

In the sequel, we will employ $\Omega(\mathbf{W}) = \frac{\lambda}{2}||\mathbf{W}||_F^2$. Although any loss function ℓ can be applied, we focus on two loss functions

1. Logistic loss, $\ell(y, v) = \log(1 + e^{-yv})$, mainly due to the existence of efficient solvers for linear logistic regression, such as such as LibLinear[1] [Fan et al., 2008], and its suitability for estimating class-probabilities, which we will see in Section 5.2.2 is especially appropriate for learning in OC-CF setting,

2. Squared loss, $\ell(y, v) = (y - v)^2$, mainly due to closed-form solution, which we will exploit for scaling the model to large scale dataset.

We now discuss the intuition for the LRec model, before contrasting it to existing methods for OC-CF.

5.2.1 A linear classification perspective

Equation 5.1 can be interpreted as a linear classification model for each user u. The feature matrix for each such model is the *entire* purchase matrix transposed, while the label vector for each model comprises the purchase information for that user. That is, for each model, we have a separate training example corresponding to each item. The feature vector for the item comprises the purchase history for *all* users (including the one in consideration[2]), while the label for the item is simply whether or not the user in consideration purchased it. Each $\mathbf{W}_{uu'}$ is a weight indicating the relevance of the purchases of user u in predicting the purchases of the user u'.

Given a learned $\mathbf{W}$, LRec produces a recommendation matrix

$$\hat{\mathbf{R}}(\mathbf{W}) = \mathbf{W}^T \mathbf{R}, \tag{5.2}$$

or, for a specific (user, item) pair (u, i),

$$\hat{\mathbf{R}}_{ui} = \sum_{u' \in U} \mathbf{W}_{uu'} \cdot \mathbf{R}_{u'i}.$$

We see that the score assigned to the (u, i) pair is simply the *sum of the similarities of all other users who purchased item i*. (The similarity to neighbourhood methods will be discussed more in §5.4.2.)

We can also interpret the scores by considering the matrix $\alpha \in \mathbb{R}^{n \times m}$, whose columns correspond to the *dual* weights of the objective in Equation 5.1. The weights α_{iu} can be interpreted as the relevance of item i to user u. From the relationship between the primal and dual weights [Zhang, 2002], we can write

$$\hat{\mathbf{R}}_{ui} = \sum_{i' \in I} \alpha_{i'u} \cdot (\mathbf{R}_{:i}^T \mathbf{R}_{:i'}). \tag{5.3}$$

The dot product $\mathbf{R}_{:i}^T \mathbf{R}_{:i'}$ is simply the *number of users who co-purchased items i and i'*. Therefore, the predicted score for (u, i) is the *sum of the number of co-purchases with every other item, weighted by the*

[1] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[2] This is only done for the *training* purchases of each user. Therefore, there is no leakage of test purchases into the model.

relevance of these items to user u. Further, the dual formulation can be used for efficient optimisation when $m \gg n$.

5.2.2 A positive and unlabelled perspective

In the linear classification model for each user, LRec treats the known preferences as positive examples, and unknown preferences as negative examples. It appears that this falls into the trap of suppressing preferences for items that the user is unaware of. However, one can justify this strategy as follows.

In a binary classification context, we can view the learning problem for each user as one of learning from positive and unlabelled data [Denis, 1998]. Elkan and Noto [2008] proved a surprising fact about learning from such data, namely, that for the purposes of *ranking*, it suffices to simply treat the unlabelled examples as negative, and estimate class-probabilities. As we are only interested in ranking performance for top-N items for each user, this justifies learning user focused model by treating unknown preferences as negative, provided we choose ℓ in Equation 5.1 to be some loss suitable for estimation of class-probabilities [Buja et al., 2005], such as the logistic ($\ell(y, v) = \log(1 + e^{-yv})$) or squared ($\ell(y, v) = (y - v)^2$) loss.

We now detail some salient properties of the LRec model.

5.2.3 Properties of LRec

Several observations are worth emphasising at this point. First, training of the LRec model is highly parallelisable across the users, as the weights $\mathbf{w}^{(u)}$ do not depend on each other. Parallelisation may thus be conducted *without distributed communication*. Further, the design matrix $\mathbf{R}^T$ is identical across all users u. Therefore, this matrix can be *shared* across all such parallel executions of a per-user model, which is useful on a multi-core architecture.

Second, the feature matrix for each model is highly sparse: we expect that for most items, only a small fraction of users will have purchased them. This means that the actual optimisation of the individual linear classification models can be done efficiently, employing e.g. sparse matrix-vector computations.

Third, the use of ℓ_2 regularisation is crucial beyond its standard value in preventing overfitting. Recall that for each user u's model, that user's purchase history is included as a feature. Without regularisation, then, the solution to Equation 5.1 is trivial: we simply let $\mathbf{W}_{uu'} = 0$ for all $u \neq u'$, and let $\mathbf{W}_{uu} \to +\infty$. However, with ℓ_2 regularisation, such a solution is penalised. What is favoured instead is a spreading of weight across other *similar* users.

Fourth, the objective in Equation 5.1 is strictly convex. Therefore, there are no issues of local optima that methods such as matrix factorisation face.

In Table 5.1, we summarise properties of various collaborative filtering methods.

5.3 Extensions of LRec

In this Section, we discuss several possible extensions to the LRec model.

5.3.1 Incorporating side-information

It is easy to incorporate any side-information in the form of feature vectors for users and items. Suppose each item has a feature vector $\mathbf{v} \in \mathbb{R}^d$, and let $\mathbf{X}^I \in \mathbb{R}^{n \times d}$ denote the matrix of features for all

Table 5.1: Comparison of recommendation methods for OC-CF. The * for PureSVD is added because the objective is not convex, but has a closed-form solution via the singular value decomposition. The * for BPR is added because the extension in Gantner et al. [2012] is needed to ensure user focus.

Method	**Reference**	**Learning?**	**Convex?**	**User-focussed?**	**Embarrasingly Parallelisable**
U-KNN	[Herlocker et al., 1999]	×	NA	✓	✓
I-KNN	[Sarwar et al., 2001]	×	NA	×	✓
PURESVD	[Cremonesi et al., 2010]	✓	✓*	✓	×
WRMF	[Pan et al., 2008]	✓	×	✓	×
BPR	[Rendle et al., 2009]	✓	×	✓*	×
SLIM	[Ning and Karypis, 2011]	✓	✓	×	✓
LREC	Our method	✓	✓	✓	✓

items. Then, we can use exactly the same core LRec model as Equation 5.1, but change the feature representation to

$$\begin{bmatrix}\mathbf{R}^T & \mathbf{X}^I\end{bmatrix} \in \mathbb{R}^{n\times(m+d)}.$$

The learned weights will now be $\mathbf{W} \in \mathbb{R}^{m\times(m+d)}$, so that for each user we additionally learn affinities to the features of an item.

5.3.2 Weighting for class-imbalance

In addition to the feature matrix for each model being sparse, we also expect the label vector for each model to be highly imbalanced, as each user will typically have purchased only a small fraction of all available items. In a binary classification context, we can think of this as an *imbalanced learning* problem. In such cases, the estimates of parameters produced by e.g. logistic regression are known to be biased [King and Zeng, 2001]. To obtain better estimates of the parameters, a simple option is to balance the loss by the inverse base rate of each class [Lin et al., 2002, Menon et al., 2013]. Thus, the objective becomes

$$\min_{\mathbf{W}} \sum_{u\in U}\sum_{i\in I} \frac{1}{\pi_i^{(u)}} \cdot \ell(\mathbf{y}_i^{(u)}, \mathbf{X}_{i:}^T \mathbf{w}^{(u)}) + \Omega(\mathbf{W}),$$

where

$$\pi_i^{(u)} = [\![\mathbf{y}_i^{(u)} = 1]\!] \cdot |I_u| + [\![\mathbf{y}_i^{(u)} = -1]\!] \cdot (n - |I_u|).$$

Weighted logistic regression objectives can be solved with a variant of LIBLINEAR that accepts weights for training instances[3].

5.3.3 Subsampling negatives

In scenarios with a large number of items, it may be infeasible to train a linear classification model with n training examples. As noted above, each model typically involves a highly imbalanced training set. For computational convenience, we can thus apply another technique standard in learning from imbalanced

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/weights/liblinear-weights-1.94.zip

data, namely, subsampling of the dominant class [Kubat and Matwin, 1997]. Here, for each model corresponding to a user u, we keep all items which the user purchased as training examples. Amongst all items the user did not purchase, we only retain a random subsample. The precise downsampling ratio can be chosen empirically; once the ratio is fixed, we can again apply the weighting scheme above to ensure reliable estimates of parameters.

5.4 Relation to existing models

We now contrast LRec to existing methods for recommendation with implicit feedback.

5.4.1 Relation to SLIM

The model most closely related to LRec is SLIM (Equation 2.11). To see the connection, observe that we may rewrite SLIM as (see also [Ning and Karypis, 2011, Equation 4])

$$\min_{\mathbf{W}\in\mathcal{C}} \sum_{i\in I} \sum_{u\in U} \ell(\mathbf{y}_u^{(i)}, \mathbf{R}_{u:}^{(i)}\mathbf{w}^{(i)}) + \Omega(\mathbf{W}),$$

where $\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{n\times n} \colon \text{diag}(\mathbf{W}) = 0, \mathbf{W} \geq 0\}$, $\ell(y, v) = (y - v)^2$ is the square loss, and the feature matrix $\mathbf{R}$ and label vector $\mathbf{y}^{(i)} = \mathbf{R}_{:i}$.

LRec has three main advantages over SLIM.

User-focussed modelling. We can see SLIM as learning a separate model for each *item*, where in each such model we treat each user as a training instance, represented by their purchase history. This is in contrast to LRec, which learns a separate model for each *user*. While this difference appears trivial, it is very important in the context of recommendation. In most recommendation tasks, one is interested in achieving good recommendation performance for an *average user*, i.e. the natural correspondence to a "query" in information retrieval is the user. It is thus of interest to ensure that for each such user, we optimise the recommendation performance. By contrast, we can see SLIM as solving a subtly different problem, namely, ensuring that for each *item*, there is a good ranking over the users that would be interested in it. While this will of course be correlated with the solution from the former, in practice we shall see that there is a performance gap between the two approaches.

Inclusion of target user. Another subtle difference between LRec and SLIM is the definition of the constraint set $\mathcal{C}$. In SLIM, the use of $\text{diag}(\mathbf{W}) = 0$ is crucial, because otherwise as noted we can just set $\mathbf{W} = \mathbf{I}$ and attain zero loss. However, such a solution will *not* be optimal when using logistic loss, as in LRec. Therefore, this means that the constraint $\text{diag}(\mathbf{W}) = 0$ is unnecessary. Dropping the constraint is beneficial from the perspective of parallelisation, as we can share the feature matrix $\mathbf{R}^T$ across multiple models.

Optimisation. As noted earlier, with the choice of logistic loss or squared loss, the LRec objective (Equation 5.1) can be optimised via efficient standard solvers. By contrast, optimisation of SLIM requires respecting the nonnegativity constraint on the weights, as well as performing ℓ_1 regularisation. This can be handled by e.g. proximal gradient methods, but the resulting optimisation is not as efficient as the unconstrained version [Levy and Jack, 2013].

5.4.2 Relation to neighbourhood methods

As item-based neighbourhood approaches can be seen as subsumed by SLIM, the above advantages of LRec over SLIM apply to such methods as well. Another perspective can be gained from the dual formulation of the LRec prediction in Equation 5.3. The latter equation appears similar to the item-based neighbourhood prediction, which from Equation 2.10 can be written

$$\hat{\mathbf{R}}_{ui} = \sum_{i' \in I} \mathbf{R}_{ui'} \cdot \mathbf{S}_{i'i}.$$

Observe that such a prediction only considers the influence of those items a user has purchased; it *ignores those items that the user has not purchased.* However, LRec employs a *learned* user-item affinity α_{ui}. This means that LRec can take into consideration *even those items that are not purchased by user u.*

Compared to user-based neighbourhood methods, we can see LRec as a principled means of learning a similarity matrix. In user-based neighbourhood methods, akin to Equation 2.9, we have

$$\hat{\mathbf{R}} = \mathbf{S}\mathbf{R}.$$

This is exactly the same form as the prediction from LRec (Equation 5.2), with the choice $\mathbf{S} = \mathbf{W}^T$. Crucially, however, the matrix $\mathbf{S}$ in a neighbourhood method is *not learned* by optimising some objective function.

In fact, user-based neighbourhood methods have historically not been favoured because of the difficulty in selecting a good $\mathbf{S}$ [Linden et al., 2003]. This is because the rows of $\mathbf{R}$ are typically sparser than its columns, i.e. most users tend to purchase a few items, while some items may have purchases by many users. Consequently, standard choices for $\mathbf{S}$, such as cosine similarity, will be highly noisy (and often uninformatively just 0), as one expects any two users to have only a few common purchases. By contrast, since LRec attempts to *learn* $\mathbf{S}$ from a principled objective, it has the ability to overcome the sparsity issue.

5.4.3 Relation to matrix factorisation

Compared to matrix factorisation approaches, LRec does not involve projection of users and items into a shared latent space. However, we can interpret factorisation approaches as follows. Consider WRMF as discussed in 2.12. Then, the optimal solution for the item latent factors is clearly

$$\mathbf{B} = (\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{A}\mathbf{R}.$$

This means that the factorisation equally predicts

$$\hat{\mathbf{R}} = \mathbf{S}\mathbf{R},$$

where $\mathbf{S} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{A}$ is a rank K matrix. (A similar analysis was performed in Hu et al. [2008], while Cremonesi et al. [2010] explicitly finds $\mathbf{S}$ by performing an SVD of $\mathbf{R}$.) This is again exactly the same form as the prediction from LRec (Equation 5.2). Thus, we can view matrix factorisation as solving a surrogate to

$$\min_{\mathbf{W} \in \mathcal{C}} \sum_{u \in U} \sum_{i \in I} \ell(\mathbf{y}_i^{(u)}, \mathbf{R}_{:i}^T \mathbf{w}^{(u)}) + \Omega(\mathbf{W}),$$

where $\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{m \times m} \colon \operatorname{rank}(\mathbf{W}) = K\}$.

There are thus two main advantages to LRec over such approaches.

Convexity of objective. Matrix factorisation imposes the constraint that the weight matrix is rank K. This can be viewed as performing reduced rank regression [Reinsel and Velu, 1998], an interpretation that has been noted previously [Yu and Tresp, 2005]. While a low-rank assumption is plausible, it makes the resulting optimisation non-convex. By contrast, LRec allows $\mathbf{S}$ to be full-rank, and uses an ℓ_2 penalty to prevent overfitting, resulting in a strongly convex objective.

Parallelisation w/o distributed communication. Matrix factorisation models can be trained via alternating least squares (ALS) [Hu et al., 2008]. For a fixed set of item latent features $\mathbf{B}$, we can learn the user latent features $\mathbf{A}$ by performing least squares using $\mathbf{B}$ as a design matrix and, for each $u \in U$, a target vector $\mathbf{y}^{(u)} = \mathbf{R}^T_{u:}$. As in LRec, this operation can be parallelised across users. However, when it comes to learning the item latent features, all distributed nodes must send their estimates of $\mathbf{A}$ to a central source. As ALS typically requires several such alternation iterations, this communication may become expensive. By contrast, in LRec, such distributed communication is not required: both learning and recommendation for all users can be done completely independently.

5.5 LRec for rating prediction?

The most natural natural to ask is whether LRec can be applied for rating prediction problem i.e explicit feedback setting where $\mathbf{R} \in \mathbb{R}^{m \times n}$. Unfortunately, LRec cannot be applied in such setting for two reasons. First, we cannot assume the unobserved entries (regression targets) as zeros for the rating prediction task. Second, as a consequence, if we learn only from the observed data, there are too few of them for a regression model to generalise. However, LRec can be applied on rating data for a ranked list recommendation as such task is only concerned with relative scores.

5.6 LinearFlow: Fast low rank linear model

Despite being highly parallelisable, LRec requires solving a large number of regression subproblems on huge design matrix $\mathbf{R}^T$ making it challenging on large scale datasets (SLIM also suffers from the same limitation). Further, the memory requirement is quadratic on the number of m or n. Hence, the applicability of Linear methods, both LRec and SLIM, on real world large scale dataset is challenging. In this Section, we formulate an algorithm leveraging closed-form solution of squared loss for scaling up linear methods to large OC-CF problems.

For squared loss, we can write Equation 5.1 as

$$\operatorname{argmin} \left\| \mathbf{R}^T - \mathbf{R}^T \mathbf{W} \right\|_F^2 + \lambda \left\| \mathbf{W} \right\|_F^2, \tag{5.4}$$

where the optimal solution for W is given as

$$\mathbf{W} = (\mathbf{R}\mathbf{R}^T + \lambda \mathbf{I})^{-1} \mathbf{R}\mathbf{R}^T, \tag{5.5}$$

which involves the computing inverse of large matrix and therefore does not scale to large problems.

To address this limitation, we seek an approximation $\mathbf{R} \approx \mathbf{RW}$ that attempts to capture most of the row space of the matrix $\mathbf{R}$ through a matrix $\mathbf{W}$ that is low-rank *and* has small Frobenius norm.

$$\underset{\operatorname{rank}(\mathbf{W}) \leq k}{\operatorname{argmin}} \left\| \mathbf{R}^T - \mathbf{R}^T \mathbf{W} \right\|_F^2 + \lambda \left\| \mathbf{W} \right\|_F^2, \tag{5.6}$$

where $k \ll m$. The motivation for such a double regularization is to better control the generalization error of the model, which we will prove via experiments. Moreover, it turns out that there is a natural and efficient way to compute such an approximate decomposition of the matrix $\mathbf{R}$ by randomization [Halko et al., 2011], which allows scaling to large problems.

For $\lambda = 0$, the optimal solution for W in Equation 5.6 is given by the Eckart-Young theorem (see, e.g., [Halko et al., 2011])

$$\mathbf{W} = \mathbf{Q}_k \mathbf{Q}_k^T, \tag{5.7}$$

where $\mathbf{Q}_k$ is an orthogonal matrix computed by a truncated SVD

$$\mathbf{R}^T \approx \mathbf{P}_k \mathbf{\Sigma}_k \mathbf{Q}_k^T, \tag{5.8}$$

However, under both a low-rank constraint and $\lambda > 0$ in (5.6), finding the optimal $\mathbf{W}$ involves solving a hard nonconvex problem with no analytical solution in general. Nonetheless, an analytical solution is possible for a certain parametrization of $\mathbf{W}$ as we explain next. We first compute an approximate orthogonal basis $\mathbf{Q}_k$ of the row space of $\mathbf{R}^T$, i.e.,

$$\mathbf{R}^T \approx \mathbf{R}^T \mathbf{Q}_k \mathbf{Q}_k^T \tag{5.9}$$

using randomised SVD. In Appendix A, we outline the randomised SVD algorithm (we refer to [Halko et al., 2011] for more details.) Then we re-parametrize the matrix $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{Q}_k \mathbf{Z} \tag{5.10}$$

for some matrix $\mathbf{Z}$. Note that through this parametrization the rank of $\mathbf{W}$ is automatically controlled, no optimality is lost when $\lambda = 0$, and the optimization problem (5.6) reads

$$\underset{\mathbf{Z}}{\operatorname{argmin}} \left\| \mathbf{R}^T - \mathbf{R}^T \mathbf{Q}_k \mathbf{Z} \right\|_F^2 + \lambda \left\| \mathbf{Q}_k \mathbf{Z} \right\|_F^2 . \tag{5.11}$$

Since $\mathbf{Q}_k$ is orthogonal, we have $\|\mathbf{Q}_k \mathbf{Z}\|_F = \|\mathbf{Z}\|_F$, and (5.11) becomes

$$\underset{\mathbf{Z}}{\operatorname{argmin}} \left\| \mathbf{R}^T - \mathbf{R}^T \mathbf{Q}_k \mathbf{Z} \right\|_F^2 + \lambda \left\| \mathbf{Z} \right\|_F^2 . \tag{5.12}$$

The latter can be solved analytically to give

$$\mathbf{Z} = (\mathbf{Q}_k^T \mathbf{R} \mathbf{R}^T \mathbf{Q}_k + \lambda \mathbf{I})^{-1} \mathbf{Q}_k^T \mathbf{R} \mathbf{R}^T.$$

Note that this inversion involves a $k \times k$ matrix, and hence it can be computed efficiencly for decent value of k. Same as we discussed in Section 4.3.2, $(\mathbf{Q}_k^T \mathbf{R} \mathbf{R}^T \mathbf{Q}_k + \lambda \mathbf{I})$ is a diagonal matrix for exact SVD and its inverse can be computed trivially. However, in randomised SVD, Q_k is an approximate, but we expcet this to hold approximately.

In other words, the choice of $\mathbf{W} = \mathbf{Q}_k \mathbf{Z}$ is motivated by the following observation. When $\lambda = 0$, the solution to our problem is $\mathbf{W} = \mathbf{Q}_k \mathbf{Q}_k^T$. This is also the solution to the new formulation of our problem for $\mathbf{Z} = \mathbf{Q}_k^T$. When λ is close to zero, we believe that sufficiently good solutions lie close to the span of $\mathbf{Q}_k$. Therefore, we choose $\mathbf{W} = \mathbf{Q}_k \mathbf{Z}$. We demonstrate that this choice performs well empirically in the experimental section.

We refer to (5.11) as U-LINEAR-FLOW as it corresponds to user-user LRec model. Similarly, we can define an item-item model, I-LINEAR-FLOW

$$\underset{\mathbf{Z}}{\operatorname{argmin}} \left\| \mathbf{R} - \mathbf{R} \mathbf{P}_k \mathbf{Z} \right\|_F^2 + \lambda \left\| \mathbf{Z} \right\|_F^2 . \tag{5.13}$$

In U-LINEAR-FLOW model, the user-user similarity can be recovered as $\mathbf{W} = \mathbf{Q}_k\mathbf{Z}$. Similarly, in I-LINEAR-FLOW model, the item-item similarity can be recovered as $\mathbf{W} = \mathbf{P}_k\mathbf{Z}$.

The recommendation matrix for LINEAR-FLOW , say U-LINEAR-FLOW, is given as

$$\hat{\mathbf{R}} = \mathbf{R}\mathbf{R}^T\mathbf{Q}_k((\mathbf{Q}_k^T\mathbf{R}\mathbf{R}^T\mathbf{Q}_k + \lambda\mathbf{I})^{-1})^T\mathbf{Q}_k^T\mathbf{R}.$$

5.7 Experiments and evaluation of LREC model

In this Section, we discuss datasets, experimental methodology and report detailed experimental results for the LRec model. We will discuss experiments for LINEAR-FLOW in Section 5.8. We first thoroughly evaluate the LRec model.

5.7.1 Data description

We evaluate LRecmode on four datasets. Table 5.2 summarises statistics of these datasets.

Table 5.2: Summary of datasets used in evaluation.

Dataset	m	n	$\lvert\mathbf{R}_{ui} > 0\rvert$
ML1M	6,038	3,533	575,281
KOBO	38,868	170,394	89,815
LASTFM	992	107,398	821,011
MSD	1,019,318	384,546	48,373,586

ML1M. The MovieLens 1M dataset[4] is a standard benchmark for collaborative filtering tasks. Following the “Who Rated What” KDD Cup 2007 challenge [Bennett et al., 2007a], we created a binarised version of the dataset suitable for evaluating implicit feedback methods. From the original rating matrix $\mathbf{R} \in \{0, 1, \ldots, 5\}^{m\times n}$, we created a preference matrix $\tilde{\mathbf{R}}$ with $\tilde{\mathbf{R}}_{ui} = [\![\mathbf{R}_{ui} \geq 4]\!]$. We then used this as input to all methods.

KOBO. The Kobo dataset comes from Kobo Inc.[5], a major online ebook retailer with more than 20 million readers. This is an anonymized dataset of ebook purchases of a subset of about 40,000 Kobo users; these users purchased around 90,000 books.

LASTFM. The LastFM dataset[6] [Celma, 2008] contains the play counts of $\sim$1000 users on $\sim$170,000 artists. As per ML1M, we binarised the raw play counts.

MSD. The Million Song dataset (MSD)[7] [Bertin-Mahieux et al., 2011] comprises the play counts for $\sim$1M users on $\sim$300,000 songs. The goal is to predict the listening preferences for $\sim$100K test users. As per ML1M, we binarised the raw play counts.

5.7.2 Evaluation protocol

For the ML1M and LASTFM datasets, we estimate performance based on 10 random train-test splits, similar to [Pan et al., 2008, Johnson, 2014]. If $\mathbf{R}$ is the full user-item purchase matrix, then for each split, we select a random 20% subset of all (user, item) pairs (regardless of whether they were purchased

[4] `http://grouplens.org/datasets/movielens/`
[5] `http://www.kobo.com`
[6] `http://ocelma.net/MusicRecommendationDataset/index.html`
[7] `http://labrosa.ee.columbia.edu/millionsong/`

or not), and place them in the test set. We report the mean test split performance, along with standard errors corresponding to 95% confidence intervals.

For KOBO, we followed the above except using *temporally* divided train and test splits. To prepare the training dataset, we include all purchase data prior to a specific date, starting from June 2012 and incremented by a month in each split. The test set includes all books purchased by users in the week following the train data. We use the last purchased item in the training set for each user as a validation set, used for fine-tuning any hyperparameters.

For MSD, we use the provided training set to learn all models. We evaluate perform on 10 *random subsets* of the provided test set. In each such random subset, we pick 500 random test users to evaluate performance on.

To evaluate the performance of the various recommenders, we report Precision@k for $k \in \{3, 5, 10, 20\}$ (averaged across all test fold users), and mean average precision (mAP@100).

5.7.3 Model comparison

In this Section, we discuss various baselines we used to compare with LRec model:

- Recommending the most popular items to each user (Popularity), after removing those items that the user has already purchased in the training set. This can be competitive if many users consume items from the short tail. This is a minimal baseline that any competing method must outperform to be considered useful.
- User- and item-based nearest neighbour (U-KNN and I-KNN), as discussed in Section 2.4.2, using both cosine similarity and Jaccard coefficient to define the similarity matrix $\mathbf{S}$. For each dataset, we picked the best performing of the two metrics.
- PURESVD of [Cremonesi et al., 2010], as discussed in Section 2.4.2.
- Weighted matrix factorisation (WRMF), as discussed in Section 2.4.2.
- Logistic matrix factorisation (LOGISTICMF) [Johnson, 2014], This is as per Equation 2.1 with the weights $\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} > 0]\!]$, and ℓ being the logistic loss.
- Bayesian Personalised Ranking (BPR), Equation 2.17 in Section 2.4.2.
- SLIM, as discussed in Section 2.4.2. For computational convenience, we used the SGDReg variant [Levy and Jack, 2013], which removes the nonnegativity constraint. It is identical to SLIM except that the nonnegativity constraint is removed.
- Two variants of LRec, one which uses square rather than logistic loss (LREC+SQ), and one which additionally employs ℓ_1 regularisation and nonnegativity of weights (LREC+SQ+ℓ_1+NN). The weight constraints in the latter approach are as employed by SLIM.

We use LRec and LRec+Sq to denote the proposed method with logistic and squared loss respectively. LRec+Sq can be seen as a simplification of U-SLIM, where there is no ℓ_1 regularisation, no nonnegativity constraint, and no constraint on the diagonal of $\mathbf{W}$.

For WRMF and BPR, we used the implementation as provided in the MyMediaLite package[8]. For SLIM, we used the SGDReg implementation as provided in the MRec package[9]. For LOGISTICMF and LREC, we used our own implementation in NumPy.

[8]http://www.mymedialite.net/
[9]http://www.recsyswiki.com/wiki/Mrec

For the neighbourhood methods, we tuned the neighbourhood size from $\{10, 20, 40, 80, 120, 160, n/m\}$. For methods relying on a latent dimension K, we tuned this from $\{10, 20, 40, 80, 120, 160\}$. For methods relying on ℓ_2 regularisation, we tuned this from $\lambda \in \{1000, 100, 10, 1, 0.1, 0.01, 0.001, 0.0001\}$. For WRMF, the weight α (Equation 2.12) was tuned from $\{1, 5, 10, 20, 40, 80\}$.

5.7.4 Results and analysis

Table 5.3: Results on ML1M dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
POPULARITY	0.0685 ± 0.001	0.1645 ± 0.002	0.1505 ± 0.001	0.1316 ± 0.001	0.1106 ± 0.001
U-KNN	0.1564 ± 0.001	0.3125 ± 0.001	0.2793 ± 0.001	0.2309 ± 0.001	0.1816 ± 0.001
I-KNN	0.1473 ± 0.001	0.2998 ± 0.001	0.2720 ± 0.001	0.2287 ± 0.001	0.1826 ± 0.001
PURESVD	0.1611 ± 0.001	0.3245 ± 0.002	0.2910 ± 0.003	0.2434 ± 0.002	0.1932 ± 0.002
WRMF	0.1682 ± 0.002	0.3297 ± 0.002	0.2969 ± 0.002	0.2474 ± 0.002	0.1975 ± 0.002
LOGISTICMF	0.0683 ± 0.002	0.1644 ± 0.001	0.1498 ± 0.001	0.1306 ± 0.002	0.1095 ± 0.001
BPR	0.1611 ± 0.001	0.3060 ± 0.001	0.2822 ± 0.001	0.2402 ± 0.001	0.1928 ± 0.002
I-SLIM	0.1751 ± 0.001	0.3421 ± 0.002	0.3101 ± 0.001	0.2590 ± 0.001	0.201 ± 0.001
LREC+SQ+ℓ_1+NN	0.1648 ± 0.003	0.3325 ± 0.003	0.3019 ± 0.004	0.2558 ± 0.003	0.1981 ± 0.002
LREC	**0.1806 ± 0.001**	**0.3514 ± 0.004**	**0.3154 ± 0.001**	**0.2622 ± 0.002**	**0.2078 ± 0.001**
LREC+SQ	0.1762 ± 0.001	0.3498 ± 0.002	0.3131 ± 0.001	0.2588 ± 0.001	0.2031 ± 0.001

Table 5.4: Results on LASTFM dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
POPULARITY	0.0859 ± 0.001	0.4061 ± 0.004	0.3632 ± 0.005	0.3184 ± 0.003	0.2721 ± 0.002
U-KNN	0.1483 ± 0.002	0.5454 ± 0.003	0.5078 ± 0.001	0.4551 ± 0.001	0.3968 ± 0.002
I-KNN	0.1684 ± 0.002	0.5775 ± 0.001	0.5503 ± 0.002	0.5001 ± 0.001	0.4333 ± 0.001
PURESVD	0.1703 ± 0.001	0.6019 ± 0.008	0.5658 ± 0.005	0.5127 ± 0.003	0.4441 ± 0.002
WRMF	0.1825 ± 0.001	0.6254 ± 0.012	0.5898 ± 0.009	0.5286 ± 0.006	0.4587 ± 0.002
LOGISTICMF	0.0847 ± 0.001	0.4052 ± 0.002	0.3675 ± 0.001	0.3207 ± 0.001	0.2711 ± 0.001
BPR	0.1497 ± 0.002	0.5001 ± 0.035	0.4746 ± 0.018	0.4332 ± 0.021	0.3890 ± 0.012
I-SLIM	0.0309 ± 0.003	0.0539 ± 0.005	0.0491 ± 0.003	0.0404 ± 0.003	0.0316 ± 0.004
LREC+SQ+ℓ_1+NN	**0.2030 ± 0.001**	**0.6492 ± 0.001**	**0.6147 ± 0.003**	**0.5580 ± 0.002**	**0.4801 ± 0.001**
LREC	0.1833 ± 0.001	0.6010 ± 0.010	0.5700 ± 0.008	0.5173 ± 0.004	0.4527 ± 0.002
LREC+SQ	**0.2038 ± 0.001**	**0.6498 ± 0.010**	**0.6185 ± 0.005**	**0.5589 ± 0.004**	**0.4875 ± 0.001**

In this Section, we discuss the experimental results. Tables 5.3 – 5.6 summarise the results of the various methods. We make the following observations:

- LRec consistently outperforms *all* other methods by a statistically significant margin. In terms of the mAP@100 score, the % improvement over the next best method is 7.3% on ML1M, 2.5% on KOBO, 0.4% on LASTFM, and 14% on MSD.
- LREC+SQ generally is competitive with LRec, and on the KOBO and LASTFM datasets is in fact superior. As the difference between the two approaches is simply the choice of loss function, and since using square loss in a classification context is equivalent to performing Fisher's Linear Discriminant [Bishop, 2006, Section 4.1.5], this is unsurprising.

Table 5.5: Results on KOBO dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
POPULARITY	0.0364 ± 0.015	0.0121 ± 0.005	0.0145 ± 0.007	0.0134 ± 0.007	0.0101 ± 0.005
U-KNN	0.1164 ± 0.029	0.0563 ± 0.015	0.0411 ± 0.012	0.0269 ± 0.006	0.0170 ± 0.003
I-KNN	0.1115 ± 0.010	0.0548 ± 0.006	0.0417 ± 0.004	0.0265 ± 0.003	0.0160 ± 0.002
PURESVD	0.0902 ± 0.009	0.0410 ± 0.013	0.0301 ± 0.009	0.0178 ± 0.010	0.0101 ± 0.009
WRMF	0.0972 ± 0.022	0.0472 ± 0.022	0.0337 ± 0.014	0.0200 ± 0.007	0.0123 ± 0.004
LOGISTICMF	0.0415 ± 0.021	0.0172 ± 0.011	0.0132 ± 0.007	0.0115 ± 0.004	0.0101 ± 0.003
BPR	0.0616 ± 0.025	0.0280 ± 0.014	0.0247 ± 0.009	0.0198 ± 0.006	0.0144 ± 0.003
I-SLIM	0.1092 ± 0.018	0.058 ± 0.011	0.0393 ± 0.007	0.0253 ± 0.004	0.0164 ± 0.002
LREC+SQ+ℓ_1+NN	0.1250 ± 0.013	0.0610 ± 0.007	0.0442 ± 0.004	0.0289 ± 0.001	0.0184 ± 0.001
LREC	0.1247 ± 0.020	0.0605 ± 0.010	0.0437 ± 0.006	0.0284 ± 0.003	0.0179 ± 0.002
LREC+SQ	$\mathbf{0.1282 \pm 0.017}$	$\mathbf{0.0619 \pm 0.008}$	$\mathbf{0.0462 \pm 0.005}$	$\mathbf{0.0294 \pm 0.002}$	$\mathbf{0.0189 \pm 0.002}$

Table 5.6: Results on MSD. Reported numbers are on the provided test set.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
POPULARITY	0.0181 ± 0.006	0.0466 ± 0.008	0.044 ± 0.005	0.035 ± 0.003	0.024 ± 0.002
U-KNN	0.1075 ± 0.008	0.2145 ± 0.021	0.1883 ± 0.020	0.1366 ± 0.013	0.0912 ± 0.007
I-KNN	0.1548 ± 0.011	0.3117 ± 0.072	0.2740 ± 0.063	0.2125 ± 0.038	0.1483 ± 0.016
PURESVD	Did not finish in 24 hours				
WRMF	Did not finish in 24 hours				
LOGISTICMF	Did not finish in 24 hours				
BPR	Did not finish in 24 hours				
I-SLIM	Did not finish in 24 hours				
LREC+SQ+ℓ_1+NN	0.1481 ± 0.020	0.3030 ± 0.018	0.2760 ± 0.025	0.2060 ± 0.035	0.1450 ± 0.018
LREC	$\mathbf{0.1766 \pm 0.013}$	$\mathbf{0.3301 \pm 0.012}$	$\mathbf{0.3020 \pm 0.031}$	$\mathbf{0.2310 \pm 0.038}$	$\mathbf{0.1640 \pm 0.016}$
LREC+SQ	0.1531 ± 0.012	0.3033 ± 0.012	$0.2760 \pm .028$	0.2160 ± 0.021	0.1520 ± 0.015

- LREC+SQ consistently outperforms LREC+SQ+ℓ_1+NN. This suggests that the constraints on $\mathbf{W}$ and ℓ_1 regularisation imposed by SLIM are harmful for retrieval.

- Generally, item-based neighbourhood methods perform better than the user-based counterparts. This matches our earlier observation that the high sparsity across rows of $\mathbf{R}$ results in a less informative estimate of $\mathbf{S}$. We similarly observe that LREC+SQ+ℓ_1+NN is generally better than I-SLIM.

- Neighbourhood methods are competitive with their matrix factorisation counterparts. This highlights the different challenge in the implicit compared to explicit feedback settings, as well as in top-N recommendation versus rating prediction.

- Amongst the matrix factorisation methods, WRMF performs the best in all datasets, and is competitive with the neighbourhood methods. BPR is seen to generally under perform. This is possibly due to the fact that it optimises for AUC, i.e. average performance across all thresholds, rather than for performance at the head of the ranked list.

- On all but the ML1M dataset, the number of (test) users is significantly fewer than the number of items. Therefore, methods that are item-focussed tend to strongly under perform. For example, on the LASTFM dataset, the otherwise competitive I-SLIM method performs much worse than all other methods.

- The scale of MSD is challenging for learning, in particular for all of the matrix factorisation methods. For example, WRMF, which is trained used alternating least squares, did not finish running in a day. Exploiting subsampling of negative instances, as done for LRec, is crucial to attain scalability. Compared to other scalable methods, such as the neighbourhood methods, LRec's performance is significantly superior.

5.7.5 Long-tail recommendations

We study how LRec fares in recommending items in the long-tail. Following [Cremonesi et al., 2010, Ning and Karypis, 2011], we take the short-tail to comprise the smallest set of items which account for 20% of all purchases. The long-tail comprises all other items. On the ML1M dataset for example, we find that the long tail comprises 99% of items. Figure 5.1 compares the performance of various methods when the short-tail items are removed from the training set, in terms of Prec@20. As expected, the performance of all methods drops significantly compared to the results in Table 5.3. However, LRec remains the best performing of all methods in this scenario.

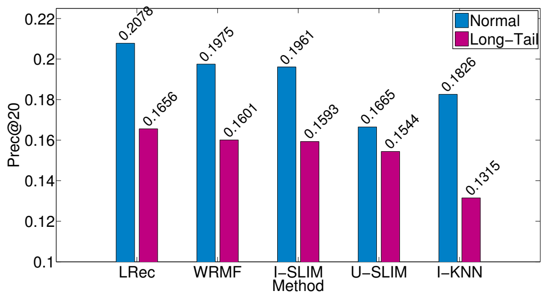


Figure 5.1: Long-tail results for LRec on ML1M dataset.

5.7.6 Near cold-start recommendation

Tables 5.3 – 5.6 summarise retrieval performance across all users. However, different users will have rated different numbers of items. It is of interest to see how LRec compares to baselines as a function of the numbers of ratings provided by a user. To do this, we segment users on the ML1M dataset based on the 1%, 25%, 50%, 75% and 99% quantiles of the number of training ratings. For each segment, we then plot the % improvement in terms of Prec@20 of LRec over WRMF.

Figure 5.3 shows the improvements of LRec across these segments. We see that, reassuringly, LRec is significantly superior to WRMF over all user segments. Of interest is that we see dramatic improvements in Prec@20 for LRec over WRMF for users that provide a small number (1 – 6) of ratings. This shows that LRec is especially useful in near cold-start recommendation scenarios.

At the same time, we expect that recommendation performance will be positively correlated with the number of training ratings for a user. Figure 5.3 confirms that as a function of the number of ratings, the Prec@20 steadily increases for LRec. (While recommendation performance is not very high for low numbers of training ratings, Figure 5.2 shows it is nonetheless significantly superior to the performance WRMF in this scenario.)

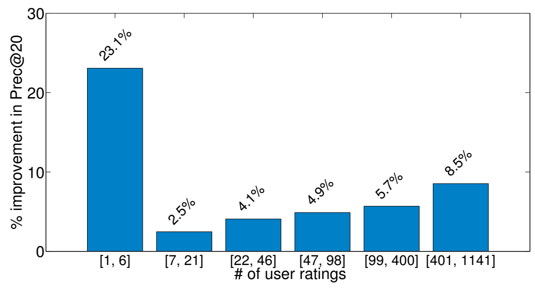


Figure 5.2: Improvement in Prec@20 of LRec over WRMF for users with varying number of training ratings, ML1M dataset.

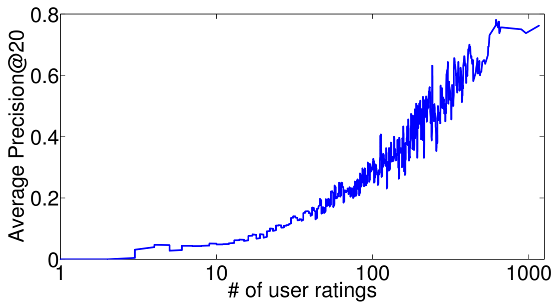


Figure 5.3: Retrieval performance of LRec as a function of number of training ratings, ML1M dataset.

5.7.7 Case-Study

We now attempt to translate the above performance gains into qualitative changes in recommendation. On the ML1M dataset, we compare the top 10 recommendations generated by LRec and WRMF for a user with relatively few training ratings (16). Table 5.7 shows a selection of this user's preferred movies on the training set, as well as the resulting recommendations. We see that the user's training preferences are towards sci-fi movies, which are picked up by both WRMF and LRec. However, we see that more specifically, the user enjoys "B-grade" Hollywood sci-fi movies, often themed on "alien invasion", with movies such as "It Came from Beneath the Sea" and "It Conquered the World". This is not picked up by WRMF, which recommends more "cerebral" sci-fi movies such as "2001" and "Gattaca". By contrast, LRec captures the user's taste at a finer granularity, resulting in successfully capturing all three preferred movies in the user's test set.

The above illustrates a general phenomenon: in addition to producing accurate recommendations for users with few training ratings, LRec produces accurate recommendations for *non-mainstream users*, i.e. users whose preferences are for items that are not popular. As an analogue to Figure 5.2, Figure 5.4 shows the % improvement in Prec@20 scores for LRec over WRMF, where users are segmented according to the average popularity of their training set movies. Here, the popularity of a movie is simply the number of training purchases of the movie. We see that for non-mainstream users, LRec achieves significant improvement over WRMF. This illustrates that LRec captures user preferences at a finer granularity.

As further illustration of LRec's ability to generalise from limited information, we can compare the recommendation performance of LRec and WRMF as a function of item popularity. For each item, we compute the number of times the item is correctly recommended in the top 20 list for LRec and WRMF (i.e. it is recommended and it is preferred in the user's test set), normalised by the number of times the item is preferred by a user. We call this the % of correct recommendations for an item. Figure 5.5 plots this as a function of the popularity of each movie. While both methods are comparable for items in the

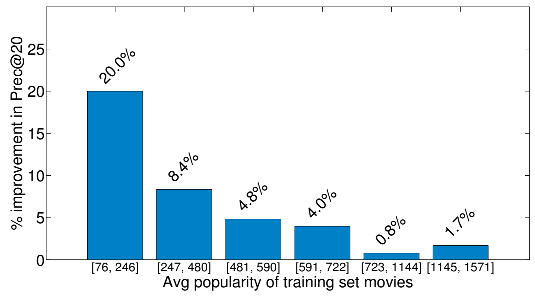


Figure 5.4: Improvement in Prec@20 of LRec over WRMF for users with varying average popularity of their training set movies, ML1M dataset.

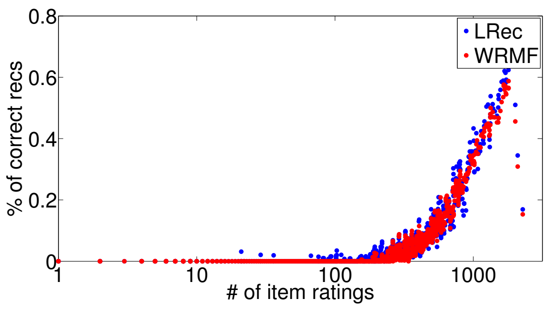


Figure 5.5: Comparison of % of correct recommendations for movies by LRec and WRMF, ML1M dataset. See text for details.

Table 5.7: Preferred movies on the training set for a candidate user, and resulting top 10 recommendations on ML1M dataset. Bolded entries are actually enjoyed by user on test set, which is shown as the last column. The training set movies are a subset of the 16 that the user enjoys.

Preferred training movies	**WRMF recommendations**	**LRec recommendations**	**Preferred test movies**
• Day the Earth Stood Still, The	• Planet of the Apes	• **Them!**	• Blob, The
• Forbidden Planet	• Thing, The	• Godzilla (Gojira)	• Them!
• Kronos	• Night of the Living Dead	• **Blob, The**	• It Came from Outer Space
• Tarantula	• Star Trek: The Wrath of Khan	• 20,000 Leagues Under the Sea	
• Thing From Another World, The	• Fly, The	• Soylent Green	
• War of the Worlds, The	• Alien	• Village of the Damned	
• It Came from Beneath the Sea	• Dark City	• Metropolis	
• Invasion of the Body Snatchers	• Star Trek IV: The Voyage Home	• Quatermass and the Pit	
• Earth Vs. the Flying Saucers	• 2001: A Space Odyssey	• **It Came from Outer Space**	
• It Conquered the World	• Gattaca	• Plan 9 from Outer Space	

short tail, we notice that WRMF never recommends movies with less than 150 ratings, i.e. "extreme long tail" movies. By contrast, LRec is able to correctly recommend movies with as few as 21 ratings.

5.8 Experiments and evaluation of LINEAR-FLOW model

In this Section, we discuss datasets, experimental methodology and report detailed experimental results for LINEAR-FLOW model. We report experiments on a larger collections of datasets, where LRec is not applicable.

5.8.1 Data description

First, we evaluate LINEAR-FLOW with top-performing models from the Section 5.7.1. Further, we evaluate LINEAR-FLOW on three additional large-scale datasets as summarised in Table 5.8. In these three datasets, we remove users with fewer that 3 corresponding items and vice-versa.

Table 5.8: Summary of datasets used in evaluation.

Dataset	m	n	$\lvert\mathbf{R}_{ui} > 0\rvert$
ML10M	69,613	9,405	5,004,150
PROPRIETARY-1	26,928	14,399	120,268
PROPRIETARY-2	264,054	57,214	1,398,332

ML1M The MovieLens 10M dataset[10] is a standard benchmark for collaborative filtering tasks. Following the "Who Rated What" KDD Cup 2007 challenge [Bennett et al., 2007a], we created a binarized version of the dataset suitable for evaluating implicit feedback methods. From the original rating matrix $\tilde{\mathbf{R}} \in \{0, 1, \ldots, 5\}^{m \times n}$, we created a binarized preference matrix $\mathbf{R}$ with $R_{ui} = [\![\tilde{\mathbf{R}}_{ui} \geq 4]\!]$.

PROPRIETARY-1 & PROPRIETARY-2 are two real but anonymized purchase datasets. PROPRIETARY-1 dataset consists of ∼27,000 users, ∼14,000 items and ∼120,000 item purchases. Similarly, PROPRIETARY-2 dataset consists of ∼264,000 users, ∼57,000 items and ∼1 million item purchases.

5.8.2 Evaluation protocol

First, we evaluate the performance of LINEAR-FLOW on the datasets defined in 5.7.1 following the same evaluation strategy as discussed in 5.7.2. For the ML10M , PROPRIETARY-1 , and PROPRIETARY-2 dataset, we split the datasets into random 90%-10%[11] train-test set and hold out 10% of the training set for hyperparamater tuning. We report the mean test split performance, along with standard errors corresponding to 95% confidence intervals. To evaluate the performance of the various recommenders, we report precision@k and recall@k for $k \in \{3, 5, 10, 20\}$ (averaged over test users), and mean average precision (mAP@20) [12].

5.8.3 Model comparison

First, we compare LINEAR-FLOW with the top performing models as discussed in the section 5.7.4. For the additional three datasets, we compared the proposed method to the following baselines:

- User- and item-based nearest neighbour (U-KNN and I-KNN). For each dataset, we use Jaccard and Cosine similarity metric and picked the best performing one.
- PURESVD of [Cremonesi et al., 2010], as discussed in Section 2.4.2.
- Weighted matrix factorisation (WRMF), as discussed in Section 2.4.2.
- MF-RSVD of [Tang and Harrington, 2013], as discussed in Section 2.4.2. We ran this method with user and item based initialization, U-MF-RSVD and I-MF-RSVD, as discussed in Eq. (2.16) and (2.15) respectively.
- SLIM, as discussed in Section 2.4.2. For computational convenience, we used the SGDReg variant [Levy and Jack, 2013], which removes the nonnegativity constraint. It is identical to SLIM except that the nonnegativity constraint is removed.

[10] `http://grouplens.org/datasets/movielens/`

[11] We choose 90%-10% split due to the sparsity of the datasets.

[12] The choice of map@20 instead of map@100 is due to computational reason. Further, it is resonable to do so as map focuses on the top of the ranked list.

We do not compare against LRec due to its memory complexity on a large dataset. For instance on the PROPRIETARY-2 dataset, LRec requires ∼260GB of memory. Also, we didn't compare against the methods, such as BPR, and LOGISTICMF, due to its inferiror performance as observed in Section 5.7.4.

5.8.4 Results and analysis

Evaluation of LINEAR-FLOW with other linear methods

In Table 5.9 – 5.11, we compare LINEAR-FLOW methods to state-of-the-art linear models. The result demonstrates that LINEAR-FLOW yields competitive results, sometimes superiror, compared to the other methods (with a significant reduction in computational cost, as we will show shortly). In the following section, we thoroughly compare LINEAR-FLOW with state-of-the-art OC-CF methods on large-scale datasets.

Table 5.9: Results on ML10M dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
I-SLIM	0.1751 ± 0.001	0.3421 ± 0.002	0.3101 ± 0.001	0.2590 ± 0.001	0.201 ± 0.001
LREC+SQ+ℓ_1+NN	0.1648 ± 0.003	0.3325 ± 0.003	0.3019 ± 0.004	0.2558 ± 0.003	0.1981 ± 0.002
LREC	0.1806 ± 0.001	0.3514 ± 0.004	0.3154 ± 0.001	0.2622 ± 0.002	0.2078 ± 0.001
LREC+SQ	0.1762 ± 0.001	0.3498 ± 0.002	0.3131 ± 0.001	0.2588 ± 0.001	0.2031 ± 0.001
I-LINEAR-FLOW	0.1854 ± 0.001	0.3525 ± 0.002	0.3194 ± 0.001	0.2666 ± 0.001	0.2111 ± 0.001
U-LINEAR-FLOW	**0.1859** ± 0.001	**0.3526** ± **0.002**	**0.3194** ± **0.003**	**0.2672** ± **0.002**	**0.2115** ± **0.002**

Table 5.10: Results on LASTFM dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
I-SLIM	0.0309 ± 0.003	0.0539 ± 0.005	0.0491 ± 0.003	0.0404 ± 0.003	0.0316 ± 0.004
LREC+SQ+ℓ_1+NN	**0.2030 ± 0.001**	**0.6492 ± 0.001**	**0.6147 ± 0.003**	**0.5580 ± 0.002**	**0.4801 ± 0.001**
LREC	0.1833 ± 0.001	0.6010 ± 0.010	0.5700 ± 0.008	0.5173 ± 0.004	0.4527 ± 0.002
LREC+SQ	**0.2038 ± 0.001**	**0.6498 ± 0.010**	**0.6185 ± 0.005**	**0.5589 ± 0.004**	**0.4875 ± 0.001**
I-LINEAR-FLOW	0.1913 ± 0.001	0.6217 ± 0.006	0.5862 ± 0.006	0.5319 ± 0.004	0.4624 ± 0.002
U-LINEAR-FLOW	0.1869 ± 0.001	0.6210 ± 0.006	0.5854 ± 0.003	0.5274 ± 0.004	0.4592 ± 0.002

Table 5.11: Results on KOBO dataset. Reported numbers are the mean and standard errors across test folds.

Method	**mAP@100**	**Prec@3**	**Prec@5**	**Prec@10**	**Prec@20**
I-SLIM	0.1092 ± 0.018	0.058 ± 0.011	0.0393 ± 0.007	0.0253 ± 0.004	0.0164 ± 0.002
LREC+SQ+ℓ_1+NN	0.1250 ± 0.013	0.0610 ± 0.007	0.0442 ± 0.004	0.0289 ± 0.001	0.0184 ± 0.001
LREC	0.1247 ± 0.020	0.0605 ± 0.010	0.0437 ± 0.006	0.0284 ± 0.003	0.0179 ± 0.002
LREC+SQ	**0.1282 ± 0.017**	**0.0619 ± 0.008**	**0.0462 ± 0.005**	**0.0294 ± 0.002**	**0.0189 ± 0.002**
I-LINEAR-FLOW	0.0967 ± 0.008	0.0631 ± 0.004	0.0495 ± 0.002	0.0341 ± 0.001	0.0226 ± 0.001
U-LINEAR-FLOW	0.0969 ± 0.008	0.0631 ± 0.004	0.0497 ± 0.002	0.0342 ± 0.001	0.0226 ± 0.001

Evaluation of LINEAR-FLOW with baselines on large-scale datasets

Tables 5.12 – 5.14 summarize the results of our methods and the various baselines. The results demonstrate that in terms of the quality of recommendation, the linear methods (LINEAR-FLOW and I-SLIM) always outperform other methods in all datasets. LINEAR-FLOW and I-SLIM perform equally well and there is little separation between them in terms of the quality of the recommendation. However LINEAR-FLOW is much more efficient than SLIM, as we will show later. Also, unlike other methods, the performance of LINEAR-FLOW is not sensitive to the choice of user vs. item based formulation. From these tables, we make the following additional observations:

- Among the matrix factorisation methods, those that use Randomised SVD [Tang and Harrington, 2013] consistently performed the best. This is possibly due to the fact that SVD provides a good initialization whereas matrix factorisation optimises a highly nonconvex bilinear objective and is sensitive to the initialization and hyperparameters. In Table 5.12 – 5.14, we observe that the performance of MF-RSVD varies significantly based on whether we initialize user or item latent factors. Also, the factorisation methods do not directly provide item-to-item or user-to-user similarity measures. Hence, they are not applicable when a recommendation of similar items or users is needed.

- We observe that the neighbourhood models yield inferior results compared to the Linear models. Also, the performance varies with the user and item based models. While they perform competitively in the PROPRIETARY-2 and PROPRIETARY-1 datasets, they perform poorly on ML10M dataset. Hence, we conclude that neighbourhood methods are not consistent with their performance.

Table 5.12: Results on the PROPRIETARY-1 dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	**prec@5**	**prec@10**	**prec@20**	**recall@3**	**recall@5**	**recall@10**	**recall@20**	**mAP@20**
I-KNN	0.039 ± 0.001	0.029 ± 0.000	0.019 ± 0.000	0.011 ± 0.000	0.080 ± 0.001	0.097 ± 0.002	0.123 ± 0.002	0.152 ± 0.003	0.072 ± 0.001
U-KNN	0.051 ± 0.001	0.039 ± 0.001	0.025 ± 0.000	0.015 ± 0.000	0.107 ± 0.004	0.132 ± 0.004	0.168 ± 0.003	0.205 ± 0.004	0.097 ± 0.003
PURESVD	0.038 ± 0.001	0.027 ± 0.001	0.016 ± 0.000	0.009 ± 0.000	0.078 ± 0.002	0.091 ± 0.002	0.107 ± 0.003	0.125 ± 0.003	0.069 ± 0.002
WRMF	0.040 ± 0.001	0.029 ± 0.001	0.018 ± 0.001	0.011 ± 0.000	0.079 ± 0.002	0.096 ± 0.005	0.119 ± 0.005	0.147 ± 0.004	0.071 ± 0.003
U-MF-RSVD	0.050 ± 0.001	0.038 ± 0.000	0.025 ± 0.000	0.015 ± 0.000	0.100 ± 0.002	0.130 ± 0.002	0.169 ± 0.003	0.207 ± 0.004	0.096 ± 0.003
I-MF-RSVD	0.048 ± 0.001	0.036 ± 0.001	0.023 ± 0.000	0.014 ± 0.000	0.098 ± 0.003	0.121 ± 0.003	0.155 ± 0.004	0.189 ± 0.004	0.089 ± 0.003
I-SLIM	$\mathbf{0.052 \pm 0.001}$	0.039 ± 0.000	0.026 ± 0.000	0.016 ± 0.000	$\mathbf{0.107 \pm 0.003}$	0.134 ± 0.003	0.173 ± 0.004	0.212 ± 0.003	$\mathbf{0.098 \pm 0.003}$
U-LINEAR-FLOW	0.052 ± 0.001	0.039 ± 0.000	0.026 ± 0.000	0.016 ± 0.000	0.106 ± 0.004	0.135 ± 0.003	0.171 ± 0.003	0.212 ± 0.003	0.097 ± 0.003
I-LINEAR-FLOW	0.052 ± 0.001	$\mathbf{0.040 \pm 0.000}$	$\mathbf{0.026 \pm 0.000}$	$\mathbf{0.016 \pm 0.000}$	0.106 ± 0.003	$\mathbf{0.136 \pm 0.003}$	$\mathbf{0.176 \pm 0.003}$	$\mathbf{0.214 \pm 0.003}$	0.097 ± 0.003

Table 5.13: Results on the PROPRIETARY-2 dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	**prec@5**	**prec@10**	**prec@20**	**recall@3**	**recall@5**	**recall@10**	**recall@20**	**mAP@20**
I-KNN	0.087 ± 0.000	0.064 ± 0.000	$0.040 \pm 2.497 \times 10^{-5}$	$0.024 \pm 2.216 \times 10^{-5}$	0.174 ± 0.000	0.210 ± 0.001	0.257 ± 0.001	0.298 ± 0.001	0.159 ± 0.001
U-KNN	0.084 ± 0.001	0.060 ± 0.000	0.037 ± 0.000	$0.021 \pm 7.640 \times 10^{-5}$	0.171 ± 0.001	0.203 ± 0.001	0.241 ± 0.001	0.270 ± 0.001	0.153 ± 0.001
WRMF	0.058 ± 0.001	0.044 ± 0.000	0.028 ± 0.000	$0.017 \pm 8.668 \times 10^{-5}$	0.115 ± 0.001	0.142 ± 0.001	0.180 ± 0.001	0.218 ± 0.001	0.105 ± 0.001
PURESVD	0.033 ± 0.000	0.024 ± 0.000	0.015 ± 0.000	$0.009 \pm 9.786 \times 10^{-5}$	0.069 ± 0.001	0.082 ± 0.000	0.102 ± 0.001	0.123 ± 0.001	0.062 ± 0.000
U-MF-RSVD	0.070 ± 0.000	0.050 ± 0.000	$0.031 \pm 6.921 \times 10^{-5}$	$0.018 \pm 2.443 \times 10^{-5}$	0.141 ± 0.001	0.166 ± 0.001	0.198 ± 0.000	0.228 ± 0.000	0.128 ± 0.001
I-MF-RSVD	0.088 ± 0.000	0.065 ± 0.000	$0.041 \pm 6.596 \times 10^{-5}$	$0.024 \pm 3.470 \times 10^{-5}$	0.172 ± 0.001	0.220 ± 0.001	0.269 ± 0.001	0.320 ± 0.001	0.158 ± 0.001
I-SLIM	$\mathbf{0.089 \pm 0.000}$	$\mathbf{0.067 \pm 0.000}$	$\mathbf{0.043 \pm 9.560 \times 10^{-5}}$	$\mathbf{0.026 \pm 4.358 \times 10^{-5}}$	$\mathbf{0.180 \pm 0.001}$	$\mathbf{0.221 \pm 0.001}$	$\mathbf{0.279 \pm 0.001}$	$\mathbf{0.334 \pm 0.001}$	$\mathbf{0.165 \pm 0.001}$
U-LINEAR-FLOW	0.089 ± 0.000	0.067 ± 0.000	$0.043 \pm 8.660 \times 10^{-5}$	$0.026 \pm 4.089 \times 10^{-5}$	0.176 ± 0.001	0.221 ± 0.001	0.274 ± 0.001	0.329 ± 0.001	0.161 ± 0.001
I-LINEAR-FLOW	0.089 ± 0.000	0.066 ± 0.000	$0.043 \pm 8.660 \times 10^{-5}$	$0.026 \pm 4.089 \times 10^{-5}$	0.178 ± 0.001	0.220 ± 0.001	0.272 ± 0.001	0.326 ± 0.001	0.160 ± 0.001

Runtime evaluation

To compare the training times of the various algorithms, we choose PROPRIETARY-2 and ML1M, the two largest datasets for analysis. We benchmarked the training time of the algorithms by training

Table 5.14: Results on the ML10M dataset. Reported numbers are the mean and standard errors across test folds.

	prec@3	**prec@5**	**prec@10**	**prec@20**	**recall@3**	**recall@5**	**recall@10**	**recall@20**	**mAP@20**
I-KNN	0.175 ± 0.001	0.151 ± 0.001	0.118 ± 0.001	0.088 ± 0.000	0.099 ± 0.000	0.139 ± 0.001	0.212 ± 0.001	0.307 ± 0.001	0.122 ± 0.000
U-KNN	Out of Memory								
WRMF	0.183 ± 0.001	0.156 ± 0.001	0.120 ± 0.000	0.089 ± 0.000	0.102 ± 0.000	0.143 ± 0.001	0.214 ± 0.001	0.306 ± 0.001	0.125 ± 0.000
PURESVD	0.122 ± 0.001	0.105 ± 0.001	0.084 ± 0.001	0.063 ± 0.000	0.073 ± 0.001	0.103 ± 0.001	0.157 ± 0.001	0.226 ± 0.001	0.084 ± 0.001
U-MF-RSVD	0.223 ± 0.001	0.189 ± 0.001	0.146 ± 0.001	0.107 ± 0.000	0.127 ± 0.000	0.176 ± 0.001	0.259 ± 0.001	0.366 ± 0.002	0.159 ± 0.000
I-MF-RSVD	0.223 ± 0.001	0.190 ± 0.001	0.146 ± 0.000	0.103 ± 0.000	0.125 ± 0.001	0.174 ± 0.001	0.260 ± 0.001	0.368 ± 0.002	0.159 ± 0.001
I-SLIM	0.221 ± 0.001	0.189 ± 0.001	0.146 ± 0.000	0.108 ± 0.000	0.126 ± 0.000	0.175 ± 0.001	0.261 ± 0.001	0.369 ± 0.002	0.158 ± 0.001
U-LINEAR-FLOW	**0.227 ± 0.001**	**0.193 ± 0.001**	**0.148 ± 0.001**	**0.108 ± 0.000**	**0.129 ± 0.000**	**0.178 ± 0.001**	**0.262 ± 0.001**	**0.370 ± 0.002**	**0.160 ± 0.001**
I-LINEAR-FLOW	0.224 ± 0.001	0.191 ± 0.001	0.147 ± 0.000	0.108 ± 0.000	0.128 ± 0.001	0.176 ± 0.001	0.261 ± 0.001	0.368 ± 0.002	0.159 ± 0.001

the model on a workstation with 128 GB of main memory and Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz with 32 cores. All of the methods exploit multi-core enabled via numpy linear algebra library, whereas SLIM and WRMF attains parallelism via multiprocessing. For a fair comparison, we ran SLIM and WRMF in parallel to use all available cores. In Table 5.15 we compare the runtime of the proposed method with the baseline methods.

The results demonstrate that while LINEAR-FLOW offers the same quality of recommendation as I-SLIM, its training time is an order of magnitude faster than I-SLIM. Among the baselines, I-SLIM is computationally expensive and is the slowest among the baselines. Neighbourhood methods are computationally cheap as they only involve sparse linear algebra, however as demonstrated previously, their recommendation quality is not consistent and lagging behind the other methods. Further, factorisation approaches that use randomised SVD have similar computational footprints as LINEAR-FLOW while WRMF is much more computationally expensive.

Table 5.15: Training times of various methods on PROPRIETARY-2 and ML10M Dataset.

	PROPRIETARY-2	ML1M
I-KNN	2.5 sec	10.7 sec
U-KNN	46.9 sec	-
PURESVD	3 min	1 min 27 sec
WRMF	27 min 3 sec	12 min 38 sec
U-MF-SVD	3 min 10 sec	1 min 38 sec
I-MF-SVD	3 min 8 sec	1 min 39 sec
I-SLIM	32 min 37 sec	7 min 40 sec
U-LINEAR-FLOW	3 min 27 sec	1 min 44 sec
I-LINEAR-FLOW	3 min 32 sec	1 min 42 sec

Qualatitive analysis of learned similarities

In this Section, we provide a qualitative evaluation of the similarities, in particular item-item, learned by our I-LINEAR-FLOW model. We use PROPRIETARY-3 , a dataset from a major stock image market site[13]. The data provides whether a given user has clicked on a particular image category, and from these our model can infer the similarity measure between the image categories. We choose this dataset for the qualitative evaluation mainly because the category names are much easier to interpret compared to the other datasets.

In Table 5.16, we show some examples[14] of top-5 similar items learned by I-LINEAR-FLOW model. We observe that the model discovers meaningful and explainable similarities, hence making it applicable in similar item recommendations.

[13]The dataset sharing agreement with the provider restricts us from reporting the statistics and quantitative results. Hence, we do not report summary statistics and quantitative results on this dataset

[14]Visit http://ssedhain.com/demos/Item-Item.html for interactive visualization

Table 5.16: Top-5 similar items learned by I-LINEAR-FLOW model.

Item	Chemistry	Chilling out	Workers	Unemployment	Divorce and Conflict	Museums
	Test and Analysis	Beach Holidays	Construction	Job Search	Depression	Painting
	Drug and Pills	Tourism	Teamwork	Tax and Accounting	Getting upset	Statues
Similar items	Health Care	Relaxing	Manufacturing	Breaking the law	Crying	Artistic monuments
	Scientists	Hiking	Service industry	Money	Loneliness	Paris
	Medical Equipments	Consumer service	Beaches	Workers	Rage	Italy
Item	Dance	Vinegar	Pearls	Graduation	Aging	Homelessness
	Exercise	Olive Oil	Wealth	High School	Patients	Depression
	Running And Jumping	Spice	Wedding	School	Grand Parenting	Loneliness
Similar items	Disco And Clubs	Salads	Accessories	Exams	Disability	Crying
	Circus And Performing	Garlic	Gold	Job Search	Health Care	Getting Upset
	Gymnastics	other	Make Up	E-Learning	Doctors	Risk And Danger

5.9 Conclusion

In this Chapter, we formulated LREC, a user-focused linear model for OC-CF. We demonstrated the superior performance of LREC over the state-of-the-art baselines. Despite being embarrassingly parallel(parallelisable without distributed communication), LREC, like other linear models, suffers from computational and space limitations. To address the limitations we formulated LINEAR-FLOW, a fast low-dimensional regularised linear model, which levearges randomised SVD algorithm. We showed that LINEAR-FLOW is computationally superior to the state-of-the-art models and yields competitive performance.

So far, we discussed linear models for various recommendation tasks. However, as an inherent limitation, linear models are not capable of capturing complex nonlinear relationship which might be predictive of users' preferences. In the next chapter, we investigate a general neural architecture for the recommendation building upon the ideas from LINEAR-FLOW.

Chapter 6

Beyond Linear Models: Neural Architecture for Collaborative Filtering

Up to this point in the dissertation, we have only discussed linear models for various recommendation tasks. Despite the superior performance of linear models, such models are not capable of capturing complex nonlinear relationships that may be predictive of users' preferences. Recently, nonlinear methods, in particular, deep neural networks have gained a lot of attention for various machine tasks. Deep learning has revolutionised many areas of machine learning, namely computer vision [Krizhevsky et al., 2012], natural language processing [Mikolov et al., 2013] and speech recognition [Hinton et al., 2012] . However, there has been very limited research in the application of deep learning for CF.

In this Chapter, we take a departure from linear models and investigate deep learning architectures for CF. Building upon the ideas from LINEAR-FLOW and LoCo model, we identify CF as an auto-encoding problem and propose a general nonlinear neural network CF architecture AUTOREC, bridging the core of the thesis, Linear CF models, to the deep learning literature. We show significant gain in performance using these models and thus opening the door for exploration of deep learning for collaborative filtering.

6.1 Problem setting

In this Chapter, we take a departure from the previous chapters by investigating nonlinear models for CF. Further, instead of focusing on a particular CF scenario, we focus on formulating a neural network architecture for CF that generalises to the models discussed in previous Chapters. For experimentation, we choose rating prediction setting where we have partially observed user-item historical rating data $\mathbf{R} \in \mathbb{R}^{m \times n}$. Based on the observed rating data, the goal of a recommendation algorithm is to predict ratings for the unobserved user-item pair.

6.2 Background : Neural network architectures

In this Section, we will discuss relevant neural network architectures in the context of CF.

6.2.1 Restricted Boltzmann Machine (RBM)

Restricted Boltzmann Machine (RBM) is a two layered generative neural architecture consisting of a hidden and visible layer, where each layer is composed of a number of nodes known as units. The RBM is an energy-based model and is generally used with the binary inputs. Let $\mathbf{r} \in \mathbb{R}^d$ denote the input units and $\mathbf{h} \in \mathbb{R}^k$ denote the hidden units, and $\mathbf{W} \in \mathbb{R}^{d \times k}$ be the weights between them, then the energy is

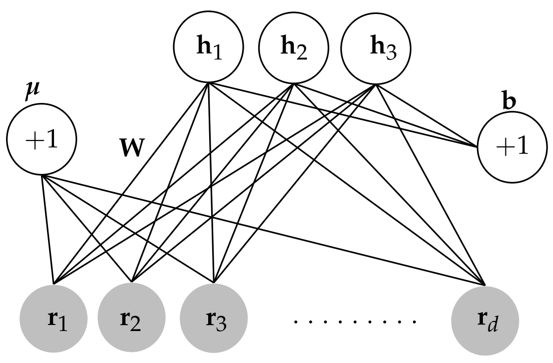


Figure 6.1: Restricted Boltzmann Machine.

defined as:

$$E(\mathbf{r},\mathbf{h}) = -\sum_i \mu_i \mathbf{r}_i - \sum_j \mathbf{b}_j \mathbf{h}_j - \sum_{i,j} \mathbf{r}_i \mathbf{h}_j \mathbf{W}_{ij}$$

where $\mathbf{W}_{ij}$ is weight of connection between $\mathbf{r}_i$ and $\mathbf{h}_j$; μ_i and $\mathbf{b}_j$ are bias for visible and hidden units respectively. The joint and marginal distribution of the hidden and visible unit is given by

$$p(\mathbf{r},\mathbf{h}) = \frac{1}{\mathbf{Z}} \exp(-E(\mathbf{r},\mathbf{h}))$$
$$p(\mathbf{r}) = \frac{1}{\mathbf{Z}} \sum_{\mathbf{h}} \exp(-E(\mathbf{r},\mathbf{h}))$$

where the partition function, $\mathbf{Z}$, is defined as

$$\mathbf{Z} = \sum_{\mathbf{r},\mathbf{h}} \exp(-\mathbf{E}(\mathbf{r},\mathbf{h}))$$

Given the visible layer, the hidden layer is defined as:

$$\mathbf{p}(\mathbf{h_j} = \mathbf{1}|\mathbf{r}) = \mathbf{g}(\mathbf{b_j} + \sum_{\mathbf{i}} \mathbf{r_i W_{ij}}),$$

where g is some activation function, such as a sigmoid. Simlarly, given the hidden layer, the visible layer is obtained as:

$$\mathbf{p}(\mathbf{r_i} = \mathbf{1}|\mathbf{h}) = \mathbf{f}(\mu_{\mathbf{i}} + \sum_{\mathbf{j}} \mathbf{h_j W_{ij}})$$

where f is some activation function or normalisation function.

The parameter update rule for log-likelihood of the RBM model is defined as:

$$\delta \mathbf{W}_{ij} = \frac{\partial \log\ p(\mathbf{r})}{\partial \mathbf{W}_{ij}} = \left(\langle \mathbf{r}_i, \mathbf{h}_j \rangle_{data} - \langle \mathbf{r}_i, \mathbf{h}_j \rangle_{model} \right),$$

where $\langle \mathbf{r}_i, \mathbf{h}_j \rangle_{data}$ denotes the frequency with which visible unit i and hidden unit j are on together when the model is driven by observed data. On the other hand, $\langle \mathbf{r}_i, \mathbf{h}_j \rangle_{model}$ is the expectation with respect to the distribution defined by the model. Unlike the expectation w.r.t. to the data which is easy to compute, the expectation w.r.t. the model requires alternate Gibbs sampling for a very long period of time [Hinton, 2012]. Hence, we typically approximate the expectation using a fast MCMC technique [Hinton, 2002, Tieleman, 2008] known as *Contrastive Divergence*.

6.2.2 Autoencoders

Autoencoders [Bourlard and Kamp, 1988, Hinton and Salakhutdinov, 2006] are the widely used neural architectures that consists of three layers, namely *input*, *hidden*, and *output* layers. Given a set of vectors in $\{\mathbf{r}_i\} \in \mathbb{R}^d$, an autoencoder minimises the reconstruction error by optimiising

$$\min_{\theta} \sum_{i} \ell(\mathbf{r}, \mathcal{R}(\mathbf{r}_i; \theta)) + \Omega(\theta), \tag{6.1}$$

where $\mathcal{R}(\mathbf{r};\theta)$ is the *reconstruction* of input $\mathbf{r}$, $\mathcal{R}(\mathbf{r};\theta) = f\left(g(\mathbf{r}\mathbf{V} + \boldsymbol{\mu}) \cdot \mathbf{U} + \mathbf{b}\right)$ for *activation functions* $f(\cdot), g(\cdot)$. Here, $\theta = \{\mathbf{U}, \mathbf{V}, \boldsymbol{\mu}, b\}$ for transformations $\mathbf{U} \in \mathbb{R}^{k\times d}, \mathbf{V} \in \mathbb{R}^{d\times k}$, and biases $\boldsymbol{\mu} \in \mathbb{R}^k, \mathbf{b} \in \mathbb{R}^d$. This objective corresponds to an auto-associative neural network with a single, k-dimensional hidden layer. The parameter θ are learned using backpropagation [Rumelhart et al., 1988].

For squared loss and ℓ_2 regularisation, we have:

$$\min_{\theta} \sum_{i} ||\mathbf{r}^{(i)} - \mathcal{R}(\mathbf{r}^{(i)}; \mathbf{U}, \mathbf{V}))||_2^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2), \tag{6.2}$$

For, identity activation function $f(\cdot), g(\cdot)$, autoencoder corresponds to Principal Component Analysis (PCA) [Baldi and Hornik, 1989].

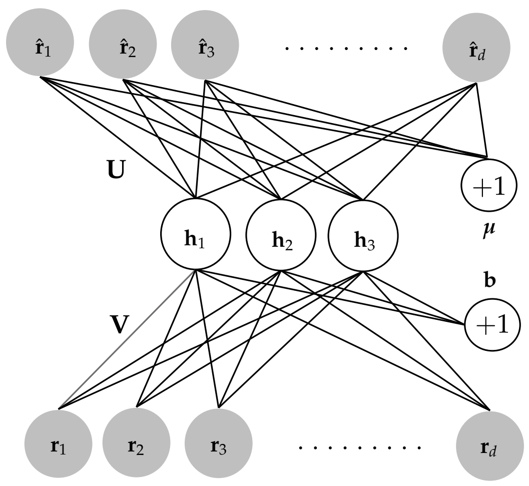


Figure 6.2: Autoencoder model.

6.2.3 RBM for collaborative filtering (RBM-CF)

RBM based collaborative filtering (RBM-CF) [Salakhutdinov et al., 2007] treats each user as an input to the model. In RBM-CF, each user $u \in U = \{1 \ldots m\}$ is represented by a partially observed vector $\mathbf{R}_{u:} = (R_{u1}, \ldots R_{un})$. Since the input is partially observed, standard RBM training, as discussed in 6.2.1, cannot be applied. Further, for rating prediction, we cannot impute 0 for unobserved entries as the model will learn to predict ~ 0, which is not desirable. RBM-CF incorporates partially observed input by following an alternative training strategy where for each user u, the weights associated with the observed entries are only updated as illustrated in Figure 6.4.

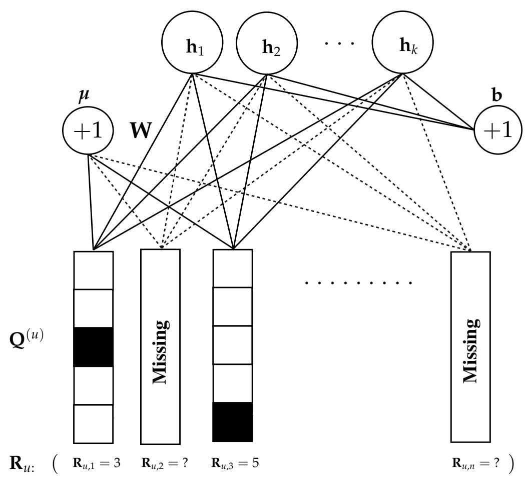

$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_k$
μ
+1
W
b
+1
$\mathbf{Q}^{(u)}$
Missing
Missing
$\mathbf{R}_{u:}$ ($\mathbf{R}_{u,1}=3$ $\mathbf{R}_{u,2}=?$ $\mathbf{R}_{u,3}=5$ $\mathbf{R}_{u,n}=?$)

Figure 6.3: RBM based collaborative filtering model. The solid weights indicate the weights that are updated while training for the input $R_{u:}$.

Let us consider integer [1] rating matrix $\mathbf{R} \in \{1,...L\}^{m\times n}$. First, to incorporate non-binary ratings, RBM-CF maps $\mathbf{R}_{u:}$ to a binary indicator matrix, $\mathbf{Q}^{(u)} \in \{0,1\}^{L\times n}$, where

$$\mathbf{Q}^{(u)}_{li} = \begin{cases} 1 & R_{ui} = l \\ 0 & \text{otherwise.} \end{cases}$$

RBM-CF uses sigmoid activation for hidden units and softmax for the input. The conditional probablity of a hidden unit is given as:

$$p(\mathbf{h}_j = 1|\mathbf{Q}^{(u)}) = \sigma(\mathbf{b}_j + \sum_l^L \sum_i^n \mathbf{Q}^{(u)}_{li} \mathbf{W}^l_{ij}).$$

Similarly, the conditional probability of a visible unit is given as:

$$p(\mathbf{Q}^{(u)}_{li} = 1|\mathbf{h}) = \frac{\exp(\mu^l_i + \sum_j^k \mathbf{W}^l_{ij}\mathbf{h}_j)}{\sum_l^L \exp(\mu^l_i + \sum_j^k \mathbf{W}^l_{ij}\mathbf{h}_j)}$$

The optimisation method is the same as discussed in 6.2.1 where only the weights associated with the observed entries are updated for each training instance. Once the model is trained, the rating R_{ui} can be computed as

$$R_{ui} = \sum_{l=1}^{L} p(\mathbf{Q}^{(u)}_{li} = 1|\mathbf{h}) \cdot l$$

We can define two class of RBM-CF model based on the input, namely U-RBM-CF and I-RBM-CF. While U-RBM-CF takes $R_{u:}$ as input, I-RBM-CF takes $R_{i:}$ as input. In this Chapter, we will show that the choice of model can significantly affect the performance.

There are several limitations of the RBM-CF model. First, RBM-CF proposes a generative, probabilistic method that estimates parameters by maximising the log likelihood, instead of root mean square error (RMSE) which is a canonical measure for rating prediction problem. Second, it estimates parame-

[1]Real valued rating can also be used via binning.

ters using slow contrastive divergence optimisation. Third, it is only applicable for discrete ratings, and requires binning for real valued inputs. For l possible ratings, this implies nkr or (mkr) parameters for user- (item-) based RBM.

6.3 AutoRec: Autoencoders meet collaborative filtering

In this section, we introduce AUTOREC, an autoencoder based model for CF. As in RBM-CF model, we can define user and item based models. Here, we discuss a user based AUTOREC model[2]. The user based AUTOREC model optimises Equation 6.3 for a set of user vectors, $\mathbf{R}_{u:}$, with one important change. We account for the fact that $\mathbf{R}_{u:}$ is partially observed by only updating during backpropagation those weights that are associated with observed inputs, as in RBM-CF. Formally, the objective function for the User-based AUTOREC (U-AUTOREC), is defined as:

$$\min_{\mathbf{U},\mathbf{V}} \sum_i ||\mathbf{R}_{u:} - \mathcal{R}(\mathbf{R}_{u:};\mathbf{U},\mathbf{V})||_{\mathcal{O}}^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2), \tag{6.3}$$

where $||\cdot||_{\mathcal{O}}^2$ means that we only consider the contribution of observed ratings; $\mathcal{R}(\mathbf{R};\theta)$ is the *reconstruction* of input $\mathbf{R}_{u:}$,

$$\mathcal{R}(\mathbf{r};\theta) = g\left(f(\mathbf{r}\mathbf{V} + \boldsymbol{\mu}) \cdot \mathbf{U} + \mathbf{b}\right)$$

for *activation functions* $f(\cdot), g(\cdot)$. Further, item-based AUTOREC (I-AUTOREC) is defined in a similar way with $\mathbf{R}_{:i}$ as input. In total, U-AUTOREC requires estimation of $(2nk + n + k)$, I-AUTOREC requires estimation of $(2mk + m + k)$ parameters.

We optimise AUTOREC parameters using RPROP [Riedmiller and Braun, 1992] algorithm, which updates the neural network parameters, $\mathbf{W}$, using following update rule:

$$\mathbf{W}_{ij}^{t+1} = \mathbf{W}_{ij}^{(t)} - sgn\left(\frac{\partial}{\partial \mathbf{W}_{ij}^{(t)}}\right)\Delta_{ij}^t,$$

where Δ_{ij}^t is defined as:

$$\Delta_{ij}^t = \begin{cases} \eta^+ \cdot \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial \ell}{\partial \mathbf{W}_{ij}^{(t-1)}} \cdot \frac{\partial}{\partial \mathbf{W}_{ij}^{(t)}} > 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial \ell}{\partial \mathbf{W}_{ij}^{(t-1)}} \cdot \frac{\partial}{\partial \mathbf{W}_{ij}^{(t)}} < 0 \\ 0 & \text{otherwise} \end{cases}$$

where $0 < \eta^- < 1 < \eta^+$. In a nutshell, RPROP adjust the weights update by checking whether the gradient at t and $t-1$ are in the same direction or not , which approximates the curvature of the loss function at given time t.

AUTOREC is distinct to existing CF approaches. Compared to the RBM-based CF model (RBM-CF), there are several differences. First, RBM-CF proposes a generative, probabilistic model based on restricted boltzmann machines, while AUTOREC is a discriminative model based on autoencoders. Second, RBM-CF estimates parameters by maximising log likelihood, while AUTOREC directly minimises RMSE, the canonical performance in rating prediction tasks. Third, training RBM-CF requires the use

[2]Instead of autoencoding $R_{u:}$ as in user based method, item based method autoencodes $R_{:i}$

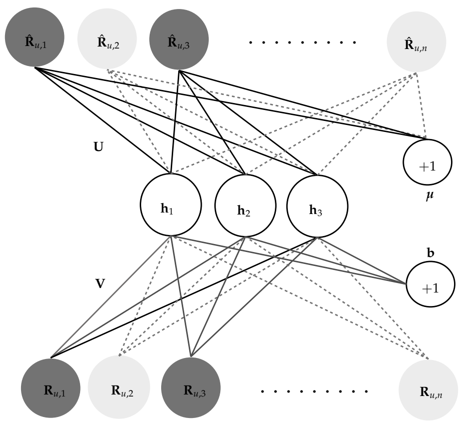


Figure 6.4: AUTOREC model. The dark nodes in the input layer correspond to the observed ratings.

of contrastive divergence, whereas training AUTOREC requires the comparatively faster gradient-based backpropagation. Finally, RBM-CF is only applicable for discrete ratings, and estimates a separate set of parameters for each rating value. For r possible ratings, this implies nkr or (mkr) parameters for user-(item-) based RBM. AUTOREC is agnostic to r and hence requires fewer parameters. Fewer parameters enables AUTOREC to have less memory footprint and makes it less prone to overfitting.

Compared to matrix factorisation (MF) approaches, which embed both users and items into a shared latent space, the item-based AUTOREC model only embeds items into latent space. Further, while MF learns a linear latent representation, AUTOREC can learn a *nonlinear* latent representation through activation function $g(\cdot)$.

To our knowledge, ours is the first contribution to introduce autoencoders in the context of collaborative filtering. However, there are recent independent works in applying autoencoders for collaborative filtering [Wang et al., 2015, Wu et al., 2016, Li et al., 2015].

6.4 Relation to existing models

In this section, we discuss the relation of AUTOREC to existing models. First, for the convenience of analysis, we write AUTOREC, with linear activations, as a generalised neural network equation defined in terms of input and output

$$\min_{\mathbf{U},\mathbf{V}} \sum_i ||\mathbf{R} - \mathbf{MVU}||^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2). \tag{6.4}$$

where $\mathbf{R}$ is a target output (preference matrix), and $\mathbf{M}$ is an input matrix. When $\mathbf{M} = \mathbf{R}$, this corresponds to standard AUTOREC equation i.e.

$$\min_{\mathbf{U},\mathbf{V}} \sum_i ||\mathbf{R} - f(g(\mathbf{RV})\mathbf{U})||^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2). \tag{6.5}$$

In Figure 6.5, we outline a generalised neural network architecture that defines models for various CF scenarios, as we will show shortly.

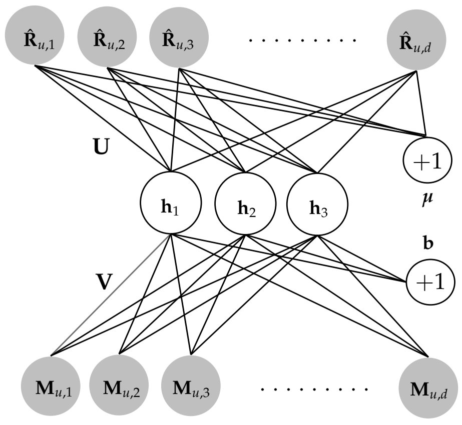


Figure 6.5: Generalised neural architecture for CF. For $\mathbf{M} = \mathbf{R}$, it is equivalent to AUTOREC model; For $\mathbf{M} = \mathbf{X}$, it is equivalent to LoCo model.

6.4.1 Relation to matrix factorisation

In this section, we define MF models in terms of generalised neural architecture. First, we define users' one hot encoding matrix $\mathbf{I} \in \mathbb{R}^{m \times m}$. In other words, $\mathbf{I}$ is an identify matrix. For an input $\mathbf{I}$, we have

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{V}} \sum_u ||\mathbf{R}_{u:} - \mathbf{I}_{u:}\mathbf{V}\mathbf{U}||^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2) \\ =&\min_{\mathbf{U},\mathbf{V}} \sum_u ||\mathbf{R}_{u:} - \mathbf{V}_{u:}\mathbf{U}||^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2) \\ =&\min_{\mathbf{U},\mathbf{V}} \sum_{u,i} (\mathbf{R}_{ui} - \mathbf{V}_{u:}\mathbf{U}_{:i})^2 + \frac{\lambda}{2} \cdot (||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2), \end{aligned} \tag{6.6}$$

where $\mathbf{V} \in \mathbb{R}^{m \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times n}$. In Equation 6.6, we see matrix factorisation model is equivalent to AUTOREC model with indicator matrix $\mathbf{N}$ as an input.

In Table 6.1, we summarise how various models can be expressed in the proposed architecture. In general, AUTOREC is more powerful compared to MF as it allows nonlinearity via activation functions. Further, it allows building complex models by constructing deep architectures.

Input (**M**)	**Target** (**R**)	**Model**
I	**R**	MF
R	**R**	AUTOREC
X	**R**	LoCo

Table 6.1: Expressing various CF model in a generalised neural architecture. Here, **I** is an identity matrix, **R** is a user-item preference matrix and **X** is a feature matrix.

6.4.2 Relation to LRec and LINEAR-FLOW

AUTOREC is closely related to LRec. To see the connection, we rewrite LRec with squared loss as

$$\operatorname*{argmin}_{\mathbf{W}} ||\mathbf{R} - \mathbf{R}\mathbf{W}||_F^2 + \lambda\, ||\mathbf{W}||_F^2 \tag{6.7}$$

Comparing Equation 6.7 and Equation 6.5, we see AUTOREC is equivalent to LRec with a factorised parameters i.e. $\mathbf{W} \sim \mathbf{UV}$. Similarly, AutoRec is also closely related with Linear-FLow model.To see the connection, we rewrite Linear-FLow, as discussed in section 5.6:

$$\underset{\mathbf{Z}}{\operatorname{argmin}} \, ||\mathbf{R} - \mathbf{R}\mathbf{Q}_k\mathbf{Z}||_F^2 + \lambda \, ||\mathbf{Z}||_F^2, \tag{6.8}$$

where $\mathbf{R} \approx \mathbf{P}_k \Sigma \mathbf{Q}_k^T$ is given by SVD. Now, comparing Equation 6.5 with Equation 6.8, we see Linear-FLow is equivalent to AutoRec model where $\mathbf{V}$ is kept fixed as $\mathbf{Q}_k$ and $\mathbf{U}$ is learned minimising squared error via multi-regression.

The main advantage of AutoRec over LRec and Linear-FLow is that it allows nonlinear modeling via the choice of activation function, and hence is more flexible in capturing nonlinear relationship that may exist in the data.

6.4.3 Relation to LoCo

To compare AutoRec model with LoCo, we rewrite the LoCo objective as (discussed in Section 4.3.2):

$$\min_{\mathbf{Z}} ||\mathbf{R}^{(\mathrm{tr})} - \mathbf{X}^{(\mathrm{tr})}\mathbf{Q}_k\mathbf{Z}||_F^2 + \lambda ||\mathbf{Z}||_F^2 \tag{6.9}$$

where $\mathbf{X}^{(\mathrm{tr})} \sim \mathbf{P}_k \sum_k \mathbf{Q}_k^T$ is given by SVD. While AutoRec learns to auto-associate the preferences matrix $\mathbf{R}$, LoCo learns to map user or item features, $\mathbf{X}$, to the preference matrix $\mathbf{R}$. Equation 6.4 corresponds to LoCo when the input to the model is a feature matrix i.e $\mathbf{M} = \mathbf{X}$, .

6.5 AutoRec for OC-CF

The key challenge in applying AutoRec model for OC-CF setting is in scalability. In OC-CF setting, we assume all unobserved entries as 0 for the obvious reason as discussed in Chapter 2. Hence, we need to compute the gradients for all entries of $\mathbf{R}$ which makes the learning computationally expensive. We can employ negative sub-sampling to address this limitation. However, it is non-trivial to come up with a sub-sampling strategy that yields competitive results.

6.6 Experiments and evalution

In this section, we discuss the dataset used for experiments, experimentation methodology, and report detailed experimental results.

6.6.1 Data description

In this Chapter, we evaluate the performance of the models for rating prediction task on three real-world datasets.

- ML1M and ML10M are the standard datasets[3] for CF. ML1M consists of $\sim 6,000$ users, $\sim 3,700$ items and ~ 1 million ratings. Similarly, ML10M consists of $\sim 70,000$ users, $\sim 10,000$ items and ~ 10 million ratings.

[3]http://grouplens.org/datasets/movielens/

- NETFLIX is the popular large scale dataset from the Netflix prize competition Bennett et al. [2007b]. NETFLIX dataset consists of $\sim$ 480,000 users, $\sim$ 17,000 items and $\sim$ 100 million ratings.

We summarise the datasets in the Table 6.2.

Table 6.2: Summary of datasets used in evaluation.

Dataset	**m**	**n**	**ratings**
ML1M	6,040	3,706	1,000,209
ML10M	69,878	10,677	10,000,054
NETFLIX	480,189	17,770	100,480,507

We split the data into random 90% − 10% train-test sets, and hold out 10% of the training set for hyperparameter tuning. We repeat this splitting procedure 5 times and report average RMSE. In each experiment, 95% confidence intervals on RMSE were ± 0.003 or less . For all baselines, we tuned the regularisation strength $\lambda \in \{0.001, 0.01, 0.1, 1, 100, 1000\}$ and the appropriate latent dimension $k \in \{10, 20, 40, 80, 100, 200, 300, 400, 500\}$.

6.6.2 Model comparison

In this Section, we discuss various baselines we used to compare with AUTOREC model, I-AUTOREC and U-AUTOREC:

- BIASEDMF of equation 2.7, as discussed in Section 2.4.1.
- I-RBM-CF and U-RBM-CF as discussed in Section 6.2.3.
- LLORMA as discussed in Section 2.4.1. Essentially, LLORMA weights 50 different local matrix factorisation models.

6.6.3 Results and analysis

Evaluation of User and Item based RBM-CF and AUTOREC models

In Table 6.3, we compare user and item based RBM-CF andAUTOREC models. We observe that item based models perform significantly better than user based models. This is likely since the average number of ratings per item is much more than per user. The high variance in the number of ratings per user leads to less reliable prediction for user-based models. Further, I-AUTOREC yields the best results.

Table 6.3: Comparison of the RMSE of AUTOREC and RBM-CF models.

	ML-1M	ML-10M
U-RBM	0.881	0.823
I-RBM	0.854	0.825
U-AutoRec	0.874	0.867
I-AutoRec	**0.831**	**0.782**

Does performance of AutoRec vary with the choice of optimisation algorithm?

In Table 6.4, we compare the performance of I-AutoRec on Ml1M dataset for four different optimisation methods, namely (a) Stochastic Gradient Descent (SGD), (b) Batch Gradient Descent (BGD), (c) RProp, (d) LBFGS. We see that the performance of AutoRec varies significantly with the choice of the optimization technique. Methods that consider local curvature (LBFGS and RProp) gave the best performance and faster convergence, while gradient based first order methods(SGD and Batch gradient descent) performed poorly. This may be due to pathological curvature in the objective function. Hence, it highlights the importance of choosing a right optimization technique while training neural network models. Even though LBFGS yields better results than RProp, we choose RProp for rest of our experiments due to scalability issue.

	RMSE
SGD	0.8720
BGD	0.8586
RProp	0.8305
LBFGS	0.8284

Table 6.4: RMSE of the item based AutoRec model on the Ml1Mdataset for various optimization methods.

Variation of AutoRec performance with the choice of activation functions $f(\cdot)$, $g(\cdot)$

In Table 6.5, we compare the performance of I-AutoRec for various activation functions in hidden and output layers. We observe that the nonlinearity in hidden layer (via $g(\cdot)$) is critical for good performance of the models. This indicates AutoRec's potential advantage over bilinear *MF* methods. Further, we experimented with Rectifier Linear Units(ReLU), however they performed worse than sigmoid activation. In the following experiments, we use identify $f(\cdot)$ and sigmoid $g(\cdot)$ functions.

Table 6.5: RMSE for I-AutoRec model with choices of linear and nonlinear activation functions on Ml1M dataset.

$f(\cdot)$	$g(\cdot)$	RMSE
Identity	Identity	0.872
Sigmoid	Identity	0.852
Identity	Sigmoid	**0.831**
Sigmoid	Sigmoid	0.836

Variation of performance of AutoRec with the number of hidden units

In Figure 6.6, we evaluate the perfomrance of AutoRec model as the number of hidden units varies. We note that performance steadily increases with the number of hidden units, but with diminishing returns. In the following experiments, we use $k = 500$ for AutoRec.

Comparison of AutoRec with the baselines

In Table 6.6, we compare the performance of AutoRec with all baselines. We observe AutoRec consistently outperforms all baselines, except for comparable results with LLORMA on Ml10M dataset.

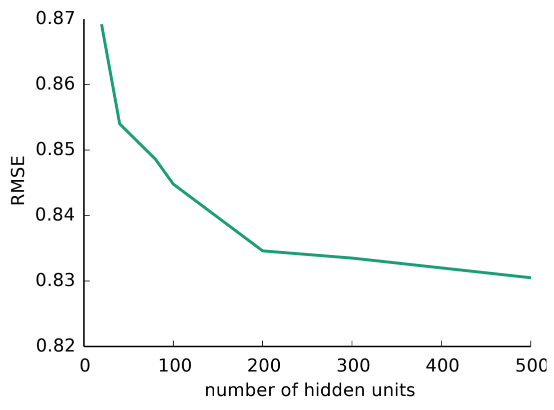


Figure 6.6: RMSE of I-AutoRec on Ml1M as the number of hidden units k varies.

Competitive performance with LLORMA is significant, as the latter involves *weighting 50 different local matrix factorisation models*, whereas AutoRec only uses a single latent representation via a neural net autoencoder.

Table 6.6: Comparison of I-AutoRec with baselines on MovieLens and Netflix datasets. We remark that I-RBM-CF did not converge after one week of training. LLORMA's performance is taken from Lee et al. [2013].

	ML-1M	ML-10M	Netflix
BiasedMF	0.845	0.803	0.844
I-RBM	0.854	0.825	-
U-RBM	0.881	0.823	0.845
LLORMA	0.833	**0.782**	0.834
I-AutoRec	**0.831**	**0.782**	**0.823**

Deep AutoRec : Do deep extensions help?

We developed a deep version of I-AutoRec with three hidden layers of (500, 250, 500) units, each with a sigmoid activation. We used greedy pre-training and then fine-tuned using RProp algorithm. On Ml1M, RMSE reduces from 0.831 to 0.827 indicating potential for further improvement via deep AutoRec. This indicates the further potential of deep architectures for collaborative filtering problems. There has been some work in investigating deep learning models for collaborative filtering [Wang et al., 2015, Wu et al., 2016, van den Oord et al., 2013]. However, most of the success of deep learning methods is in dense input data. Hence, it is still an open question to leverage the full potential of deep learning techniques on sparse input which requires further careful investigation.

6.7 Conclusion

In this Chapter, we formulated AutoRec, a neural architecture that generalises to various CF methods contributed in earlier Chapters. Further, we demonstrated that the proposed model yields substantial improvement over the state-of-art models for rating prediction task.

Despite its superior performance, one of the key limitations of the AutoRec model is that, for OC-CF where the unobserved preferences are treated as negative feedback, learning is inefficient. In particular, unlike MF that leverages efficient closed-form solution via Alternate Least Squares(ALS) optimisation, AUTOREC's parameters are learned jointly using gradient based optimisation. So, it requires further investigation for efficient optimisation and effective negative sampling techniques. Also, it is unclear that how much we gain in performance by adding layers. Further, we need to investigate emerging deep learning architectures, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), for their potential in CF problems.

Chapter 7

Conclusion

7.1 Summary of Contributions

In this dissertation, we have investigated linear models for different collaborative filtering scenarios that address all collaborative filtering desiderata, namely being (1) applicable to wide range of recommendation scenarios, (2) learning-based, (3) amenable to convex optimisation, and (4) scalable. We summarise our contributions as follow:

- In Chapter 3, we investigated the Social-CF problem. We addressed the limitations of existing Social-CF algorithms by formulating a linear model, *Social Affinity Filtering*, that leverages fine-grained user interactions and activities in a social network. We presented a thorough experimental evaluation to demonstrate the superior performance of proposed method and the predictive power of fine-grained social signals.

- In Chapter 4, we investigated linear models for a cold-start recommendation in the context of OC-CF settings. Building upon the insights from Chapter 3, we proposed two large-scale linear models, namely (a) Generalised neighborhood-based model, and (2) LoCo, which leverages high dimensional social information for cold-start recommendation. We demonstrated the superior performance of proposed methods in thorough experiments.

- In Chapter 5, we investigated linear models for OC-CF. We proposed LRec, a user-focused linear CF model that generates user-personalised recommendation by training a convex unconstrained objective. Despite being embarrassingly parallelizable across the users, LRec requires solving a large number of regression subproblems and requires quadratic memory limiting its applicability to large-scale problems. To address these limitations, we proposed low dimensional regression model LINEAR-FLOW which leverages randomised SVD for fast dimensionality reduction. We presented detailed experimental results to demonstrate the efficacy of LRec and LINEAR-FLOW on a wide range of real-world datasets.

- In Chapter 6, we took a departure from linear models by investigating deep learning architectures for collaborative filtering. We proposed a generalised autoencoder based architecture, AUTOREC, which yields substantial improvement over the state-of-the-art models for rating prediction problem. Further, we discussed how various CF models can be expressed as special cases of the AUTOREC framework elucidating the flexibility of AUTOREC.

In Table 7.1, we concisely summarise the contribution of the thesis.

	Model	Recommendation	Model parameters
Linear	SAF	$\mathbf{X}diag(\mathbf{w})\mathbf{X}^T\mathbf{R}$	$\mathbf{w}$
	LoCo	$\mathbf{X}^{(\text{te})}\mathbf{V}_k\mathbf{Z}$	$\mathbf{Z}$
	LRec	$\mathbf{WR}$	$\mathbf{W}$
	Linear-FLow	$\mathbf{ZP}_k^T\mathbf{R}$	$\mathbf{Z}$
Non Linear	AutoRec	$f(\mathbf{U}\cdot g(\mathbf{VR}))$	$\mathbf{U}, \mathbf{V}$

Table 7.1: Summary of the proposed models.

7.2 Future work

While this thesis addresses the wide range of collaborative filtering scenarios in a unified linear framework, there are many unexplored areas for future research as we discuss next.

7.2.1 Exploration of emerging deep learning architectures

In Chapter 6, we demonstrated the efficacy of AutoRec model for CF. However, in the recent years, advance neural architectures, such as *Recurrent Neural Network (RNN)* [Goller and Kchler, 1996] and *Convolutional Neural Network (CNN)* [Krizhevsky et al., 2012, Simard et al., 2003], have been very successful for various machine learning tasks. In particular, RNNs are popular in modeling sequential data. Typically, in many CF problems, users' actions such as items purchased, clicks, songs listened, etc. can be treated as a sequence of events. In such a setting, RNN's can be employed to learn users behaviour. Further, RNN's can incorporate contextual information by using recently proposed attention models [Xu et al., 2015]. Similarly, learning meaningful representation of users and items, for instance using CNN to learn features from the image of the items, can yield better recommendation algorithms. Such features can be used in content-based as well as CF models.

7.2.2 Incorporating temporal information

Temporal patterns are very important in personalisation [Koren, 2010a, Lathia et al., 2009]. For instance, an early teen Netflix user might not be interested in the same genre of a movie she used to watch as a kid; a user might not be interested in summer clothing in winter. Despite it's relevance in personalisation, there has been very limited work in leveraging temporal signals in collaborative filtering. One of the limitations of the existing approaches is that there is no principle objective and model specification for which learning temporal patterns is tractable. Since users' behaviour and preferences drift over time, it is highly desirable to have a model that automatically adapts with the shift of users preferences over time.

7.2.3 Ensemble methods for OC-CF

Ensemble methods involve combining the predictions of multiple models. Such models are widely used in practice due to their superior performance [Sollich and Krogh, 1996, Adeva et al., 2005]. Further, they have low model variance making it ideal for real world problems. In Netflix competition, the winning model was an ensemble of hundreds of different CF models [Koren, 2009].

One of the problems with OC-CF is in combining various CF models. While rating prediction models optimise the objective function that directly corresponds to evaluation metrics (such as RMSE), most OC-CF methods don't optimise ranking objectives due to computational reasons. Hence, it is non-trivial to combine the recommendations from various OC-CF models. Further, rank aggregation methods [Dwork et al., 2001] can be applied to merge the ranked list from multiple methods. Despite being practically pervasive, there has been a very limited research in this area and it is not clear as to how to combine different OC-CF models in an effective way. Hence, this requires further investigation.

7.2.4 Models for location aware recommendation

Due to the ubiquity of mobile devices, location has become one of the important aspects of personalization. For instance, a shopper might be interested in offers from nearby shops; a person would be more interested in nearby restaurants. Despite the abundance of location data, there has been very limited work in leveraging such signals in the context of recommender systems [Levandoski et al., 2012, Bao et al., 2012, Ye et al., 2010, Wang et al., 2013]. It is not clear as how to incorporate location data into existing collaborative filtering models in a meaningful way. Further, the trade-off between preferences and distance itself requires personalisation. Hence, this requires further investigation.

7.3 Conclusion

This thesis demonstrates the efficacy of linear models for various collaborative filtering scenarios. Our main contribution is in the formulation of a unified linear model for collaborative filtering and bridging the core of the thesis to deep learning literature. Our experimental results showed that our models outperform existing methods yielding the state-of-the-art performance. We hope that the presented work encourages further investigation of linear models for recommendation and engender future research in the application of deep learning architectures for collaborative filtering.

Appendix A

Randomised SVD

SVD is the archetypal matrix factorisation algorithm and has been widely used in machine learning for dimensionality reduction. However, SVD is computationally expensive and not scalable to large scale datasets. It has been recently shown that SVD can be significantly scaled up, at a negligible cost in performance, by randomization [Halko et al., 2011]. We outline the randomised SVD of the matrix **R** in Algorithm 1

Algorithm 1 Given $\mathbf{R} \in \mathbb{R}^{m \times n}$, compute approximate rank-k SVD; $\mathbf{R} \approx \mathbf{P_k}\mathbf{\Sigma}_k\mathbf{Q}_k$

```
1: procedure RSVD($\mathbf{R}$, k)
2:     Draw $n \times k$ Gaussian random matrix $\mathbf{\Omega}$
3:     Construct $n \times k$ sample matrix $\mathbf{A} = \mathbf{R\Omega}$
4:     Construct $m \times k$ orthonormal matrix $\mathbf{Z}$, such that $\mathbf{A} = \mathbf{ZX}$
5:     Constuct $k \times n$ matrix $\mathbf{B} = \mathbf{Z}^T\mathbf{R}$
6:     Compute the SVD of $\mathbf{B}$, $\mathbf{B} = \hat{\mathbf{P}}_k\mathbf{\Sigma}_k\mathbf{Q}_k$
7:     Compute $\mathbf{P}_k = \mathbf{Z}\hat{\mathbf{P}}_k$
8:     return ($\mathbf{P}_k$, $\mathbf{\Sigma}_k$, $\mathbf{Q}_k$)
9: end procedure
```

Chapter 8

Bibliography

G. J. J. Adeva, U. B. Cerviño, and R. A. Calvo. Accuracy and Diversity in Ensembles of Text Categorisers. *CLEI Electronic Journal*, 9(1), 2005. 84

D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM. 40

M. S. Amin, B. Yan, S. Sriram, A. Bhasin, and C. Posse. Social referral: Leveraging network connections to deliver recommendations. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 273–276, New York, NY, USA, 2012. ACM. 1

A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Effects of user similarity in social media. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 703–712, New York, NY, USA, 2012. ACM. 16, 37

S. Asur, B. A. Huberman, G. Szabo, and C. Wang. Trends in social media: Persistence and decay. In *Proceedings for the fifth International AAAI Conference on Weblogs and Social Media*, 2011. 36

L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM. 1

L. Backstrom, E. Bakshy, J. Kleinberg, T. Lento, and I. Rosenn. Center of attention: How facebook users allocate attention across friends. In *Proceedings for the fifth International AAAI Conference on Weblogs and Social Media*, 2011. 36

E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 65–74, New York, NY, USA, 2011. ACM. 36

E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. *Facebook report, http://www.scribd.com/facebook*, 2012. 36

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53 – 58, 1989. 73

J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 199–208, New York, NY, USA, 2012. ACM. 85

R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *1st KDDCup'07*, San Jose, California, 2007. 2, 9, 12

J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk. KDD cup and workshop 2007. *SIGKDD Explorations Newsletter*, 9(2):51–52, Dec. 2007a. 59, 66

J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD*, 2007b. 7, 9, 18, 79

T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011. 59

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006. 61

H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4):291–294, 1988. 73

P. B. Brandtzg and O. Nov. Facebook use and social capital — a longitudinal study. In *Proceedings for the fifth International AAAI Conference on Weblogs and Social Media*, ICWSM'11, 2011. 16, 37

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications, manuscript, available at www-stat.wharton.upenn.edu/ buja, 2005. 53

B. Cao, N. N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 159–166, 2010. 15

Ò. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008. 59

J. Chang, I. Rosenn, L. Backstrom, and C. Marlow. epluribus : Ethnicity on social networks. pages 18–25, 2010. 36

P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*, 2010. 13, 54, 56, 60, 63, 66

P. Cremonesi, R. Turrin, and F. Airoldi. Hybrid algorithms for recommending new items. In *HetRec '11*, pages 33–40, New York, NY, USA, 2011. ACM. 13

P. Cui, F. Wang, S. Liu, M. Ou, S. Yang, and L. Sun. Who should share what?: item-level social influence prediction for users and posts ranking. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 185–194, New York, NY, USA, 2011. ACM. 4, 16, 21, 29, 37

J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM. 1

F. Denis. PAC learning from positive statistical queries. In *Algorithmic Learning Theory (ALT)*, volume 1501 of *Lecture Notes in Computer Science*, pages 112–126. Springer Berlin Heidelberg, 1998. 53

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM. 85

C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 213–220, New York, NY, USA, 2008. ACM. 53

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, pages 1871–1874, 2008. 28, 52

J. Fürnkranz and E. Hüllermeier. *Preference Learning*. Springer-Verlag, New York, NY, USA, 1st edition, 2010. 14

Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 176–185, Washington, DC, USA, 2010. IEEE Computer Society. 15, 40, 41, 46

Z. Gantner, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. Personalized ranking for non-uniformly sampled items. In *Proceedings of KDD Cup 2011 Competition*, KDD '11, pages 231–247, 2012. 15, 54

E. Gilbert and K. Karahalios. Predicting tie strength with social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 211–220, New York, NY, USA, 2009. ACM. 37

S. Goel, D. J. Watts, and D. G. Goldstein. The structure of online diffusion networks. In *EC*, EC '12, pages 623–638, New York, NY, USA, 2012. ACM. 36

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *ACM Communications*, 35(12):61–70, Dec. 1992. 8, 9

C. Goller and A. Kchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the IEEE International Conference on Neural Networks*, ICNN-96, pages 347–352. IEEE, 1996. 84

B. Golub and M. O. Jackson. Using selection bias to explain the observed structure of internet diffusions. *Proc. Nat. Academy Sci.*, 107, 2010. 36

C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):13:1–13:19, Dec. 2015. 1

P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. Wtf: the who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 505–514, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee. 1

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011. 57, 58, 87

J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM. 2, 9, 12, 54

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan 2004. 9

R. Hill and R. Dunbar. Social network size in humans. *Human Nature*, 14(1):53–72, 2003. 36

G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012. 3, 71

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, Aug. 2002. 73

G. E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 72

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 73

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society. 14, 56, 57

M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, 2010. 15

C. C. Johnson. Logistic matrix factorization for implicit feedback data. In *NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*, 2014. 59, 60

G. King and L. Zeng. Logistic regression in rare events data. *Political Analysis*, 9(2):137–163, 2001. 54

Y. Koren. Factorisation meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM. 10

Y. Koren. The bellkor solution to the netflix grand prize, 2009. 84

Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, Apr. 2010a. 84

Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transaction of Knowledge Discovery and Data*, 4(1):1:1–1:24, 2010b. 1, 8

Y. Koren, R. Bell, and C. Volinsky. Matrix factorisation techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. 1, 2, 8, 10, 12, 37

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012. 3, 71, 84

A. Krohn-Grimberghe, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM '12*, 2012. 15, 40, 46

M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *International Conference on Machine Learning (ICML)*, 1997. 55

N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 796–797, New York, NY, USA, 2009. ACM. 84

N. D. Lawrence and R. Urtasun. Non-linear matrix factorisation with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 601–608, New York, NY, USA, 2009. ACM. 11

D. Lee and K. Hosanagar. Impact of recommender systems on sales volume and diversity. 2014. 1

J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *Proceedings of the 30th International Conference on Machine Learning*, 2013. 12, 81

J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 450–461, Washington, DC, USA, 2012. IEEE Computer Society. 85

M. Levy and K. Jack. Efficient top-*n* recommendation by linear regression. In *RecSys Large Scale Recommender Systems Workshop*, 2013. 13, 55, 60, 66

S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 811–820, New York, NY, USA, 2015. ACM. 76

W.-J. Li and D.-Y. Yeung. Relation regularized matrix factorization. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1126–1131, San Francisco, CA, USA, 2009a. Morgan Kaufmann Publishers Inc. 16, 17, 21, 37

W.-J. Li and D.-Y. Yeung. Relation regularized matrix factorisation. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1126–1131, San Francisco, CA, USA, 2009b. Morgan Kaufmann Publishers Inc. 4, 29

Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1-3):191–202, 2002. 54

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003. 1, 2, 8, 9, 13, 56

P. Lops, M. de Gemmis, and G. Semeraro. *Recommender Systems Handbook*, chapter Content-based Recommender Systems: State of the Art and Trends, pages 73–105. Springer US, Boston, MA, 2011. 8

H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 931–940, New York, NY, USA, 2008a. ACM. 15, 17, 37

H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorisation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 931–940, New York, NY, USA, 2008b. ACM. 2, 4, 29, 37

H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 203–210, New York, NY, USA, 2009a. ACM. 17

H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 203–210, New York, NY, USA, 2009b. ACM. 2, 29

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 287–296, New York, NY, USA, 2011a. ACM. 16, 17, 21, 37

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 287–296, New York, NY, USA, 2011b. ACM. 2, 29

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. 2008. 19, 26, 44

A. K. Menon, H. Narasimhan, S. Agarwal, and S. Chawla. On the statistical consistency of algorithms for binary classification under class imbalance. In *ICML '13*, pages 603–611, 2013. 54

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013. 3, 71

J. A. Nelder and R. J. Baker. *Generalized Linear Models*. 2004. 3

X. Ning and G. Karypis. SLIM: sparse linear methods for top-n recommender systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, ICDM '11, pages 497–506, 2011. 2, 13, 54, 55, 63

J. Noel, S. Sanner, K.-N. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna. New objective functions for social collaborative filtering. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 859–868, New York, NY, USA, 2012. ACM. 2, 4, 17, 21, 28, 29, 37

R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society. 12, 14, 54, 59

M. J. Pazzani and D. Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007. 1, 8

G. C. Reinsel and R. Velu. *Multivariate Reduced-Rank Regression: Theory and Applications (Lecture Notes in Statistics)*. Springer, 1998. 56

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalised ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press. 14, 54

P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Mar. 1997. 1, 8

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM. 9

M. Riedmiller and H. Braun. Rprop - a fast adaptive learning algorithm. 1992. 75

D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. WWW '11, 2011. 36

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988. 73

D. Saez-Trumper, D. Nettleton, and R. Baeza-Yates. High correlation between incoming and outgoing activity: A distinctive property of online social networks? In *Proceedings for the fifth International AAAI Conference on Weblogs and Social Media*, ICWSM '11, 2011. 23, 36

S. Sahebi and W. W. Cohen. Community-based recommendations: a solution to the cold start problem. In *RSWEB*, 2011. 15, 40

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorisation using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 880–887, New York, NY, USA, 2008a. ACM. 11

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorisation. In *Advances in Neural Information Processing Systems*, 2008b. 2, 10, 12, 28

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM. 3, 73

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. 1, 2, 8, 9, 12, 54

A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM. 15

P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. Institute of Electrical and Electronics Engineers, Inc., August 2003. 84

A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM. 15

P. Singla and M. Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 655–664, New York, NY, USA, 2008. ACM. 16, 37

P. Sollich and A. Krogh. Learning with ensembles: How over-fitting can be useful. 1996. 84

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning*, ICML '03, pages 720–727, 2003. 10

D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 111–120, 2009. 11

L. Tang and P. Harrington. Scaling matrix factorization for recommendation with randomness. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 39–40. International World Wide Web Conferences Steering Committee, 2013. 14, 66, 68

L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 817–826, New York, NY, USA, 2009. ACM. 45

T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1064–1071, New York, NY, USA, 2008. ACM. 73

A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013. 40, 81

G. Ver Steeg, R. Ghosh, and K. Lerman. What stops social epidemics? In *Proceedings for the fifth International AAAI Conference on Weblogs and Social Media*, ICWSM '11, 2011. 36

J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 47–56, New York, NY, USA, 2009. ACM. 3

H. Wang, M. Terrovitis, and N. Mamoulis. Location recommendation in location-based social networks using user check-in data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 374–383, New York, NY, USA, 2013. ACM. 85

H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, New York, NY, USA, 2015. ACM. 76, 81

D. J. Watts and P. S. Dodds. Influentials, networks, and public opinion formation. *Journal of Consumer Research*, 2007. 36

C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys'09. ACM, 2009. 36

Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 153–162, New York, NY, USA, 2016. ACM. 76, 81

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings, 2015. 84

S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 537–546, New York, NY, USA, 2011a. ACM. 16, 21, 37

S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 537–546, New York, NY, USA, 2011b. ACM. 29

M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 458–461, New York, NY, USA, 2010. ACM. 85

K. Yu and V. Tresp. Learning to learn and collaborative filtering. In *Neural Information Processing Systems Workshop on Inductive Transfer: 10 Years Later*, 2005. 56

T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46(1-3):91–129, 2002. 52

Y. Zhang and M. Pennacchiotti. Predicting purchase behaviors from social media. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 1521–1532. International World Wide Web Conferences Steering Committee, 2013. 1

Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '14, pages 83–92, New York, NY, USA, 2014. ACM. 11

Zhang, Zi-Ke, Liu, Chuang, Zhang, Yi-Cheng, and Zhou, Tao. Solving the cold-start problem in recommender systems with social tags. *EPL*, 92(2), 2010. 15, 40