Developing Optimization Models for Dyadic Prediction in the Context of Sparse Data

A Doctoral Dissertation Submitted in Partial Fulfilment of the Requirements for the

Fellow Program in Management

Indian Institute of Management Ranchi

By

Bipul Kumar

Thesis Advisory committee

1. **Dr. Pradip Kumar Bala [Advisor]……**
2. **Dr. Amit Sachan [Member]……**
3. **Dr. N. Sivasankaran [Member]…….**

DEED OF DECLARATION

I, **Bipul Kumar**, hereby submit my FPM Thesis, entitled **Developing Optimization Models for Dyadic Prediction in the Context of Sparse Data** for examination, in partial fulfilment of the requirements for the award of the degree of Fellow Program in Management. I truthfully declare that the above-titled Thesis is a product of my original research investigation.

I further declare that, should the IIM Ranchi discover that a substantial portion of my Thesis is plagiarized, IIM Ranchi reserves the right to recall my FPM title and cancel the FPM Award granted to me.

Date: _____ Signature: _____

Indian Institute of Management Ranchi Name: _____

Acknowledgements

I would like to express my sincere gratitude to those who helped me complete this dissertation. Foremost, I am deeply grateful to my advisor, Dr Pradip Kumar Bala, for his excellent guidance and support throughout the dissertation process. His keenness and drive in research have always inspired me, and I appreciate his being such a great mentor with his patience and constant encouragement.

I would also like to thank my committee members, Prof. Amit Sachan, and Prof. N. Sivasankaran, who provided me with helpful advice and insight towards the completion of my dissertation. I also thank other faculty members at Indian Institute of Management for their readiness to help me. I would also like to thank reviewers of 'operational research' and 'journal of theoretical and applied e-commerce research' for their valuable suggestions which has help in publishing part of this thesis in the respective journals.

A special thanks to Abhishek Srivastava, Shubhadeep Mukherjee, and Rahul Kumar for open discussion on issues related to dissertation work. Without their inputs, this dissertation would not have been so well-rounded. I feel greatly privileged to be a part of this wonderful group and want to extend my special appreciation to the FPM office for the timely resolution of any administrative issues.

Last but most importantly, I thank my parents, who made me strong enough to continue the journey of completing this dissertation. I am also greatly indebted to my beloved brothers and sister for their unconditional love, support, and prayers.

Abstract

There is a renewed interest in Recommender Systems (RS) research after the Netflix prize competition. In parallel, there is also a growth of social media which has driven research in link prediction, with the aim of predicting labels about friends, groups, etc. Both these problems involve the prediction of labels (star ratings or friendship) between a pair of entities (user-movie or user-user). This type of problem is known as a dyadic prediction. Dyadic prediction in context of 'personalization' on e-service platforms is a broad research area of this dissertation. In particular, this dissertation seeks to contribute three folds on the implementation of personalization tools on e-commerce.

The first contribution of this work is to develop novel optimization algorithms in dyadic prediction that improves the precision of RS. Since the foremost utility of RS is to recommend good items to users, it is worth concentrating on improving the precision of the RS model. Keeping in line with this thought, this work pursues to develop an entirely novel latent factor model which optimizes its parameters on the likes and dislikes of users for items instead of numerical ratings. The novelty of the proposed algorithm is to optimize latent factors of user and items that are derived from the likeability of users to items. The proposed algorithm has also been extended to the ranking task which considers matching the order of labels rather than the actual label itself.

Secondly, the work extends regularized singular value decomposition (RSVD), state-of-the-art algorithm in rating prediction, to provide optimal matching for long-tail items to idiosyncratic users based on prior likeness to long-tail items. The second model is also extended to an additive latent factor model that promotes long-tail items to users based on their prior interests. This work discusses business goals that a firm has and pushes according to its requirement so that the trade-off between novelty and diversity can be linked with the accuracy of the RS.

Lastly, this work also develops a new model for context-aware recommender systems (CARSs) which is an upcoming and challenging research area. The priority of this work would be to model contexts with lesser complexity. Comprehensive experiments are conducted on publicly available benchmark datasets such as Amazon and Movie Lens to verify proposed research work with existing state-of-the-art algorithms. The experimentation evaluation frameworks are in line with existing frameworks proposed in related literatures.

“A **lot of times**, **people don't know what they want** until you show it to them.”

-Steve jobs

Table of Contents

List of Figures

List of Tables

List of major journal publications

- "Ranking the recommendations of e-commerce portal using list wise loss function" is accepted for publication in Operational Research (ssci indexed) which has been carved out from chapter 3 and chapter 4

- "Fattening the long-tail items in E-commerce" is accepted for publication in Journal of Theoretical and Applied Electronic Commerce Research (ssci indexed) which has been carved out from chapter 5

1 Introduction

1.1 Dyadic data

Dyadic data refers to a domain with two finite sets of objects in which observations are made for dyads (pairs corresponding with one element from either set). Examples include sparse matrices of user- product ratings in recommender systems (RS), documents–words pairs in text retrieval systems, and user-user pairs in link prediction in the case of social networking sites (Meeds, Ghahramani, Neal, & Roweis, 2006; Hofmann, Puzicha, & Jordan, 1999). Sparseness is a key challenge in modelling dyadic data in the context of recommender systems and link prediction. Sparseness arises due to a lack of interaction available on these kinds of dyadic data, viz., a typical e-commerce platform has sparseness because not all users would be making all their purchases on e-commerce platforms. This poses a challenge in applying a model on such datasets.

To begin with, this chapter emphasizes the usefulness of dyadic prediction by describing the various examples in the context of the business world. Then, it provides an overview of the list of contributions in modelling sparse data to 'personalization' on e-service platforms, and finally presents the structure of dissertation.

1.2 Dyadic prediction

1.2.1 Recommender systems

In the age of Internet of Things, there is a growing importance of personalized recommendation systems (RS). RS are typical software solutions used in e-commerce for personalized services (Sarwar, Karypis, Konstan, & Riedl, 2000a). It helps customers to find

interesting items by providing recommendations based on their prior preferences. It also benefits e-commerce sites that offer millions of products for sale by targeting the right customer for the right product (Kim, Yum, Song, & Kim, 2005).

RS can be described as an information filtering technology that is used to present information on items to the user that are in line with the user's tastes. These systems involve predictive models, heuristic search, data collection, user interaction, and model maintenance. RS collects information on the preferences of its users for a set of items. The information can be collected explicitly and/or implicitly. RS may also use demographics of users such as age, location, and gender. There is a growing trend of utilizing social information like followers, followed, tweets for personalized recommendation, etc. (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013).

1.2.2 Text retrieval systems

A set of queries is stated by a user against a set of textual records, which has to be matched in a text retrieval system. With huge databases of e–resources available in the form of text, it is advantageous to have documents which match specifically to a user's query be produced. The text retrieval systems have the '{documents, word} pair' as a dyadic data which can be extracted in the form of a query based on the input of a user (Pahikkala et al., 2014).

1.2.3 Link prediction

The study of online networks has been a priority in current scenario. There is a surge on its usefulness in the industry as well. Social networks are defined as structures whose nodes represent users embedded in a social context and its edges represent interaction, collaboration, or influence between entities. Link prediction involves accurately predicting the edges that will be added to the network during the next time period based on social networks available in the current time period (Liben-Nowell & Kleinberg, 2003).

1.3 Motivation

The three problems described above are relevant in the current business scenario. However, this dissertation has delimited the scope on RS which are 'web personalization' tools on e-commerce platforms. E-commerce platforms provide a lot of research opportunities due to their constant evolution. This dissertation seeks to leverage the research opportunities in web personalization by introducing optimization models to solve various aspects and issues described in detail later in the chapter.

As noted by Doukidis, Pramatari, and Lekakos (2008) in their review chapter, the ever-increasing rise of digital economy is creating abundant opportunities for operations research (OR) applications. Decision support systems (DSS) that enhance the electronic service offering by providing advanced decision support capability to the electronic service of consumers are also considered a key research area in OR (Doukidis et al., 2008). Another research work describing OR application related to web personalization suggests the use of optimization based algorithms and terms this area as a "wealth of opportunities" for the OR community to contribute to (Olafsson, Li, & Wu, 2008). Similar observations for using quantitative modelling techniques like machine learning for 'personalization' in web based platforms are also made by Bose and Chen (2009).

In addition, the business value that personalization adds to web based platforms can be confirmed with various industry reports mentioned in previous research literature. Some of the top companies which have taken strategic advantages in their business by personalization tools have been listed below:

- Netflix, a DVD rental company generates 60% of rentals from the implementation of RS (Hosanagar, Fleder, Lee, & Buja, 2014)

- RS has helped Google News in increasing article view by up to 38% (Das, Datar, Garg, & Rajaram 2007).
- 35% of sales by Amazon are contributed by RS (Hosanagar et al., 2014).

Web personalization

A few definitions of 'web personalization' in the literature are stated below:

According to Mulvenna, Anand, and Büchner (2000), personalization on e-commerce context is "the provision to the individual of tailored products, service, information, or information relating to products or services. This broad area also covers recommender systems, customization, and adaptive web sites." Eirinaki and Vazirgiannis (2003) define web personalization as "any action that adapts the information or services provided by a web site to the needs of a particular user or a set of users, taking advantage of the knowledge gained from the users' navigational behavior and individual interests..." Adomavicius and Tuzhilin (2005) summarize that "personalization tailors certain offerings (such as content, services, product recommendations, communications, and e-commerce interactions, which is in fact a facet of comparison-shopping agents) by providers (such as e-commerce web sites) to consumers (such as customers and visitors) based on knowledge about them, with certain goal(s) in mind."

Being accurate is one of the major objectives of web personalization. Offering the right products to a user not only satiates his needs but also enables trust on the web platforms. Another view of personalization is that it should help customers widen their interest in products rather than narrow their interests. However, there is an argument that personalization tools on e-commerce platforms are fragmenting the online customers and narrowing the scope of consumption (Hosanagar et al., 2014). Some of the research chapters have also argued that RS mostly generate recommendations of popular products which lead to 'long tail' problems in within them (Goel, Broder, Gabrilovich, & Pang, 2010).

Context is another factor which plays an important role in personalization on web platforms. According to Dey (2001), "Context is any information that can be used to characterize the situation of an entity." Using contextual data for personalization adds to the problem of quantitative modelling mainly due to two reasons:; a) higher dimension of data to be considered, and b) increase in sparseness due to the unavailability of interaction of users with items and contexts.

Based on the above discussions, this dissertation has identified three major questions in this area of research:

1. How to apply optimization algorithms for personalization of products for users to improve the overall accuracy?
2. How to address the accuracy vs diversity dilemma without affecting personalized recommendations to users?
3. How to handle high dimensional data of context- aware systems while efficiently keeping intact the core characteristic of personalization?

1.4 Contribution of the dissertation

To provide concrete answers to the above research questions, this dissertation makes the following key contributions:

- The main objective of recommender systems is better personalization or in a more technical sense, to predict the ratings of unseen items by users as accurately as possible. This has been a traditional objective of the RS algorithm and is termed as the rating prediction task. However, there is another view of 'personalization' on web platforms, where the objective is to present the good or interesting items to a user based on his prior purchase. In this view, personalization is often treated as a classification task, and

the objective is to learn the model in a manner that good and bad items can be classified. Moving ahead with the classification task, in a more practical sense, only 'top-k' items in ordered rank are provided to a user. This problem is often termed as the ranking prediction task where the objective is to build a model such that the first 'top-k' item for every user is provided in order.

In learning any classification models, the features or the independent variables are very important. However, the real database often contains only ratings of users ascribed to items and those ratings are also very few. This means the features do not exist and moreover the data is sparse. This poses a real challenge in dealing with these kinds of problems. The acquisition of features for each user and item externally is costly and sometimes impossible. The solution lies in learning the features from the database alone, but sparseness in the dataset makes it also a difficult proposition.

To learn the features from the sparse database, this work proposes a novel model, the 'cosine based latent factor model'. The latent features of each user and items can be learnt based on the user's rating for a product. The key concept is that the higher the cosine similarity between features of users and products, the higher will be the user's rating for the product, and vice versa. This is a unique approach in solving the problem of generating a good recommendation model where this dissertation emphasizes on classifying good and bad items and ranking of items based on a user's preference. Summing up with the first objective, this dissertation proposes a 'cosine based latent factor model' that is applicable in learning parameters of a model for the rating prediction task, classification task, and ranking task.

- Another key objective of this dissertation is to develop an algorithm catering to 'long-tail' items in recommendation systems. 'Long-tail' items are less popular items on e-commerce platform but can be a good recommendation to idiosyncratic users. The

accurate and popular recommendation system limits the benefit of an RS by providing users with already known products which fizzles out the possibility of cross-selling. To address this problem this dissertation develops an algorithm that incorporates 'diversity' and 'novelty' in a single model of RS. Algorithms which have mainly focussed on 'long tail' have tried to address the problem by constrained optimization and re-ranking the RS list (Ribeiro, Lacerda, Veloso, & Ziviani, 2012). By taking accuracy as the goal of the RS and taking 'diversity' as constraints, RS models have addressed the 'long tail' problem. The proposed model makes a key contribution by developing an algorithm that captures the prior taste of every user in the context of long-tail items. The formulation of the 'long-tail' problem in this manner is unique and has not been considered in any of the previous approaches. For learning this formulation, a stratified learning technique of the latent factor model has been proposed, which not only improves personalization but also recommends the 'long tail' items to idiosyncratic users.

- Finally, varying contexts such as time, place, weather, and mood impact an individual's preference, and therefore, providing personalization services with respect to context is indispensable. The problem in modelling context arises due to the higher dimensionality of the dataset which ensues from multiple contexts. In the case of RSs, context has been incorporated in modelling and predicting interesting items to users based on their prior interaction with items and context. It is important to note that inclusion of context in modelling of RS has helped better personalization in previous works. The main contribution of the proposed model is its ability to capture 'user-item-context' interaction with a lesser number of parameters, thereby reducing the model complexity. The proposed model also introduces the concept of an ensemble of models to generate recommendations in multi-dimensional contexts.

1.5 Structure of the dissertation

The dissertation is organized in following manner: Chapter 2 reviews relevant literature comprising of traditional recommender systems, latent factor models used in RS, an overview of the accuracy vs diversity/novelty dilemma in recommendation and an overview of context-aware recommendation. The literature reviews are followed by research gaps and finally the objectives that emerge out of research gaps. The following four chapters consist of four studies. Chapters 3 and 4 deals with the first objective of this dissertation. Chapter 3 presents a new optimization model to learn latent features from a user-item rating matrix. The latent features that are learnt are applied to generate a recommendation list. Chapter 4 extends the proposed model for ranking the recommended items. Chapter 5 deals with the second objective of the dissertation. It presents a new optimization model to solve the apparent accuracy vs diversity/novelty dilemma. The model aids diversification of recommendations from user interpretation and not from the system perspective. Chapter 6 deals with the third objective of the dissertation. It presents a contextual model in RS with lesser complexity without compromising on the accuracy of the RS. Chapter 7 summarizes the results and contributions of the three themes with a discussion of scope for future research.

2 Literature Review, literature gap, and objectives

2.1 Research background

This chapter, firstly, presents a formal definition of the dyadic prediction problem. Thereafter, the dyadic prediction problem has been described in the context of web personalization tools, popularly known as 'recommender systems'. With particular emphasis on personalization tools on web platforms, this chapter will discuss popular methods in literature that describe web personalization services. Since the focus of this dissertation is on developing optimization based models in the context of sparse data for web personalization services, therefore, relevant literatures on this topic have been reviewed.

2.1.1 Overview of Dyadic prediction

More formally, dyadic data refers to two abstract sets of objects, $U = \{u_1, u_2 \ldots\ldots\ldots u_m\}$ and $V = \{v_1, v_2 \ldots\ldots\ldots v_n\}$; and observations S are made for dyads (u_i, v_j) with the corresponding label r_{ij}. However, it may be possible that dyads pair labels are null for sets of U and V. This means that there are some values in S which may be observed while others may be unobserved. A training set $\mathcal{T} = \{((u_i, v_j), r_{ij})\}_{i=1,j=1}^{t1,t2}$ is given where each pair (u_i, v_j) is dyad while r_{ij} is the corresponding observed label. The task is to predict the label for the new set of dyads(u_i', v_j'). In dyadic prediction, the objective is to learn a function that maps U and V with a corresponding label r_{ij} in a training set such that its function minimizes the error between the predicted label $(\widehat{r_{ij}})$ and actual label (r_{ij}) in the training set. The learnt function will be used to predict the label for new sets of dyads.

2.1.2 Web personalization and Dyadic prediction

Table 1: Dyadic data representation

R	I1	I2	I3	I4	I5
U1	2	NA	5	2	NA
U2	1	3	NA	1	4
U3	5	NA	3	4	NA
U4	4	NA	4	5	NA
U5	NA	1	NA	NA	4

Table 1 depicts a typical matrix representation of a dyadic data. Here the dyads are users in rows, and items in columns. The interaction between users and items results in feedback by users in the form of ratings. The ratings imply the degree of interest for items by users on a scale of 1 to 5, where 1 implies the lowest degree of interest, and vice versa. To model these kinds of situations web personalization tools have been studied in academic disciplines as well as implemented on various e-services portal like Amazon, Netflix, Google news, etc. The analogy of web personalization tools can be drawn with personalization agents in brick and mortar retail shops who aid a customer's decision to buy a product of their interest.

Personalization agents form an integral part of a shopping experience. These agents help in creating customer value and the ensuing customer loyalty (Burke, 2002). The advent of internet shopping has positively contributed towards the evolution of web personalization agents. Though the web platform offers customers choice and convenience, users also face challenges to find the products they want due to plenty of choice (Maes, 1999). These agents not only aid users to find the product they want but also improve the sales of e-commerce firms (Hostler, Yoon, & Guimaraes, 2012). One of the important aspects of academic research on web-personalization agents has been to develop automated software that improves the accuracy of the overall system. Collaborative filtering algorithms are considered

successful models in solving the problem of web personalization (Bobadilla et al., 2013). Other major kinds of algorithms for solving web-personalization problem are content- based filtering and hybrid models (Bobadilla et al., 2013) .

2.1.3 Collaborative filtering models for web-personalization

Historically, people have relied on recommendation by peers for watching a movie, buying electronic goods, etc. The above notion of recommendation is applied on web platforms for developing personalization tools. In academic literature, this concept of providing recommendation by involving peers is known as collaborative filtering. Collaborative filtering involves collaboration of users to filter interesting item based on likes (dislikes) of users for various items (Goldberg, Nichols, Oki, & Terry, 1992). Here the {user, item} pair can be regarded as dyads, and the labels are the users' numeric ratings for an item e.g. on the scale of 1 to 5. The task for predicting labels for the unobserved {user, item} pair and recommending higher predicted rating for an active user is a typical case of collaborative filtering (Goldberg et al., 1992).

Collaborative filtering (CF) is one of the most popular approaches in RS. The fundamental assumption of CF is that "if users X and Y rate *n* items similarly, or have similar behaviors (e.g., buying, watching, listening), and hence will rate or act on other items similarly" (Su & Khoshgoftaar 2009). CF techniques use a database of preference for items by users to predict additional items to them. In a typical CF database, there is a list of users (say *m* users) and items (say *n* items) and each user either explicitly or implicitly indicates their preferences corresponding to items. The preferences are not indicated by every user for every item as not every user will consider every item when there are millions of items in e-commerce. So, to generate a recommendation list, an active user can be recommended items with the help of other users who have indicated similar preferences for items in the CF database.

2.2 Overview of collaborative filtering techniques

CF is classified into two types: memory based and model based. Memory based CF are those which generate a recommendation list based on similarity measures either between user-user or between item-item. The similarity measures generally employed are cosine similarity or Pearson Correlation similarities, which are quite effective. Model based CF learns parameters of models using data mining and machine learning algorithm on training data. The learned parameters are used to predict real data. Latent factor models, Bayesian networks, Latent Dirichlet Allocation(LDA), and Markov Decision Process based models are frequently researched models in model based CF (Su & Khoshgoftaar, 2009; Cheung, Kwok, Law, & Tsui, 2003; Kumar, Raghavan, Rajagopalan, & Tomkins, 2001; Marlin, 2004; Shani, Heckerman, & Brafman, 2005).

2.2.1 Memory based CF

According to Ekstrand (2010), a memory based CF typically consists of the following steps

1. Construct a user profile from labels (rating) provided by other users on items in a database
2. Select neighbours of a target user using similarity measure (Pearson correlation, cosine similarity) based on the labels (rating) information of co-rated items
3. Predict the ratings of the target user on target items using prediction function.

$$\mathrm{pred}(a,p) = \bar{r_a} + \frac{\sum_{b=1}^{t} \mathrm{sim}(a,b) \times (r_{b,p} - \bar{r_b})}{\sum_{b=1}^{t} \mathrm{sim}(a,b)} \quad (2.1)$$

Here, $\bar{r_a}$ is average rating of active user; $r_{b,p}$ indicates rating of user b for item p; $\bar{r_b}$ is average rating of user b; t is top t number of users who have rated item p; $\mathrm{sim}(a,b)$ represents similarity between user a and b.

4. Recommend the top-n items predicted for the target user according to the prediction function

Similarity measures

RS has traditionally used cosine similarity and Pearson correlation as the similarity measure.

- **Pearson correlation:** It computes statistical linear relationship between items based on the ratings of users (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Hill et al., 1995).

$$\mathrm{sim}(a,b) = \frac{\sum_{i \in I_a \cap I_b}(r_{a,i} - \overline{r_a})(r_{b,i} - \overline{r_b})}{\sqrt{\sum_{i \in I_a \cap I_b}(r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i \in I_a \cap I_b}(r_{b,i} - \overline{r_b})^2}} \quad (2.2)$$

Where a, b represents users for which similarity is being calculated; $r_{a,i}$ is the rating of user a for item i; $r_{b,i}$ is the rating of user b for item i ; $\overline{r_a}$, $\overline{r_b}$ are average rating of user a and b; $I_a \cap I_b$ represents the set of commonly rated items for user a and b.

The problem of using this similarity measure is that there is equal weightage given to a few commonly rated items as well as substantial commonly rated items. In order to alleviate this problem a threshold limit (Y) is set for the commonly rated item, scaling the similarity when the commonly rated item falls below this threshold (Herlocker, Konstan, Borchers, & Riedl, 1999). In this case the above computed similarity measure is multiplied by a factor $\min\{\frac{|I_a \cap I_b|}{Y}, 1\}$.

- **Constrained Pearson correlation:** Instead of taking the relative deviation for each user, absolute deviation can also be used for numerical discrete ratings on the 5-point scale. Absolute deviation fixes a neutral scale, 3 in case of the 5-point scale, and then

computes the similarity measure. This measure helps in correlating absolute likes/dislikes rather than relative deviations (Shardanand & Maes, 1995).

$$\mathrm{sim}(a,b) = \frac{\sum_{i \in I_a \cap I_b}(r_{a,i} - r_z)(r_{b,i} - r_z)}{\sqrt{\sum_{i \in I_a \cap I_b}(r_{a,i} - r_z)^2}\sqrt{\sum_{i \in I_a \cap I_b}(r_{b,i} - r_z)^2}} \quad (2.3)$$

- **Cosine similarity:** This measure considers items as vectors in 'n' dimensional space and computes the similarity as the cosine of the angle between vectors (Billsus & Pazzani, 1998).

$$\mathrm{sim}(a,b) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \quad (2.4)$$

Where $\vec{a}, \vec{b}$ represents user rating in the form of vector for all items that have been rated by a and b respectively.

2.2.2 Model based CF

The memory based CF, though simple in implementation, performs well in small to medium databases. One more limitation of the memory based CF is that searching for similar neighbours becomes difficult as the database gets sparser. The problem in a real-time RS is that users need to be added very frequently in the database which tends to make the database large and complex. This is known as scalability of RS, which can generally not be handled by memory based CF. To alleviate the issue of scalability and perform real time recommendation, the model based CF has been proposed as a better alternative.

Latent factor model

Learning latent factors for a database is a powerful method in the model based CF. This dissertation has used this model as the base model, as it has been shown to give state of- the-art performances (Baltrunas, Ludwig, & Ricci, 2011; Bauer & Nanopoulos, 2014; Cremonesi, Koren, & Turrin, 2010; Huang, Zhong, & Yao, 2014; Konstan & Riedl, 2012; Nguyen,

Karatzoglou, & Baltrunas, 2014; Park, Kim, Choi, & Kim, 2012; Weimer, Karatzoglou, & Bruch, 2009). Specifically, the focus is on matrix factorization methods, which are the most popular latent feature approaches in CF. This approach is best known for being a key ingredient in the Netflix Prize winner model (Koren, 2009). The latent factor assumes that the data points have some underlying features which can be extracted to explain the observed ratings. Matrix factorization models such as singular value decomposition (SVD) have gained popularity due to their scalability and accuracy.

The following table 2 summarizes previous works on latent factor models applied in the field of RS.

Table 2: summary of latent factor models

Models	**Method**	**Key features**
SVD (Sarwar, Karypis, Konstan, & Riedl, 2000b)	Deterministic	• Decomposes the user-item preference (rating) matrix into three matrices, viz., user feature matrix, singular matrix, and item feature matrix of lower rank • Sparse data in user-item preference (rating) matrix to be filled by imputation • Not scalable
Incremental SVD (Sarwar, Karypis, Konstan, & Riedl, 2002)	Deterministic	• Decompose the user-item preference (rating) in the same way as SVD • Incremental SVD is made scalable and faster by applying folding-in technique by adding new users and items

		• Folding-in can result in loss of quality
SVD+ANN (Billsus & Pazzani, 1998)	Deterministic	• Convert user-item preference (rating) matrix into Boolean form; resulting in the matrix filled with 0s (dislike) and 1s (like) • Compute SVD in the same way as above • Train an ANN with user and item feature vectors computed using SVD which is used for prediction
Regularized SVD (Paterek, 2007)	Deterministic	• Decomposes the user-item preference (rating) matrix into two matrices, user feature matrix and item feature matrix of lower rank • Parameters are estimated by minimizing the sum of squared residuals against user-item preference (rating), one feature at a time, using the gradient descent method with regularization and early stopping.
SVD++ (Koren, 2008)	Deterministic	• Integrates implicit preference (purchase behavior) with regularized SVD • It is regarded as the best single model in Netflix Prize for accurate prediction.

SVD + Demographic data (Vozalis & Margaritis, 2007)	Deterministic	• Demographic data and SVD are combined to predict the rating • Utilizes SVD as an augmenting technique and demographic data as a source of additional information in order to enhance the efficiency and improve the accuracy of the generated predictions
Probabilistic latent semantic analysis (pLSA) (Hofmann, 2004)	Probabilistic	• Introduces latent class variables in a mixture model setting to discover user communities and prototypical interest profiles using the statistical modeling technique • It can be thought as probabilistic modeling of SVD • The expectation maximization (EM) algorithm ensures learning probabilistic user communities and prototypical interest profile
Probabilistic matrix factorization (PMF) (Salakhutdinov & Mnih, 2008)	Probabilistic	• Full Bayesian analysis by introducing prior distribution over latent factors of items and users. • , Training of parameters in PMF is done using the Markov Chain Monte Carlo (MCMC) technique to avoid over-fitting • Ensures improvement in accuracy in comparison to SVD

Regression-based latent factor model (RLFM) (Agarwal & Chen, 2009)	Probabilistic	• Ratings are predicted on the basis of the features of users and items as well as latent features learned from the database using SVD • PMF uses zero mean prior over latent factors but in RLFM the prior is estimated by running regression over features of items and users. • Suitable for cold start and warm start situations in RS
Latent Factor Augmented with User preference Model (LFUM) (Ahmed et al., 2013)	Probabilistic	• A hybrid model that combines the observed item attributes with a latent factor model • It doesn't learn a regression function over item attributes but rather learns a user-specific probability distribution over item attributes • Training of dataset is done using discriminative Bayesian personalized ranking (BPR) which takes both purchased and non-purchased items by users into account
Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan 2003)	Probabilistic	• While pLSA does not assume a specific prior distribution over the number of dimensions in hidden variables, LDA assumes that priors have the form of the Dirichlet distribution

		• Gibbs sampling or Expectation maximization (EM) is used to estimate the parameters of LDA
Probabilistic factor analysis (Canny, 2002)	Probabilistic	• Factor analysis is a probabilistic formulation of a linear fit, which generalizes SVD and linear regression. • EM is used to learn the factors of the model.
Eigentaste (Goldberg, Roeder, Gupta, & Perkins, 2001)	Deterministic	• Offline phase: uses principal component analysis(PCA) for optimal dimensionality reduction and then clusters users in the lower dimensional subspace • Online phase: uses eigenvectors to project new users into clusters and a lookup table to recommend appropriate items
Maximum-margin Matrix Factorization (MMF) (Rennie & Srebro, 2005)	Deterministic	• Decomposes the user-item preference(rating) matrix into two matrices: user feature matrix and item feature matrix • It works on the principle of lowering the norm of matrices instead of reducing the rank of matrices

Non –parametric matrix factorization (Yu, Zhu, Lafferty, & Gong, 2009)	Deterministic	• Decomposes the user-item preference (rating) matrix into two matrices: user feature matrix and item feature matrix • In non-parametric matrix factorization, the number of factors is learned from the given data rather than prefixing it to a lower rank.
Discrete wavelet transform (DWT) (Russell & Yoon, 2008)	Deterministic	• Haar wavelet transformation is used on the original user-item preference matrix • k-nearest neighborhood model is used over transformed matrix for prediction of rating of the test user
Restricted Boltzmann Machine (RBM) (Salakhutdinov, Mnih, & Hinton, 2007)	Probabilistic	• Two-layer undirected graphical models with hidden units which learn features of users and items • It is a scalable method for rating prediction

2.2.3 Research Gap 1

The summarized works on latent factor models in the above table illustrate the various techniques that are employed to solve the rating prediction problem in the context of web personalization tools on e-commerce platforms. The objective of the above models is to predict the ratings accurately, however, in a more practical scenario the e-commerce platforms like Amazon, Flipkart, etc. provide top-k list of good items in a recommendation list. The top-k recommendation problem is different from the rating prediction problem and

its accuracy can be measured by decision support metrics like precision, recall, and f-measure. To establish the accuracy of the rating prediction problem, the root mean square error (RMSE) and mean absolute error (MAE) are widely used metrics. This dissertation develops a novel latent factor model to solve the prediction of top-k list of good items in a recommendation list and is presented in subsequent chapters.

The various matrix factorization schemes such as RSVD suffer from out of bound predictions. Since the function is not bounded there is a possibility that the obtained predicted values may fall out of range (Salakhutdinov & Mnih, 2008). The predicted values so obtained are clipped (Paterek, 2007) or passed through a bounded function such as a logistic function (Salakhutdinov & Mnih, 2008). These may not be appropriate since the mapping function of the actual rating in a training set of data is not mapped according to the bounded function (Salakhutdinov & Mnih, 2008).Clipping the out of bound values of predicted ratings does not ensure that the learning of latent features will be accurate, which is an important part of the rating prediction. The clipping or applying of the logistic function is also not intuitive. This dissertation proposes a cosine based latent factor model in chapter 3 to overcome these shortcomings and to model the latent features in an intuitive manner.

Further, the position of items in the top-k list of good items is also an important parameter in determining the accuracy of a model. In an e-commerce setup, a customer is often provided with a top-k ranked list of items instead of predicted rating for products or top-k list of good items in the recommendation list. Hence, in order to cater to this set of problems, models that can directly optimize ranks of the items are gradually studied in the literature of recommender systems (Balakrishnan & Chopra , 2012; Harrington, 2003; Weimer & Karatzoglou, 2007). This ranking model is often termed as ‘learning to rank’ in the machine learning community where it has attracted broad attention due to its importance in

information retrieval and collaborative filtering (Karatzoglou, Baltrunas, & Shi, 2013; Rupnik, 2010; Valizadegan & Jin, 2009; Xia, Liu, Wang, Zhang, & Li, 2008). The key idea behind the position of items in the top-k list of recommended items is that users look at the items at the top of the recommendation list rather than those at the bottom of the list more often. Hence, it is important that the recommendation system should be such which generates a list of good items at the top positions rather that at the bottom of the recommendation list in a top-k list of recommendations. In order to measure the accuracy of a recommendation system that caters to ranking task, the normalized discounted cumulative gain (NDCG) metric is often used in information retrieval, and has been subsequently adopted in recommender systems (Balakrishnan & Chopra , 2012). In chapter 4, this dissertation extends the proposed model that solves the prediction of top-k list of good items to the ranking task of t top-k items in the recommendation list.

2.3 Accuracy vs diversity dilemma

Accurate recommendations are one of the major objectives of a recommendation system; however, relying on accuracy alone may not fully utilize the potential of a recommender system (Ribeiro et al., 2012). The following example will illustrate what an RS can miss while relying merely on accuracy. Suppose, a user looks for a ‘Star Wars: Episode IV - A New Hope” movie and instead he is recommended “Star Wars: Episode VI - Return of the Jedi”. There is a high probability that he might already know about the movie that he is being recommended. Therefore, although the recommendation is accurate, it is not adding much value for the user. It has been rightly suggested by previous researchers that the contribution of RSs as web personalization tools is to help the discovery of newer items (Herlocker, Konstan, Terveen, & Riedl, 2004; Hu & Pu, 2011).

The importance of diverse recommendations has been emphasized in several studies (Clemons & Nunes, 2011; Gan & Jiang, 2013; Pu, Chen, & Kumar, 2008; Said, Kille, Jain, & Albayrak, 2012; Zhang & Hurley, 2008; Zhou et al., 2010; Ziegler, McNee, Konstan, & Lausen, 2005)**.** Apparently, it has been observed that more diverse recommendations, lead to more sales of obscure/niche items. This could be beneficial for both individual users and web based market platforms (Brynjolfsson, Hu, & Smith, 2010; Goldstein & Goldstein, 2006). In another recent research, it has been empirically established that product diversity is also positively related to consumer retention rates (Park & Han, 2013).

While more diverse recommendations would be helpful for individual users, they could also be beneficial for e-commerce platforms. Since diverse recommendation may help users to navigate to niche items of their choice, thereby reducing the search time for suitable items, it also increases their trust towards such recommender systems. This may in turn generate a 'recommendation-buying' spiral cycle which means that users will tend to buy from recommendation lists, since they will have learnt to rely on recommender systems (Yoon, Hostler, Guo, & Guimaraes, 2013).. Unexpected yet useful recommendations will indeed help e-commerce sellers to utilize their unlimited shelf space which is a major advantage over brick and mortar shops (Said et al., 2012; Vargas, Castells, & Vallet, 2011; Zhang, 2009). In this way, recommender systems can act as personalized agents to promote cross selling and upselling of products, which are recognized as traditional marketing practices in the brick and mortar setup.

There are two views of diverse recommendation, viz., individual diversity and aggregate diversity. Individual diversity of recommendations for a user can be measured by calculating the average dissimilarity between all pairs of items that have been recommended to a user (Zhang & Hurley, 2008). Aggregate diversity, on the other hand, is calculated across all users

and is measured by the total number of distinct items among top-N items recommended across all users (Adomavicius & Kwon, 2011; Augustin, Niemann, & Wolpers, 2013).

In contrast to *individual diversity*, the *aggregate diversity* of recommendations across all users has been relatively less studied. However, recent studies pertaining to the impact of recommender systems on product variety and sales concentration patterns has renewed the interests on this topic (Brynjolfsson, Hu, & Simester, 2011; Oestreicher-Singer & Sundararajan, 2012; Brynjolfsson et al., 2010) . The sales concentration patterns on e-commerce platforms reveal a "long-tail" pattern which is contrary to the Pareto Principle observed in brick and mortar sales platforms. Pareto Principle, also known as the 80/20 rule, describes the phenomenon of sales concentration on brick and mortar platforms. This rule states that a small volume (e.g., 20%) of products in a market often generates a large proportion (e.g., 80%) of sales in the market. However, with the advent of information technology, the digital marketplace can shift this balance. This shift in balance can be due to the lowering of search costs and availability of a large number of items on the e-commerce platform (Brynjolfsson et al., 2011). This may increase the combined share of obscure/niche products, and flatten the sales distribution. The phenomenon that niche products can make up a significant share of total sales gives way to — "The Long Tail"— pattern and therefore it is believed that the Pareto Principle may give way to the "Long Tail" on the e-commerce platforms.

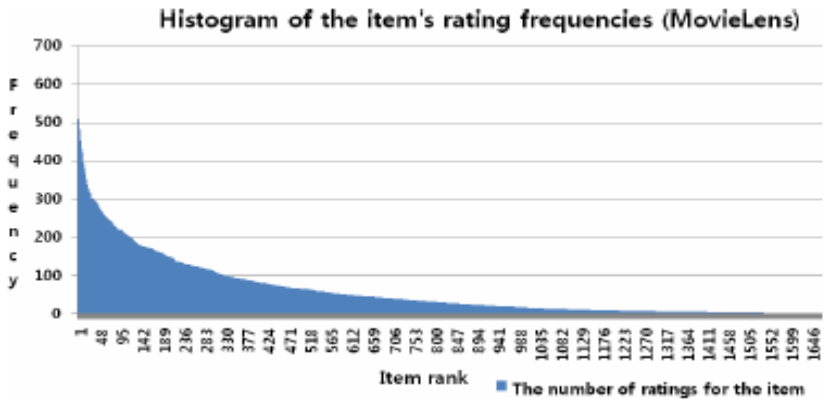


Figure 1: The long tail items

Previous research has focused on examining the impact of the internet on sales concentration on the one hand, while on the other, researchers have also studied how to design recommender systems to take advantage of the long-tail phenomenon. This can be done by intentionally generating recommendations of more niche items across all users and, in turn, increasing sales of long-tail items (Park & Tuzhilin, 2008). The detailed discussions on the various designs of recommender system catering to diversity are presented in chapter 5.

Novelty is another metric related to the concept of diversity in the context of RS that adds value to the recommendation list. The novelty as defined in RS categorizes how different is the experience of items in the recommendation list with respect to what has been previously experienced by a specific user or by a community as a whole (Castells, Vargas, & Wang, 2011). Diversity is defined over a set of items, and measures how different the items are with respect to each other. So, the concept of diversity is related to novelty in a sense that when a set of items is diverse, each item is “novel” with respect to the rest of the set. In summary, novelty, diversity, and long-tail are different yet related concepts.

2.3.1 Research gap 2

Based on the above literature surveys, a few interesting insights are found which will be used as the building blocks for the proposed model.

The first key insight is about novelty in a recommendation list, where long-tail items, which few users have accessed, is a common way in which novelty is understood (Vargas & Castells, 2011). Therefore, novelty can be taken as a surrogate measure for long-tail, which has also been the common practice. This measure is improved by considering the time dependent aspect of measuring novelty. In other words, novelty is a time dependent concept, as with the passage of time a novel item may turn up to be a popular item owing to increased user and item base. Introducing time dependent modelling of novelty is an important contribution to the present literature on long-tail RS research.

The second key insight is the relationship between the novelty and diversity of a recommendation list. Novelty and diversity are different, though related notions. The novelty of an item refers to how different it is with respect to "what has previously been seen" by a specific user, or by a community. Diversity generally applies to a set of items, and is associated to variances among the items, with respect to each other. This is related to novelty in the sense that a set of diversity results in each item within that set being "novel" with respect to the rest of the set. Additionally, a system that promotes novel recommendations tends to generate global diversity over time in the user experience and also enhances the global "diversity of sales" from the system perspective (Vargas & Castells, 2011).

The third important insight is that the above-mentioned algorithms have assumed the uniform response of every user, irrespective of his liking on the aspect of novelty and diversity. However, it would be naïve to assume that each user will have the same response towards novel and/or diverse items. It is worth mentioning two empirical studies to

corroborate the above finding. In one of the empirical studies it is found that personality has an impact on the choice of diverse products (Chen, Wu, & He, 2013). More specifically, Chen, Wu, and He (2013) established significant correlations between some personality values and the diversity of items. For instance, the "director" is significantly positively correlated with the personality factor neuroticism, which suggests that the more reactive, excited, and nervous person is more inclined to choose diverse directors. Extending the concept of variation of diversity with change in the personality factor, one can safely infer that this variation also holds true at the level of an individual. The extension of this concept is vital for the proposed model as the proposed model considers the variation of taste for novel/diverse items at the individual level and not at the personality level. This extension is also validated by the second empirical study. The second empirical study considers diverse domains—movies, music, Web search, and Web browsing. The finding appears to suggest that some users draw disproportionately from the head (popular) while others draw disproportionately from the tail(novel) (Goel et al., 2010).Thus, the above findings conclude that indeed, different users accord different weightages for novelty in e-commerce platforms. The proposed models are based on the above identified notions of individual preferences for long-tail and popular items.

2.4 Context aware Recommender systems

The importance of contextual information and its incorporation has been studied and recognized by researchers and practitioners in many disciplines, including e-commerce personalization, information retrieval, ubiquitous and mobile computing, data mining, marketing, and management. The impact of additional contextual information such as time, location, or the company of other people (e.g., in watching movies) on recommending most relevant items through various approaches has been the recent interest.

A well-known business researcher and practitioner, C. K. Prahalad stated about the context and its relevance, that “the ability to reach out and touch customers anywhere at any time means that companies must deliver not just competitive products but also unique, real-time customer experiences shaped by customer context” (2004). Context has been studied in multiple disciplines; each discipline tends to take its own distinctive view that is different from others.

Data Mining: context has been defined as any information that portrays the situation of an entity where the entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including user and applications themselves (Dey, 2001). It is associated with the life event of an individual and his preferences may change with context. Examples of context include a new job, the birth of a child, marriage, divorce, and retirement.

Marketing: According to marketing researchers, the purchasing process is directly related to the context in which the transaction takes place (Adomavicius, Sankaranarayanan, Sen, & Tuzhilin, 2005). The same customer can adopt different decision strategies and prefer different products or brands, depending on the context (Lussier & Olshavsky, 1979). Accordingly, context is defined as a task complexity in the brand choice strategy (Lussier & Olshavsky, 1979). The context is defined by Prahalad (2004) as “the precise physical location of a customer at any given time, the exact minute he or she needs the service, and the kind of technological mobile device over which that experience will be received.”

Web personalization: In the context of web personalization, it has been demonstrated that contextual information improves the recommendation quality (Adomavicius et al., 2005; Baltrunas, Kaminskas, et al., 2011; Kwon, 2006; Larson, 2010; Palmisano, Tuzhilin, & Gorgoglione, 2008). One of the earlier works on context- aware systems took intent of

purchase by a customer on an e-commerce platform as contextual information (Palmisano et al., 2008). Similarly, another work took additional contextual dimensions (such as time, companion, and weather) to be incorporated into the recommendation process and used support vector machines to provide recommendations in a restaurant recommender system (Hosseini-Pozveh, Nematbakhsh, & Movahhedinia, 2009). Recent works in context- aware systems have described the activity of users as context (studying, running, songs) and developed algorithms for music recommendation systems (Wang, Rosenblum, & Wang, 2012). The role of emotions as a contextual variable and its competition with other contextual variables such as location and time has also been studied in one of the recent works (Zheng, Mobasher, & Burke, 2014).

The definition of context from the above related fields provides the multifaceted concept of context in the real world. The context defined above has been modelled in context- aware recommender systems (CARSs) which has in turn, facilitated better personalization. The basic approaches that are followed in CARSs can be divided mainly in three: 1) contextual pre-filtering, 2) contextual post-filtering, and 3) simultaneous modeling of context and user-item interaction (Champiri, Shahamiri, & Salim, 2015). As the names suggest, in contextual pre-filtering, based on the context of interest, the data is pre-processed before applying the recommender. In contextual post filtering, the recommender system is applied irrespective of context and then contexts are post-processed. In the third approach, the interaction between user, item, and contexts are simultaneously modelled to process the recommendation system.

2.4.1 Research Gap 3

The major challenges that arise in modelling the contexts for a recommendation system are a) complexity arising due to multi-dimensional contexts often results in higher dimension

modelling, and b) the sparsity problem arises due to the unavailability of unique context for each user and item interaction.

This dissertation seeks to solve both these problems and demonstrate better personalization in the case of contexts as added dimensions.

2.5 Objectives of the dissertation

The literature gaps enlisted the above leads for following three objectives in this dissertation.

- The latent factor model predicts the ratings out of bounds which can be solved by the proposed 'cosine based latent factor model'. The key idea is to learn latent features of each user and item based on the user's liking of a product. The key concept is that the higher the cosine similarity between features of users and products, the higher will be his rating for the product and vice versa. This is a unique approach in solving the problem of generating a good recommendation model where top-k items in the recommendation are emphasized. Further, the proposed model can also be extended to the ranking task. In the ranking task, the idea is to predict the correct ranks of items in a list so that higher rank implies better recommendations than do the items at the bottom of the list. Summing up the first objective, this work proposes a 'cosine based latent factor model' that is applicable in learning parameters of the model for rating prediction task, classification task, and ranking task.
- The second research gap relates to problems in recommending diverse yet accurate sets of items. It is revealed from literature that previous researches have not captured the extent of interest (EOI) of a user towards a diverse set of items while recommending a novel/diverse set of items. The proposed model makes a key contribution by developing an algorithm that captures the EOI of every user for long-tail items which is then used

for the recommending task. A stratified learning technique of the latent factor model has been proposed for learning this formulation, which not only improves personalization but also recommends the ‘long tail’ item to idiosyncratic users.

- Finally, the third research gap talks about modelling contexts such as time, place, weather, and mood as variables for recommendation. The objective of the third kind of work is to propose a model with its ability to capture a ‘user-item-context’ interaction with lesser number of parameters, thereby reducing its complexity. It is important to understand the accuracy and its ability to handle sparse datasets should not be compromised as a result of the reduction in complexity.

3 Cosine based latent factor for precision oriented recommendations

3.1 Introduction

The majority of the models that are developed in recommendation systems cater to addressing the problem of rating prediction (Bobadilla et al., 2013). Rating prediction implies that, given the database of users and items with corresponding ratings, it becomes problematic to predict the rating of unseen items. However, in practical scenarios, the recommendation engine generates a list of top-k items to a user based on his prior interactions and interactions of other users in the database (Ning & Karypis, 2011). In a typical RS, the predicted ratings are sorted in descending order and then the user is provided with top-k items from the sorted list (Polytechnique, Lausanne; Aberer, 2014). Therefore, in summary, such kind of an RS model first solves the rating prediction problem and then sorts the rating to generate recommendation. This chapter relates to generating the top-k recommendation list in line with the first part of the first objective.

This work argues that instead of solving the rating prediction problem, the problem that needs to be solved is the generation of top-k recommendation items in the list. In order to quantify the measure of top-k items in the recommendation list for accurate recommendation, precision can be utilized as one such measure (Bellogin, Castells, & Cantador, 2011). Precision quantifies the actual number of good items recommended from the RS. Precision is defined as the ratio of good items recommended to the total number of items recommended by the RS model (Cremonesi et al., 2010). The difference between rating prediction and precision oriented RS can be understood through the following illustration: The rating prediction problem seeks to minimize the error between actual and predicted rating of the items while the precision oriented RS seeks to generate good items in the top-k list for

recommendation. Consider the items I_1, I_2…. I_9 rated by a user and if the rating prediction models are used to generate recommendations then the two possible scenarios could be the following:

Table 3: Toy example 1

Items	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	SSE
Actual rating	5	4	3	5	1	2	5	3	2	
Predicted rating-1	4.5	4.6	3.9	3.9	1.3	1.5	3.7	1.8	2.1	6.1
Predicted rating-2	4.7	3.8	2.3	4.9	1.1	3.2	4.8	1	1.9	6.1

Both the rating prediction models yield the same sum of square of errors (SSE); however, if a recommendation engine generates the recommendation list for the top-3 items, the items in the recommended list will be different. Model 1 will predict top-3 items after sorting them in descending order as I_2, I_1 and I_3 while model 2 will predict items I_4, I_7 and I_1 in the top-3 recommendation list. If one compares the actual rating, model 2 is able to predict top-3 items in the recommendation list better than model 1, yet the SSE for both the models are the same. With the above illustration, it has been shown, that for practical scenarios, solving the rating prediction problem may not be the right strategy. This work intends to solve this problem by suitably forming the associated cost function that takes care of modelling accurate top-k items in the recommendation list. This class of solution lies in the domain of model based collaborative filtering (CF).

CF techniques use a database of preference for items (e.g., movies, songs, books, travel destinations) by users to predict additional items that may be of interest to them (Su & Khoshgoftaar, 2009). In a typical CF database, there is a list of users (say *m* users) and items (say *n* items) where each user either explicitly (typically obtained by extracting users' preferences in form of star rating) or implicitly (typically obtained by monitoring purchase history, browsing history, or even mouse clicks) indicate their preferences corresponding to items (Koren, 2008). Since every user cannot consider every item when there are millions of

items in the e-commerce setup, therefore the preferences are also not available for most of the user-item pairs. In order to generate a recommendation list, an active user can be recommended items with the help of other users who have indicated similar preferences for items in the CF database.

Since this work focuses on the 'top-k' recommendation instead of the rating prediction, formulation of the problem is transformed to a classification problem, where the task is to classify the good and bad items. Based on the above arguments, this chapter has proposed an innovative algorithm which is a fusion of similarity concepts and the latent factor model. The latent factors of user and items are learnt based on the degree of similarity between user and item. The assumption of the proposed model is that the more the similarity between user and items, the probability of liking the item by the user would be higher, and vice-versa. Experiments for validating the effectiveness of the proposed approach were conducted using benchmark datasets in RS.

The rest of this chapter is organized as follows: Section 3.2 reviews previous studies regarding latent factor models as implemented in recommendation systems. Section 3.3 describes the problem at hand in a formal manner. Section 3.4 describes the proposed model with pseudo code. The next section explains how the proposed approach makes an improvement over a state-of-the-art model. Lastly, the conclusion is drawn based on the observations.

3.2 Related work

It is often difficult to incorporate memory based CF techniques in RS successfully since the user-item matrix is often sparse due to the unavailability of feedback from most of the users for most of the items.

One of the methods to deal with the problem of sparseness is by adapting a model based approach in CF. SVD is used in model based CF which reduces the dimensionality of the user-item matrix and identifies the latent factor in the data (Goldberg, Roeder, Gupta, & Perkins, 2000). An application of SVD in the context of information retrieval has already been patented and is named as Latent Semantic Indexing (LSI).

Some of the early works in RS have been adopted with appropriate modifications by applying SVD, which are different from the applications in information retrieval. Billsus and Pazzani have described the CF algorithm as a classification problem (1998). At first, the sparse user item matrix is converted to the Boolean feature matrix for every user, based on items rated by the user. Subsequently, the Boolean feature matrix is decomposed using SVD by taking ‘k’ number of dimensions to be retained. The neural network is initially used to train the singular vectors, and thereafter for prediction (Billsus & Pazzani, 1998). Since the method described is a bit complex and not scalable for real time recommendations, Sarwar, Karypis, Konstan, and Riedl describe the methodology of SVD which is directly applied in RS (2000b). The sparse user-item matrix must be filled up by either user average rating or item average rating. After this pre-processing step, the SVD is applied on the resultant populated matrix. SVD decomposes the matrix into two matrices, two of which are orthogonal matrices and one is a diagonal or singular matrix.

The user-item matrix must be imputed (assigned value) at the first step before proceeding to SVD, which has been criticized by researchers in the field as imputations lead to over-generalization and the accuracy of the method is lost. However, the start of SVD was remarkable in the context of the recommender system and it solved the problem of sparsity to an extent (Sarwar et al., 2002). But this gave rise to a different set of problems as invariably happens in a very large data set, which is often the case in the real world: the complexity and

computation of the user-item matrix increases exponentially with the increasing user- item dataset. There is also a need to update the recommendation real time to get the most accurate recommendation. The complexity and computation time problem of an SVD can be resolved by following a technique proposed by the authors, known as folding-in in SVD (Sarwar et al., 2002).

However, it was only until the Netflix Prize (Netflix, 2006) that the SVD approach was accepted as the best single method in RS. Simon Funk popularized the regularized SVD method for the first time to explore the Netflix Prize data in order to make an accurate prediction (Paterek, 2007). Subsequently, modification to the basic regularized SVD was proposed for the Netflix Prize dataset. The top prize winner in the Netflix Prize, stressed on augmenting the basic SVD with a popular neighborhood based technique (Koren, 2009). Koren also suggested incorporating implicit as well as explicit feedback in the same model for the best prediction which was being evaluated on RMSE (2008).

Singular value decomposition is a method that has also been incorporated along with other available features of the dataset to accurately predict the ratings in case of a movie recommender system. SVD combined with demographic data is also proposed to improve the approach of collaborative filtering. The reason for using demographic data along with SVD is to supplement the collaborative filtering algorithm (Vozalis & Margaritis, 2007).

While deterministic latent factor models such as SVD have been successfully implemented and made popular, probabilistic latent factor models were also considered in information retrieval, and subsequently in RS. Thomas Hofmann, in his research, utilized the statistical base as a primary reason for using probabilistic latent semantic analysis (pLSA) (2004).

Since pLSA has a drawback that an exact estimation of the ratings is intractable, this implies that potentially slow or inaccurate approximations are required for computing the posterior distribution over hidden factors in such a model (Salakhutdinov & Mnih, 2011). A full Bayesian analysis of the model was done later in 2008 by the same authors and called a probabilistic matrix factorization (pmf) to overcome the problem of inaccurate prediction. The model can be viewed as a probabilistic extension of SVD. Using the Markov Chain Monte Carlo (MCMC), pmf training is also done to avoid tuning of parameters manually, which is required to avoid over-fitting (Salakhutdinov & Mnih, 2008).

A relatively similar approach to pLSA is Latent Dirichlet Allocation (LDA). Latent Dirichlet Allocation (LDA) is similar to pLSA in the sense that latent variables are present in a probabilistic way. While pLSA does not assume a specific prior distribution over the number of dimensions in a hidden variable, LDA assumes that priors have the form of the Dirichlet distribution (Blei et al., 2003). Gibbs sampling is used to estimate the parameters in the LDA model (Krestel, Fankhauser, & Nejdl, 2009).The Expectation Maximization (EM) algorithm and its variation can also be used in solving the parameters of the model.

Continuing with the matrix factorization method to discover latent factor models, there are other approaches as well, which have been used in the field of RS. One more way of utilizing the matrix factorization so that sparse data can be handled more effectively is a model named Eigentaste that uses a principal component analysis for optimal dimensionality reduction and then clusters users in the lower dimensional subspace. As these are model based collaborative filters, they are operated in two modes: online and offline mode. The online mode uses Eigen Vectors to project new users into clusters and uses a lookup table to recommend appropriate items so that run- time is independent of the number of users in the database (Goldberg et al., 2001).

Matrix factorization is not the only way to handle latent factor models. Discrete Wavelet Transform (DWT) has earlier been used for data reduction without any deterioration in signal and image processing. Influenced by its technique in handling sparse data, DWT has also been used in RS. The technique illustrated is a unique way applied for data reduction in RS to the best of our knowledge. The argument presented by the authors based on previous research illustrates that PCA and SVD find feature combinations that model the largest contributions in a dataset, but these may not be the same features that differentiate attributes, as weaker relationships may be lost (Russell & Yoon, 2008).

Restricted Boltzmann Machine (RBM) has also been used in order to solve the sparse and large data set such as that of Netflix (Salakhutdinov et al., 2007). RBM introduced for learning the Netflix dataset used a class of two-layer undirected graphical models, suitable for modeling tabular or count data, and presented efficient learning and inference procedures for this class of models.

This sums up the related work that advocated the use of latent factor models in RS. Since the previous models based on latent factors are guided by the loss function that optimizes the actual ratings; they couldn't quite assist in 'top-k' prediction tasks. To build a model that can handle the prediction of 'top-k' good items to every user, this chapter proposes a loss function based on cosine similarity between user and item latent features.

3.3 Problem definition

In a typical E-commerce setup, there are millions of users and thousands of products listed in a database. The user specifically searches for products which he is willing to purchase; with each transaction of a user one can build his purchase history and behavior and so forth. The structure of a user's preference based on purchase history is termed as implicit feedback. Also, a user may show his explicit preference for a product by providing ratings,

viz., 1 to 5 stars. These explicit preferences for a user-item pair are termed as explicit feedback. Based on the feedback, the user-item matrix is obtained, consisting of rows representing the users, columns representing the items, and elements of the matrix are ratings of the user for an item.

Practically, not all users may show their preferences for all the items either implicitly or explicitly, which gives rise to sparsity in a user-item matrix. This poses a challenge in the recommendation task. To model such practical scenarios in research the proposed models are tested on the Movie Lens (ml100k) data set and the FilmTrust dataset. The dataset is publicly available for research and has been used in many research chapters dealing with the recommender system (Bobadilla et al., 2013). The proposed model first learns the latent features of users and items using cosine similarity as loss function, and later the score of the unseen items for a user is generated. Top 'k' items based on predicted scores can be recommended to a user in descending order of the predicted score.

3.3.1 Notations

For distinguishing users from items special indexing letters have been used for user and items – a user is denoted by u, and an item is denoted by i. A rating r_{ui} indicates the preference of a user u for an item i, where high values mean stronger preference and low values mean low preference or no preference for an item i. For example, in a range of "1 star" to "5 stars", "1 star" rating means lower interest by a user u for a given item i and "5 stars" rating means high interest by user u for a given item i. A mapped score s is obtained from the ratings by passing it through a suitable function as described in the next section. The parameters P_{uk} and Q_{ik} denote the user and item features respectively and are in the form of a vector.

3.4 Model development

This section will first introduce the basics of RSVD model, which has been state-of-the-art model, and then cover the model building phase of the proposed algorithm for the classification of good and bad items. The algorithm is primarily built for the classification task but can also be extended for the rating prediction task.

3.4.1 Basics of matrix factorization (RSVD Model)

Matrix factorization (MF) is a state-of-the-art algorithm in RS which has been made popular by its successful implementation in the Netflix Prize. Matrix factorization is the technique of decomposing a matrix into several matrices depending upon the suitability of the context. In this section, the basics of regularized singular value decomposition (RSVD), a variant of MF, has been introduced. The beauty of RSVD over other collaborative filtering algorithms is its ability of handling sparser matrices. The basic idea incorporated in RSVD is that users and items may be described by their latent features. Every item can be associated with a feature vector (Q_{ik}) which describes the type of product e.g., convenience vs. specialty, durable vs. non-durable, etc. Similarly, every user is associated with a corresponding feature vector (P_{uk}). The inner product between user feature vector and item feature vector is approximated as the predicted rating given by a user u for an item i. Mathematically, it can be expressed as:

$$\hat{r}_{ui} \approx P_{uk} Q_{ik}^{T} \tag{3.1}$$

Along with latent features, the inherent characteristics of user and item also contribute to the rating of an item by a user. For example, a functional product may always be rated a higher than average rating while an innovative product may always be rated below average. Similarly, a demanding user tends to rate on the lower side while a docile user may rate on the higher side. These inconsistencies are called biases of user and item and have to be captured in a

model. Therefore, user bias (b_u) and item bias (b_i) are added to the model, as given in equation (5.1). User bias (b_u) is the observed deviation of a user u from the average rating of all users. Item bias (b_i) is the observed deviation of item i from the average rating for all items. An overall mean of ratings represented by μ is also added to the model to capture the bias of the dataset. The resulting model is given by following equation.

$$\min_{P_* Q_*} \sum_{(u,i \in \kappa)} \left(r_{ui} - \mu - b_u - b_i - P_{uk} Q_{ik}^T\right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro} + \|b_u\|^2 + \|b_i\|^2) \quad (3.2)$$

Here, κ is the set of known ratings in matrix R and $\|\cdot\|_{Fro}$ denotes the Frobenius norm. P_* and Q_* are the shortened notations of P_{uk} and Q_{ik} that needs to be optimized for each user and for each item respectively. The parameter $\lambda > 0$ is a regularization parameter and is used to avoid over fitting of the model. Over fitting is a common term in a machine learning algorithm which suggests that the model will perform very well in the training set but will perform very badly in a test set on which the model is not trained. Training of the model is done only on the ratings available in the user-item matrix while the missing values in the matrix are skipped during training.

3.4.2 Cosine based latent factor model

There are a few disadvantages of using matrix factorization to learn the latent factors of users and items. One of the disadvantages is that the function is not bounded; hence there is a possibility of obtaining the predicted values out of range (Salakhutdinov & Mnih, 2011). Since the predicted values may get out of range, they are clipped (Paterek, 2007) or passed through a bounded function such as a logistic function (Salakhutdinov & Mnih, 2011). This may not be appropriate since the mapping function of the actual rating in a training set of data is not mapped according to the bounded function and is generally normalized (Salakhutdinov & Mnih, 2011).

To furnish a scientific solution to this problem this work introduces a cosine based latent factor model. The cosine function is bounded between -1 and 1 which gives an advantage to map the actual ratings in a training set using the cosine function and uses the cosine latent factor model to learn the features of users and items.

The intuition behind applying latent factor models such as regularized SVD is that the interaction between user and item features results in ratings of an item by a user (Koren & Bell, 2011). In the proposed cosine latent factor model intuition is the degree of similarity between user and item features and defines the interest of a user for an item. So, if a user is highly interested in an item the similarity between the user and item features is close to 1, otherwise, the similarity is closer to 0. To map the actual ratings $R_{ij} \in \{1 \ldots . r\}$ is in between 0 and 1 it is passed over a function $\emptyset$. This function should be defined such that the minimum rating shall be close to 0 and maximum rating shall be close to 1. One such function is defined below:

$$\emptyset = \frac{e^{(r-r_{mean})}}{e^{-(r-r_{mean})}+e^{(r-r_{mean})}} \tag{3.3}$$

where, r_{mean} is the average of $\{1 \ldots . r\}$. The ratings are passed through this function to obtain a mapped score s. For obtaining the latent factors of each item and user the proposed cosine latent factor model is set equivalent to the obtained mapped scores (s). This leads to minimizing the following objective function :

$$f = \min_{P_* Q_*} \sum_{(u,i \in \kappa)} \left(s_{ui} - \frac{P_{uk} Q_{ik}^T}{\|P_{uk}\| \|Q_{ik}\|} \right)^2 + \lambda (\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro} \tag{3.4}$$

Here, $\|.\|_{Fro}$ denotes the Frobenius norm.

A regularization parameter λ is introduced to create a balance between over-fitting and variance. The optimum value of the minimization function can be obtained by using the stochastic gradient descent method. For every iteration, the learning rate (α) is multiplied against the slope of descent of the function to reach local minima. The partial derivatives with respect to P_{uk} and Q_{ik} result in the gradient of descent for this function.

Model learning

Algorithm

Input:

- R : A matrix of rating, dimension N x M (user item rating matrix)
- κ : Set of known ratings in matrix R
- P_{uk} : An initial vector of dimension 1 x F (User feature vector)
- Q_{ik} : An initial vector of dimension 1 x F (item feature vector)
- F: Number of latent features to be trained
- s_{ij} : mapped scores obtained after passing through function $\emptyset$

Parameters:

- α_1: learning rate
- λ: over fitting regularization parameter
- Steps : Number of iterations

Output: A matrix with scores to generate a recommendation list

Method:

1) Initialize random values to user feature and item feature vector with a fixed number of features F
2) Fix values of α_1, and λ.
3) Continue till error converges [error(step-1) – error(step) < ε]

$$\text{error (step)} = \min_{P_* Q_*} \sum_{(u,i \in \kappa)} \left(s_{ui} - \frac{P_{uk} Q_{ik}^T}{\|P_{uk}\| \|Q_{ik}\|} \right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro}$$

for each R ∈ κ

Update training parameters

Update $P_{uk} \leftarrow P_{uk} - \alpha 1 \, \Delta P_{uk}$, $Q_{ik} \leftarrow Q_{ik} - \alpha 1 \, \Delta Q_{ik}$;

end for

4) return P_{uk}, Q_{ik}

The obtained P_{uk}, Q_{ik} for each user and items are used to predict a score using the following equation:

$$\widehat{s_{ui}} = \frac{P_{uk} Q_{ik}^T}{\|P_{uk}\| \|Q_{ik}\|} \quad (3.5)$$

Based on the above equation, the obtained scores are arranged in descending order and top 'k' scores of items for each user are truncated to generate the top-k recommendation list.

3.4.3 Extension to rating prediction

The above model generates a precision oriented recommendation list, which means the precision of the proposed model is expected to be better than other state-of-the art models. It is also desirable sometimes to provide the predicted rating along with the recommendation list. The rating prediction model seeks to optimize the ratings for generating the predicted

rating as well as for recommendation. The proposed model differs from the rating prediction model as it generates a recommendation list, and thereafter can be used to predict ratings.

The steps to be followed for the rating prediction from the proposed model are given below:

1. Identify the target user-item dyad to be predicted
2. Calculate the similarity between item's features that are rated by target user and the target item
3. Average/ weighted average: the most similar 'top-n' rated items by the target user
4. The obtained average is the rating prediction of the target item.

It is to be noted that the procedure for obtaining the predicted rating is similar to item based collaborative filtering (IBCF), except that the latent features of the item are used here to obtain the similarity, whereas in IBCF rating the vector by every user is taken for the similarity calculation. Since the rating vector can be very sparse in certain scenarios, the use of latent features obtained after optimization provides a viable solution. It is expected that the obtained predicted ratings using the above method would outperform the IBCF when the rating vector is very sparse. It also expected that the above procedure would be as good as a state-of-the-art model for rating prediction, RSVD, for datasets with various sparsity levels.

3.5 Experimentation and evaluation

In this section, the experimental setup and evaluation protocol to test the proposed model on two publicly available datasets have been presented. The proposed model is compared with baseline and other state-of-the-art algorithms. The proposed algorithms are evaluated both on classification and rating predictions, using appropriate performance measures.

Datasets

Two different datasets are used for the experimental evaluations of the proposed method. The first one is a publicly available Movie Lens dataset (ml-100k). The dataset consists of ratings of movies provided by users with corresponding user and movie IDs. There are 943 users and 1682 movies with 100000 ratings in the dataset. Had every user rated every movie, the total ratings available should have been 1586126 (i.e. 943×1682); however, only 100000 ratings are available, which means that not every user has rated every movie and the dataset is very sparse (93.7%). This dataset resembles an actual scenario in e-commerce, where not every user explicitly or implicitly expresses preferences for every item.

The second dataset consists of movie reviews from Film Trust (Guo, Zhang, & Yorke-Smith, 2013). There are 1508 users and 2071 movies with only 35497 ratings. The sparsity levels (98.86%) are more than the Movie Lens dataset.

3.5.1 Cross-Validation

The dataset is partitioned into 5 equal disjoint sets with 4 datasets used for training and one left out dataset for testing the model. The process is repeated five times as a procedure adopted for 5-fold cross-validation. On testing the dataset, the accuracy measures such as RMSE and precision are calculated and averaged over the 5-folds which is a procedure adopted to nullify the effect of the biasness of partitioning the sample.

3.5.2 Performance Metrics

In a classification task, the performance metric that determines the top ‘k’ as used in recommendation systems is:

Precision: Precision is defined as the ratio of relevant items, Nrs, recommended to the total number of items, Ns, recommended to a user.

$$\text{Precision} = \text{Nrs} / \text{Ns}$$

In the rating prediction task, the goal is to minimize the difference in ratings between predicted and actual ratings. RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) are popular metrics used for evaluating accuracy. Variants of RMSE and MAE such as Normalized RMSE and Normalized MAE or average RMSE and average MAE are also used. The predicted ratings ($\hat{r}_{ui}$) for a test set 'Γ' of user-item pairs (u, i) for which true item ratings (r_{ui}) are known, the RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|\Gamma|} \sum_{(u,i \in \Gamma)} (\hat{r}_{ui} - r_{ui})^2}$$

MAE on the other hand is given by

$$\text{MAE} = \frac{1}{|\Gamma|} \sum_{(u,i \in \Gamma)} |\hat{r}_{ui} - r_{ui}|$$

3.5.3 Evaluating the performance of models

In this section, the performances of the proposed model with already existing state-of-the-art algorithm in this field are compared and evaluated. One of the state-of-the-art algorithms is RSVD that was designed primarily for the rating prediction task. For experimentation purposes, the number of latent features (F) is varied from 10 to 100 in steps of 10 for the proposed model and RSVD algorithm. Firstly, the focus is on the classification task where the idea is to present top-k items to each user. Based on the obtained predicted ratings, in the case of RSVD and obtained scores in the proposed model, top 5 and top 10 items are presented to the user. The predicted rating and predicted scores in descending order are presented to every user respectively, and then accuracy measures such as precision are obtained (Bellogin et al., 2011). To obtain precision in both the datasets a rating of 5 is considered as actual good items.

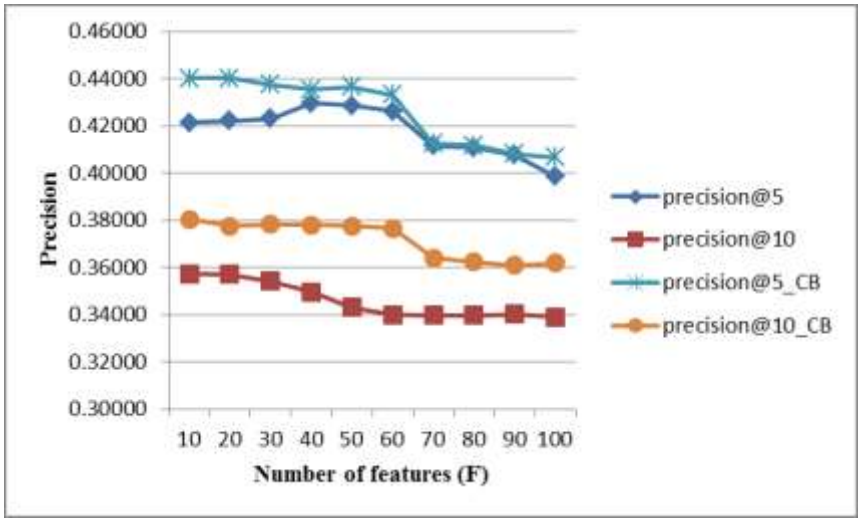


Figure 2: precision of proposed cosine latent factor model and RSVD on ml-100k dataset

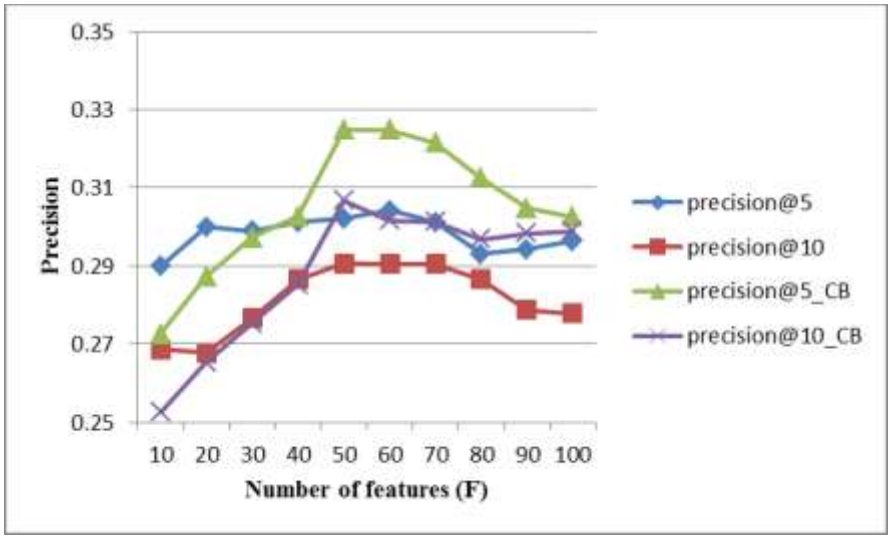


Figure 3: precision of proposed cosine latent factor model and RSVD on FilmTrust dataset

Figures 2 and 3 show the precision of the proposed cosine based latent factor model and RSVD on ml-100k dataset and Film Trust dataset, respectively. Since the precisions are computed for top 5 and top 10 items presented to each user, Precision@5 and Precision@10 are used to denote them in the figures 1 and 2. Precision@5_CB shows the precision as obtained on datasets by applying cosine based latent factor model, while precision@5 shows the precision as obtained by RSVD. In the ml-100k dataset, the highest precision for cosine based latent factor occurs when the number of features (F) for user and item are 10.

Correspondingly, the highest value of precision for RSVD occurs for F=10 but the values of precision at both top 5 and top 10 items are better for the proposed cosine latent factor model than for state-of-the-art RSVD. There is an improvement of 4.5% for precision@5 and 5% for precision@10 over RSVD algorithm on ml-100k dataset. In case of the Film Trust dataset, the maximum precision@5 for cosine based latent factor model and RSVD occurs when the F value is 60. For presicion@10, the maximum value occurs at F=50 for both the cosine based latent factor model and RSVD. Here, the cosine based latent factor model outperforms RSVD for precision@5 by approximately 6.5% and for precision@5 by approximately 5%.

In the rating prediction task, the predicted ratings obtained from both RSVD and the proposed cosine latent factor model with varying latent features (F) are compared with actual ratings in the test set using a 5-fold cross-validation. The latent features (F) in both RSVD and proposed cosine latent factor model are varied from 10 to 100 in steps of 10, the cross-validated MAE and RMSE are obtained for both the two datasets and compared.

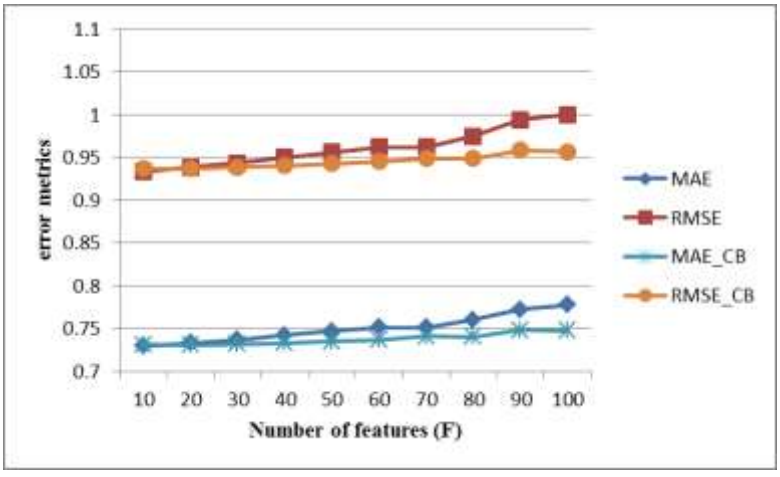


Figure 4: Error metrics of proposed cosine latent factor model and RSVD on ml-100k dataset

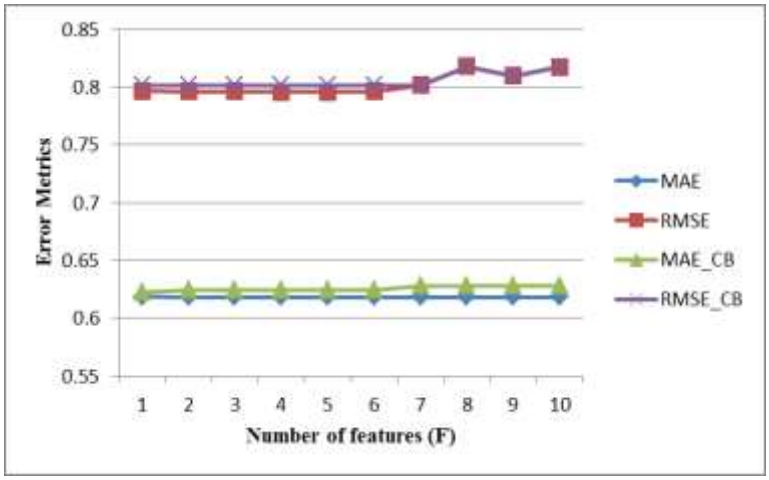


Figure 5: Error metrics of proposed cosine latent factor model and RSVD on FilmTrust dataset

From the figures 4 and 5, one can see that although MAE and RMSE for RSVD is better than the cosine based latent factor model on both the datasets, the difference is negligible for the best value obtained from both these algorithms. Note that the rating prediction task is conducted by taking the average of top-n similar items rated by the user. The use of other sophisticated techniques like weighted average can improve the rating prediction task.

Thus, this work has shown through empirical experimentation, that the proposed cosine latent factor model outperforms state-of-the-art algorithm, RSVD, in terms of precision. Also, the proposed model gives comparable results in terms of MAE and RMSE for the rating prediction task.

3.6 Discussions and conclusions

This work proposed a novel algorithm that caters to recommending top ‘k’ items for each user. It primarily falls in the domain of the model based RS with a focus on the classification of good and bad items. This work introduces the concept of the cosine similarity based latent factor model, which is a unique algorithm. Also, the rationale behind using the cosine similarity latent factor model over RSVD is theoretically sound. As mentioned in the chapter that the rating prediction using RSVD often goes out of bounds as the loss function is not

bounded, the loss function used in the proposed model is bounded, and therefore the prediction does not go out of bounds. One more advantage in using the proposed model is its ability to handle difficult (outlier) data points due to its inherent property of bounds which are not observed in an RSVD model.

In the future, this work can be taken forward to utilize the proposed cosine latent factor model in the field of information retrieval and also in the ranking prediction task for both the recommender system and information retrieval. The learning method of optimization can also be suitably modified to learn the parameters of the proposed model a bit faster. One of the other approaches of using the proposed model for the above task can be by an ensemble of weak learners which can be generated by varying the number of latent factors of the model. The present work is primarily designed for the top ‘k’ recommendation task but has also been extended to the rating prediction task by using a simple average of the ratings of most similar items that are applied. The rating prediction using more sophisticated techniques like clustering of the item features can also be obtained to check any further improvement. The techniques being applied should be carefully chosen as they may increase the complexity without improving the accuracy adequately.

4 List wise Ranking using cosine based latent factor model

4.1 Introduction

There has been a surge in clicks on e-services which has enabled many new e-commerce platforms like Alibaba, Flipkart, etc., to spring up. One of the advantages of the e-commerce platforms is their ability to offer a large number of distinct choices to customers (Adolphs & Winkelmann, 2010). With the rapid growth of customers in e-commerce and a large number of choices available to the users, it is often tough to choose the right product for their need (Kim, Yum, Song, & Kim, 2005). In order to facilitate users, the digitization of data on e-commerce and its affordable processing has enabled e-service providers to aid in the customers' decision making (Bauer & Nanopoulos, 2014; Ho, Kyeong, & Hie, 2002). Decision making is supported by means of decision support systems that collect huge data, process it, and project them to be consumed by users (Vahidov & Ji, 2005)Recommender systems are one such decision support system used in e-commerce to help navigate customers to the right product (Bauer & Nanopoulos, 2014; Cheung, Kwok, Law, & Tsui, 2003; Chung, Hsu, & Huang, 2013; Ekstrand, 2010; Gan & Jiang, 2013).

Typical recommender system makes personalized recommendations for a user, based on their prior explicit preferences. These preferences are captured implicitly (often by mouse clicks, products purchased, products placed in the basket, etc.) or explicitly in the form of "star" ratings for select items. For instance, a user may like a movie and provide "5 stars" to it (Bauer & Nanopoulos, 2014; Bell, Koren, Ave, & Park, 2007). Collaborative filtering (CF) algorithms are widely used to design recommender systems in this kind of setup (Pennock et al., 2000; Resnick et al., 1994; Kagie et al., 2009; Yu et al., 2009; Takács et al., 2009; Hung et al., 2012; du Boucher-Ryan & Bridge, 2006; Goldberg et al., 2000; Niemann & Wolpers,

2013; Liu et al., 2012). They induce unobserved user-item preferences (rating) from preferences (ratings) gathered for a target user, and the preferences (ratings) from all the other users. Recently proposed algorithms in CF take into account the collaborative effects that arise due to the inter- action of the users with the items. This is done by dynamically grouping users based on the items under consideration, and at the same time, grouping items based on the similarity of the clustering induced over the users. The resulting algorithm thus takes advantage of preference patterns in the data in a way that is parallel to collaborative filtering methods (Li, Karatzoglou, & Gentile, 2016). Another novel algorithmic approach to content recommendation is based on adaptive clustering of exploration-exploitation ("bandit") strategies (Gentile, Li, & Giovanni, 2014). A very interesting and recently proposed model suggests that a recommendation system can also been seen as solving linear bandit problems in peer to peer networks with limited communication capabilities, especially for small startups having few resources (Korda, Szörényi, & Li, 2016). After learning the parameters of machine learning and thereafter predicting ratings, it is often customary to present the ranked list of items in decreasing order of predicted ratings by CF to a target user in the form of recommendations (Burges et al., 1998; Cheung et al., 2003; Huang, Zhong, & Yao, 2014; Yang, Guo, Liu, & Steck, 2014).

In the above framework of CF, a natural way of training and testing of new algorithms on benchmark datasets has emerged (Liu & Liou, 2011; Pennock et al., 2000; Steck, Labs, Ave, & Hill, 2010; Takács et al., 2009). This paradigm of the CF algorithm is viewed as predicting the rating for an unseen user-item pair built on a database of elicited preferences (ratings) by users on items. The benchmark datasets are partitioned into disjoint training and test sets, CF models are learnt on the training set and evaluated on test sets (Gao, Liu, & Wu, 2010; Karypis, 2001; Kelleher & Bridge, 2004; Zhang, 2009). Since the recommendation task is framed as a rating prediction task for unseen user- item pairs, CF is modeled by optimizing

errors of actual and predicted rating (Bellogin, Castells, & Cantador, 2011; Ziegler, McNee, Konstan, & Lausen, 2005).

Although this has been a successful approach to build a recommender system, it is now believed that a better way of recommending items to a user can be achieved by formalizing it as a ranking task (Weimer, Karatzoglou, & Bruch, 2009). A ranking task can be illustrated by a simple example: suppose 'I1', 'I2' , 'I3' are items presented to a user 'u', and his preferences are in order of 'I2' ,'I3','I1', I2 being the most liked product and I1 being the least preferred item. This suggests that the user prefers item 'I2' over 'I3'and 'I3' over 'I1' in a ranking order. The key idea is to learn the parameters of a model appropriated for the given ranking order.

In an e-commerce setup, a customer is often provided with a top-k ranked list of items instead of a predicted rating for products in the recommendation list. Hence, to cater to this set of scenarios, models that can directly optimize ranks of the items are gradually studied in the literature of recommender systems (Balakrishnan & Chopra , 2012; Harrington, 2003; Weimer & Karatzoglou, 2007). This ranking model is often termed as 'learning to rank' in the machine learning community where it has attracted broad attention due to its importance in information retrieval and collaborative filtering (Karatzoglou, Baltrunas, & Shi, 2013; Rupnik, 2010; Valizadegan & Jin, 2009; Xia, Liu, Wang, Zhang, & Li, 2008). Different machine learning models applied in information retrieval include SVM and Neural Networks leading to the development of models like Ranking SVM (Joachims, 2002), Rank Net (Burges et al., 1998), Rank Boost (Freund, Iyer, Schapire, & Singer, 2003), and Rank Cosine (Qin et al., 2008).

Learning to order items by optimizing ranking metrics has been a practical solution to address this class of problems (Valizadegan & Jin, 2009). Normalized Discounted

Cumulative Gain (NDCG), a ranking metric, is considered a fit in the ranking task for information retrieval and collaborative filtering (Balakrishnan & Chopra , 2012; Polytechnique, Lausanne; Aberer, 2014; Taylor, Guiver, Robertson, & Minka, 2008). NDCG gives maximum scores for relevant items ranked higher than a relevant item which is ranked lower. Since top-k items are presented to users in the e-commerce setup as a recommendation list, our focus is precisely on optimizing the ranking metric. Motivated by the 'learning to rank' task from the machine learning community, we propose a list wise model of learning in the ranking task. A list wise model is considered a better approach in a 'learning to rank' problem than other approaches, viz., point wise and pairwise models (Xia et al., 2008).

A key issue in applying optimization models in a CF setting is a lack of explicit input features for training the model (del Olmo & Gaudioso, 2008; Jawaheer, Weller, & Kostkova, 2014). Another issue in a CF setting is data sparsity, mainly arising due to lack of ratings for user-item pairs, which makes training the models challenging (Anand & Bharadwaj, 2011; Deng & Wang, 2009; Liu & Liou, 2011; Liu & Yang, 2008; Ning & Karypis, 2011; Sarwar, Karypis, Konstan, & Riedl, 2000). One of the efficient ways of handling lack of explicit input features and data sparsity is by applying latent factor models in CF (Agarwal & Chen, 2009; Ahmed et al., 2013). Latent factor models learn features of users and items by optimizing the ratings in the database. However, since the task at hand is the optimization of the ranks, not ratings, suitable modification can be carried out. A traditional matrix factorization model such as RSVD, and state-of-the-art algorithm in rating prediction have some weaknesses, therefore, the cosine based latent factor model has been used to learn the features of user and items in the proposed model. This model uses the cosine similarity to learn the latent elements that describe the inherent structure of the relationship between items and users while simultaneously learning the parameters of the list wise ranking model. We claim that the proposed novel model outperforms the existing models in the ranking task. This claim is validated by running the proposed model on standard, real- world datasets.

4.2 Related work

There are mainly three approaches to address the 'learn to rank' problem, namely, point wise, pairwise, and list wise (Chen, Liu, Lan, Ma, & Li, 2009). In the 'point wise approach', optimizing the relevance level (rating) of an item is affected with respect to a user to generate the top-k recommendation list. This approach is equivalent to the rating prediction problem in collaborative filtering. The pairwise approach seeks to optimize the loss between relevance of any pair of rated items for each user. Since this approach computes the pairwise loss for all pairs of rated items, the complexity of the model is large. Another method in ranking is the list wise approach, which seeks to minimize list wise loss defined on the ranked list of predicted rating and actual ratings. It generally outperforms point wise and pairwise methods in the ranking recommendation list (Cao, Qin, Liu, Tsai, & Li, 2007).

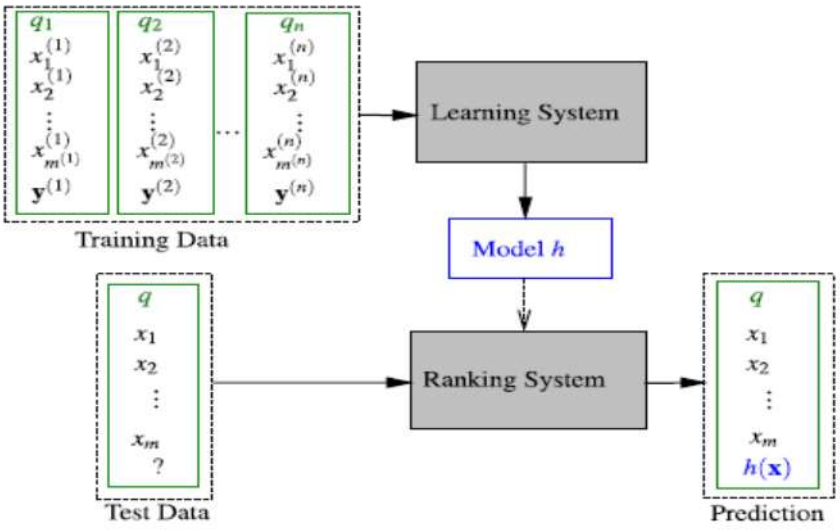


Figure 6: A typical learn to rank framework

Learn to rank models can be traced back to their application in information retrieval which have also been extended to the collaborative filtering model in recommender systems. This chapter discusses a few of these algorithms in the order of point wise, pairwise, and list wise, as applied in collaborative filtering.

A well-known algorithm in collaborative filtering that is based on ordinal regression in the point wise method of ranking is PRank (Crammer & Singer, 2002). The algorithm tries to find a direction defined by a parameter vector w, after projecting the ratings which can use thresholds to distinguish the items into ordered categories. An extension of PRank that approximates the thresholds aggregating the Bayes points, and gives a good generalization performance has also been proposed (Harrington, 2003). Continuing with the point wise approach in collaborative filtering, OrdRec proposed in (Koren & Sill, 2011) learns parameters of matrix factorization (MF) by optimizing the ordinal scores of items. In another method, Balakrishnan and Chopra proposed a two-step approach to learn the parameters of matrix factorization (2012). In the first step the parameters of MF are learnt by optimizing preferences (ratings) of items, which is a procedure in rating based methods of CF, and in the next step a regression based learning method is adopted to optimize the ranking (Balakrishnan & Chopra, 2012). It has been argued that the point wise approach does not consider the relative preference level, and therefore, is not the true reflection of the actual loss in ranking problem.

The second approach to handle the ranking problem is the pairwise consideration of items for computing the loss function. Instead of optimizing the relevance degree, as in the case of the point wise approach, the idea is to classify a pair of items at an instance and extend it to all possible pairs to learn the model parameters. RankBoost is a popular algorithm in the pairwise approach to learn ranking in collaborative filtering which optimizes item rank pairs to learn the weak classifiers by boosting (Freund et al., 2003). EigenRank solves the pairwise ranking problem by the greedy order algorithm and random walk model (Liu & Yang, 2008). The Bayesian personalized ranking (BPR) optimizes the pairwise loss function obtained from implicit feedback to rank the items (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2009). This chapter also shows the analogies to the maximization of the area under the ROC

curve. The local collaborating ranking algorithm has also been proposed in addition to the pairwise method of learning to rank in CF. This algorithm takes a pairwise 0-1 loss to learn the parameter of the local singular value method (Lee, Bengio, & Kim, 2014). The major drawback of the pairwise method is its large complexity in computing the losses; the difference in the relevant degree of losses is not accounted for and noisy relevance prediction for one item in the ranked list can create mislabelling for many items in the list (Cao et al., 2007; Xia et al., 2008).

The third approach which is gaining popularity in the learn to rank task is the list wise method of ranking. The list wise method can be further categorized into two sub-categories. The first category directly optimizes evaluation measures by an approximate or upper bound of these measures. The second category, on the other hand, optimizes the loss by revealing the inconsistencies in the actual and predicted ranks. CoFiRank is one such algorithm that falls into the first subcategory and it optimizes the upper bound of normalized discounted cumulative gain (NDCG) (Weimer & Karatzoglou, 2007). CliMF directly optimizes the lower bound of smoothed reciprocal rank (RR) which is obtained by using a logistic function (Karatzoglou, Larson, Oliver, & Hanjalic, 2012). The second category in the list wise approach of ranking learns the parameters of models such that the list of recommended items is used as ‘instances’. In one of the first approaches in this category, ‘ListNet’ introduced the concept of permutation probability and top- k probability to define list wise loss function, and neural network was applied to maximize permutation or top- k probability (Cao et al., 2007). In a similar approach of defining list wise loss function ListRank-MF defines top one probability as the loss function while the parameters of users and items are learnt using cross-entropy loss with a probabilistic matrix factorization approach (Shi, Larson, & Hanjalic, 2010a). ListPMF is a recently proposed algorithm which defines cosine based permutation probability loss function as a list wise loss and learns the parameters of probabilistic matrix

factorization to learn the latent parameters of users and items (Liu, Wu, Xiong, & Liu, 2014). The importance of list wise loss function has been recognized and is considered a better approach in ranking than point wise and pairwise methods (Chen et al., 2009; Valizadegan & Jin, 2009; Xia et al., 2008).

4.3 Problem definition

In the rating prediction task, the objective is to predict the rating of unseen items for every user, while in the ranking task the objective is to present a user with the top-k most relevant items in order. Learning to rank (LTR) is considered as a supervised learning problem, where training data 'T' consists of a pair of user and product with corresponding rating scores, T= $\{(u , i , r_{ui}) : u \epsilon [1,2,3,.....m], i \epsilon [1,2,3,.....n]\}$. The rating labels ($r_{ui}$) are typical ordinal judgments by a user (u) for a product (i) (e.g.: r_{ui} =1 means not relevant and r_{ui} =5 means most relevant). The key idea is to create a ranking function, f so as to preserve the preference order in a training set to the extent possible. Clearly, the ranking function outputs a score $f(u , i)$. A function, f is fitted such that $f(u_1, i_1) > f(u_1, i_2)$ if user u_1 prefers product i_1 to product i_2.

One way of learning the parameters of a ranking function is to sort the obtained output by directly approximating the $f(u, i) \approx r_{ui}$ for all user-item pair in training set. This type of learning is termed as point wise learning. Although this is a straight forward method of learning, it suffers due to inconsistencies of learning objective and actual output. To overcome this, a "pair-wise" approach is often considered, which learns the parameters of the model based on the order of a pair of items. This approach improves the accuracy over the point-wise model (Karatzoglou et al., 2013; Lee et al., 2014; Rendle et al., 2009). But the learning is often complex which reduces the efficiency (Lee et al., 2014; Liu et al., 2014; Shi et al., 2010a). To tackle complexity, list wise learning of the ranking function is proposed

which learns by minimizing the loss between the ranked list of output score and the ranked list of actual rating in the training dataset.

Definition1 (NDCG) Given a vector of rating $R \in \{1 \ldots . r\}$ and let a permutation of the given rating be represented by 'π', ' π_j' denotes the position of item ' j' after permutation. Additionally, let ' π_s' denote the sorted vector of rating in descending order, $k \epsilon \mathbb{N}$ be a truncation threshold. The Discounted Cumulative Gains (DCG @ k) and Normalized Discounted Gains (NDCG @ k) are given by Järvelin and Kekäläinen in 2002 (Weimer, Karatzoglou, & Bruch, 2009) .

$$\mathrm{DCG@k}(R,\pi) = \sum_{j=1}^{k} \frac{2^{R_{\pi_j}}-1}{\log(j+1)} \qquad \text{and} \qquad \mathrm{NDCG@k}(R,\pi) = \frac{\mathrm{DCG@k}(R,\pi)}{\mathrm{DCG@k}(R,\pi_s)}$$

DCG@k is maximized for $\pi = \pi s$. The truncation threshold k reflects how many recommendations users are willing to consider. NDCG is a normalized version of DCG so that the score is bounded by [0, 1].

Unlike the rating prediction task, NDCG is defined on permutations of ratings. NDCG is a position- dependent metric, where higher positions are more valuable than lower positions. Directly optimizing NDCG has gained interest in collaborative ranking tasks (Weimer & Karatzoglou, 2007; Shi et al., 2010a; Liu et al., 2014).

Since the ranks generated are piecewise constant, the NDCG corresponding to every user is a discontinuous optimization, which posits significant challenges (Weimer & Karatzoglou, 2007). Therefore, a suitable surrogate loss function that optimizes NDCG can be used (Xia et al., 2008). For list wise learning, there are a few surrogate losses proposed in the context of information retrieval systems like cross entropy loss, cosine loss and likelihood loss. Cosine and likelihood losses have been shown to possess the best properties for surrogate loss due to their low complexity, consistency, soundness, continuity, differentiability, and convexity (Xia

et al., 2008); therefore, these are considered for surrogate loss in this work. The proposed models demonstrate the adoption of the cosine loss and likelihood loss for the ranking task in recommendation systems.

4.4 Model Development

The learning to rank task has been achieved in this work by optimizing the NDCG metric. The NDCG metric has been optimized against the latent features learnt from the cosine based latent factor model. As formulated in chapter 3, the cosine based latent factor model learns the features by finding similarity between users and item latent features. This chapter uses the concept of learning features from the cosine based latent factor model, which is then fitted to optimize the ranks of items. The learn to rank problem has to be formulated with appropriate loss functions. The next section summarizes cosine based latent factor model and then deliberates on two loss functions, viz., likelihood loss and cosine permutation probability which are used for ranking task.

4.4.1 Cosine based latent factor model

Before delving into the adoption of likelihood loss and cosine loss for ranking task in recommendation system, this chapter summarizes the algorithm designed to learn the input features from the user-item rating matrix in previous chapter.

To furnish a scientific solution to address the lack of boundedness problem in RSVD loss function, the earlier chapter suggested a cosine based latent factor model. The cosine function is bounded between -1 and 1 which gives an advantage to map the actual ratings in train set using cosine function and use cosine latent factor model to learn the features of users and items. The intuition behind use of latent factor model is that interaction between user and item features result in ratings of an item by a user (Koren & Bell, 2011), while in the proposed cosine latent factor model the intuition is that the similarities between user and item

features define the interest of user for an item. Hence, if a user is highly interested in an item, the similarity between the features of user and item is close to 1and if he is not interested in an item, the similarity is close to 0. In order to map the actual ratings $R_{ij} \in \{1 \ldots . r\}$ in between 0 and 1, we use a function ∅. This function is defined such that the minimum rating lies close to 0 and maximum rating is close to 1. We have defined function $\emptyset = \frac{e^{(r-r_{mean})}}{e^{-(r-r_{mean})}+e^{(r-r_{mean})}}$, where, r_{mean} is the average of $\{1 \ldots . r\}$. The ratings are passed through this function to obtain a mapped score s. For obtaining the latent factors of each item and user the proposed cosine based latent factor model is set equivalent to obtained mapped scores (s). This leads to minimizing the following objective function.

$$f = \min_{P_* Q_*} \sum_{(u,i \in \kappa)} \left(s_{ui} - \frac{P_{uk} Q_{ik}^T}{\|P_{uk}\| \|Q_{ik}\|} \right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro}$$

Here P_{uk} and Q_{ik} are the input latent features learnt from the user-item matrix that are obtained by minimizing the difference between actual (s_{ui}) and predicted ($\widehat{s_{ui}}$) scores. These input features can be applied in learning to rank task. The learning of these latent features is shown with detail in model learning section. Once the model is learnt, the scores of unrated items by every user can be predicted according to the following equation.

$$\widehat{s_{ui}} = \frac{P_{uk} Q_{ik}^T}{\sqrt{P_{uk}^2} \sqrt{Q_{ik}^2}}$$

The predicted scores can be arranged in descending order to obtain the recommendation for every user. Nevertheless, we argue that this method can further be improved by directly optimizing the ranks of items rather than relying on rating prediction task for recommending items.

4.4.2 Likelihood Loss

Since NDCG is not directly differentiable, therefore, a concept of surrogate loss has been introduced in learning to rank problem. The likelihood loss is one such surrogate loss which is consistent, sound, differentiable, continuous, and convex. The likelihood loss can be defined on the permutation of ratings provided by a user for a list of items. Likelihood loss is defined as:

$$\theta\big(g(P_{uk}, Q_{ik}), \pi_j\big) = -\log P(\pi_j | P_{uk}, Q_{ik}; g) \tag{4.1}$$

For collaborative filtering task, $g(P_{uk}, Q_{ik})$ is denoted by predicted score $\widehat{s_{ui}}$

$$\text{Where, } P(\pi_j | \widehat{s_{ui}}; g) = \prod_{j=1}^{n} \frac{\widehat{s_{u\pi_j}}}{\sum_{k=j}^{n} \widehat{s_{u\pi_k}}} \tag{4.2}$$

Here, the above equation denotes the probability distribution over all the permutations given the predicted result by the ranking function, and defines the loss as a negative likelihood of the ground truth list. The probability distribution turns out to be the Plackett-Luce model (Xia et al., 2008). In order to learn the parameters of likelihood loss the sum of the Plackett-Luce permutation probability is maximized over all the training data. Taking this likelihood function for every user and taking the natural logarithm, equation (4.2) is reduced to minimizing the following equations:

$$L = \sum_{u=1}^{m} \sum_{j=1}^{n} \left[\ln \sum_{k=j}^{n} \big(\widehat{s_{u\pi_k}}\big) - \ln \left(\widehat{s_{u\pi_j}}\right) \right] \tag{4.3}$$

The regularization parameter is introduced in order to avoid overfitting of the above loss function (equation 4.3). The regularization parameters are often included in machine learning for a prediction task. The predicted score $(\widehat{s_{ui}})$ is replaced in terms of latent factors using equation (3.5). Therefore, the equation (4.3) now transforms to equation (4.4), which will be used to learn the latent factors of the model.

$$L = \sum_{u=1}^{m} \sum_{j=1}^{n} \left[\ln \sum_{k=j}^{n} \left(\frac{\mathbf{P_{uk}} \boldsymbol{Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2} \sqrt{\mathbf{Q_{ik}}^2}} \right) - \ln \left(\frac{\mathbf{P_{uk}} \boldsymbol{Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2} \sqrt{\mathbf{Q_{ik}}^2}} \right) \right] + \lambda_u (\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro})$$

(4.4)

4.4.3 Cosine based permutation probability

Another loss function that has been proposed to act as a surrogate loss to measure the consistency in the ranks of items is the Cosine based permutation probability (Liu et al., 2014). This permutation probability is defined as the cosine similarity between the observed rating vector and its prediction.

$$P_{cos}\left(\pi_u / \hat{r}_{ui}\right) = \frac{\sum_{i=1}^{n} I_{ui} \varphi(r_{ui}) \varphi(\hat{r}_{ui})}{\sqrt{\sum_{i=1}^{n} I_{ui} (\varphi(r_{ui}))^2} \sqrt{\sum_{i=1}^{n} I_{ui} (\varphi((\hat{r}_{ui}))^2}} \quad (4.5)$$

Where, $P_{cos}\left(\pi_u / \hat{r}_{ui}\right)$ is the probability of observing user $u's$ preference order given the prediction rating, and is called the permutation probability. I is an indicator matrix, I_{ui} is equal to 1 if user u has rated item i and 0 if user u has not rated item i. $\varphi(.)$ is a strictly increasing positive function. The larger the value of permutation probability, the more similar the predicted rating vector is to the observed rating vector.

The proposed cosine latent factor model has been used in the previous section to optimize mapped preference scores to learn the parameters of the model. But the task at hand is to optimize the ranking metrics so that a ranked list of items is generated for each user. This is achieved by using the cosine based permutation probability for computing the list wise loss and the cosine latent factor model to learn the parameters of the model. The already mapped ratings into scores of the cosine based latent factor model are applied to the cosine based permutation probability in the learn to rank problem.

$$P_{cos}\left(\pi_u/\hat{r}_{ui}\right) = \frac{\sum_{i=1}^{n} I_{ui}(r_{ui})(\hat{r}_{ui})}{\sqrt{\sum_{i=1}^{n} I_{ui}(\varphi(r_{ui}))^2}\sqrt{\sum_{i=1}^{n} I_{ui}(\varphi((\hat{r}_{ui}))^2}} \quad (4.6)$$

Accordingly, the cosine permutation probability is maximized for obtaining the best ordered preference. The equation (4.6) can be reduced to minimizing the following equation (4.7) by taking the natural logarithm and substituting the value $\hat{s}$ with the item and user features.

$$\ln[P_{cos}\left(\pi_u/\hat{r}_{ui}\right)] = \frac{1}{2}\left(\ln\left(\sum_{i=1}^{n} I_{ui}\left(\frac{\mathbf{P_{uk}Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2}\sqrt{\mathbf{Q_{ik}}^2}}\right)^2\right) + \ln(\sum_{i=1}^{n} I_{ui}(s_{ui})^2)\right) - \ln\left(\sum_{i=1}^{n}(s_i)(\frac{\mathbf{P_{uk}Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2}\sqrt{\mathbf{Q_{ik}}^2}})\right) \quad (4.7)$$

This equation (4.7) is for a user i, but can be extended for all the users in the database. The parameters $\mathbf{P_{ik}}$ and $\mathbf{Q_{jk}}$ are also regularized to avoid overfitting. Subsequently, the overall objective function has been expressed in the following equation (4.8).

This equation can be learned using the stochastic gradient descent as described in the following section.

$$L = \sum_{i=1}^{m}\left[\frac{1}{2}\left(\ln\left(\sum_{i=1}^{n} I_{ui}\left(\frac{\mathbf{P_{uk}Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2}\sqrt{\mathbf{Q_{ik}}^2}}\right)^2\right) + \ln(\sum_{i=1}^{n} I_{ui}(s_{ui})^2)\right) - \ln\left(\sum_{i=1}^{n}(s_i)(\frac{\mathbf{P_{uk}Q_{ik}^T}}{\sqrt{\mathbf{P_{uk}}^2}\sqrt{\mathbf{Q_{ik}}^2}})\right)\right] + \lambda(\|\mathbf{P_{uk}}\|^2_{Fro} + \|\mathbf{Q_{ik}}\|^2_{Fro}) \quad (4.8)$$

4.4.4 Model learning

Model learning can be achieved by performing a stochastic gradient descent (SGD). Firstly, random values between 0 and 1 can be initialized for P_i and Q_j, and then P_i and Q_j are updated by using gradients until the change of the objective function reaches a prefixed threshold. The below algorithm illustrates the leaning process of the proposed models:

Algorithm: List wise learn to rank

Input: observed rating matrix R;

Parameters: hyper-parameter λ and learning rate η;

Output: user latent feature vector $\mathbf{P_{uk}}$ **,** item latent feature vector $\mathbf{Q_{ik}}$**;**

01 Initialize $\mathbf{P_{uk}}$ and $\mathbf{Q_{ik}}$ randomly;

02 Sort the ratings of items in decreasing order for all users in training set

03 **Repeat**

04 **for** users 1 to m

05 compute $\Delta\mathbf{P_{uk}}$ and $\Delta\mathbf{Q_{ik}}$ with current $\mathbf{P_{uk}}$ and $\mathbf{Q_{ik}}$;

06 Update $\mathbf{P_{uk}} \leftarrow \mathbf{P_{uk}} - \boldsymbol{\eta}\Delta\mathbf{P_{uk}}$, $\mathbf{Q_{ik}} \leftarrow \mathbf{Q_{ik}} - \boldsymbol{\eta}\Delta\mathbf{Q_{ik}}$ **;**

07 end **for**

08 **UNTIL** (change of **L** is below a value ε)

4.5 Experimental setup and evaluation

In this section, this work presents the experimental setup and evaluation protocol to test the proposed model on three publicly available benchmark datasets. The experimentation can be divided into two parts. The first part is the evaluation of proposed algorithms with respect to baseline algorithms (rating prediction algorithms) on ranking metrics. The base line models are regularized singular value decomposition (RSVD), a state-of-the art in rating prediction, and Cosine based (CB)

latent factor model. The experimentation is carried out to study the variation of the number of latent factors (D) on the ranking metric.

The second part of experimentation is to compare the proposed ranking based algorithms with other existing ranking based algorithm on ranking metrics. List Rank MF (Shi et al., 2010), ListPMF_PL, and ListPMF_cosine (Liu et al., 2014), algorithms that optimize ranks, are used for comparison with the proposed models. The proposed algorithm and other existing ranking based algorithms are evaluated on ranking metrics, NDCG. The proposed algorithms are denoted by CB_PL for the learning to rank model using likelihood loss and CB_Cosine for learning to rank model using cosine based permutation probability.

In RS, the ranking metric is evaluated by firstly presenting top 'k' items to every user based on the decreasing order of predicted scores, and secondly, calculating ranking metrics. A similar procedure is followed for baseline algorithms, viz., RSVD and cosine based latent factor model and existing ranking algorithm. However, the algorithms for ranking task can be evaluated by presenting top 'k' items to every user based on predicted scores and arranging them in descending order.

4.5.1 Performance metrics

In the ranking task, the performance metric should be such that it takes care of position dependency of the recommended items in the recommendation list. NDCG has been widely used in recommendation systems to evaluate the graded and position dependent feedback (Weimer & Karatzoglou, 2007; Shi et al., 2010; Liu et al., 2014). The NDCG values are generally truncated to the top-k recommendation list to show the relative performances of top-k recommended items. In this chapter NDCG@5 and NDCG@10 are used to denote top 5 and top 10 recommended items in the list.

Since the focus is on the ranking task, the performance metric should be such that it takes care of position dependency of the recommended items in the recommendation list. NDCG has been widely used in recommendation systems to evaluate the graded and position

dependent feedback. The NDCG values are generally truncated to top-k recommendation list to show the relative performances of top-k recommended items. In this chapter NDCG@5 and NDCG@10 are used to denote top 5 and top 10 recommended items in the list.

Datasets

The experiments are conducted on two publicly available benchmark datasets, i.e., Movie Lens 100k and Movie Lens 1m datasets. The first dataset has 100 k ratings (scale 1-5), with 943 users and 1682 movies. The second dataset has 1m ratings (scale1-5), with 6040 users and 3952 movies. The detailed statistics of both datasets are presented in table 5.

Table 4: Statistics of Movie Lens datasets

Dataset	ML-100k	ML-1m
Number of users	943	6040
Number of items	1682	3952
Number of ratings	100000	1000209
Density	0.063	0.042

4.5.2 Results

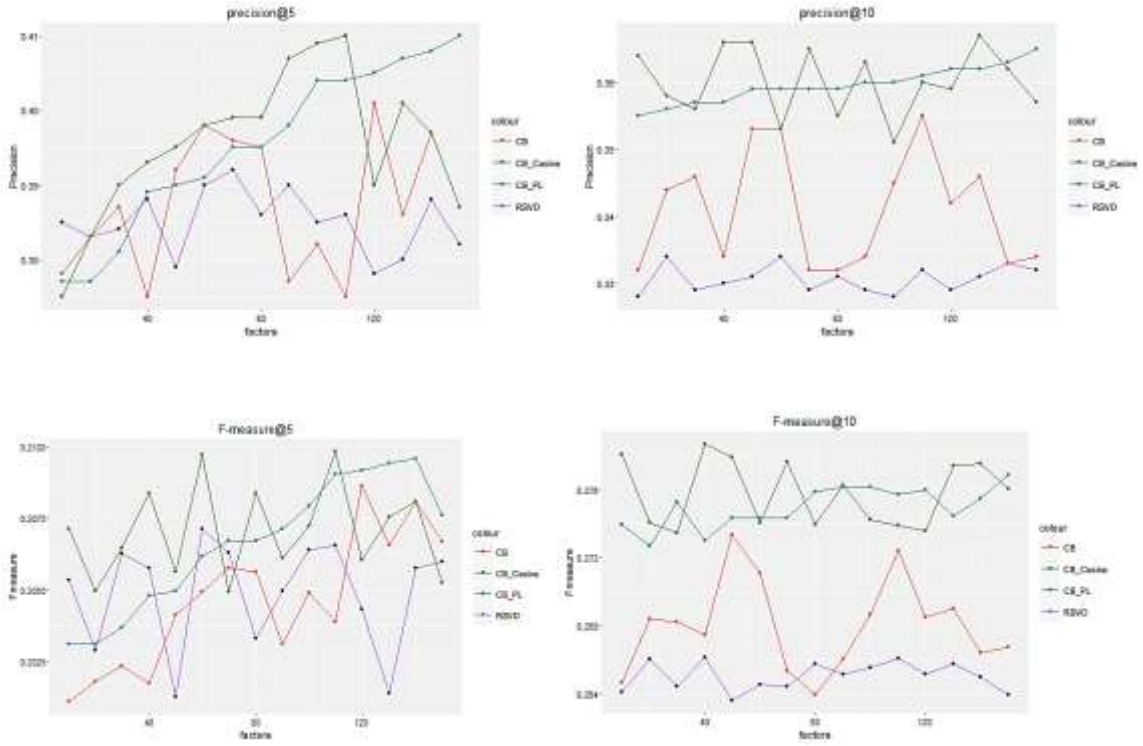
There are standard experimental setups to evaluate the performance of ranking algorithms as used in previous related works (Weimer & Karatzoglou, 2007; Yu, Yu, Tresp, & Kriegel, 2006; Balakrishnan & Chopra, 2012). Similar to the experimental setup as followed in (Weimer & Karatzoglou, 2007; Yu et al., 2006; Balakrishnan & Chopra, 2012), datasets are created for three experiments. For this, firstly a fixed number of ratings per user, N, is sampled using randomization, and placed in the training set. The proposed models are experimented with N = 10, 20, and 50 training ratings per user. The remaining ratings by all

users are placed in the test set. Since NDCG@5 and NDCG@10 are evaluated, the users who have rated at least 20, 30, and 60 items respectively are filtered from the entire dataset. This results in a slight decrease in the number of users and items in the data sets.

To validate the performances, samples of filtered datasets are replicated 10 times, each with different random samples for the training and test set. The average values of performance metrics over these 10 replicates for each N are reported.

4.5.3 Impact of latent factors

Firstly, the impact of the number of latent factors (D) on the performance metrics for both datasets are investigated. The three variants of each dataset are tested by varying the latent factors (D) from D=10 to D=150 in multiples of 10 on performance metrics, viz., precision, F-measure, and NDCG.



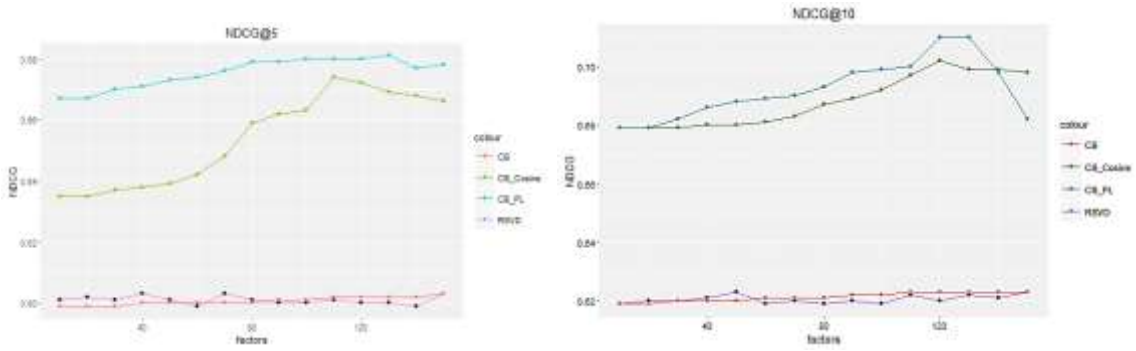


Figure 7: Impact of number of latent factors (D) on various performance metric for ml-100k data (N=10)

Figure 7 shows the impact of the variation of latent factors on performance metrics for the first variant on the ml-100k dataset. As the figure suggests, a variation in the number of latent factors (D) does not show any trend for classification metrics, viz., precision and F measure. The results on other variants of ml-100k datasets and on ml-1m datasets are shown in appendix B. Similar trends as reported in figure 7 are observed for the rest of the variants in both the datasets. All the three variants of the datasets also indicate that precision and F measure obtained by CB, CB_Cosine, and CB_PL outperform RSVD significantly. Moreover, CB_PL and CB_Cosine are better off than CB in all the three variants of the dataset.

If the ranking metrics, NDCG, are analysed on the three variants of ml-100k dataset, a trend with a variation of D for CB_Cosine and CB_PL models is observed. With an increase in D, NDCG values in all the three variants of data sets increase and then decrease for CB_Cosine and CB_PL. In the case of RSVD and CB there is no trend observed on the NDCG values obtained for either, respectively. Indeed, the proposed models CB_Cosine and CB_PL outperform RSVD and CB significantly on NDCG metrics.

4.5.4 Comparison

Secondly, to compare the proposed models on raking metrics, the NDCG values obtained on three variants of both the datasets have been reported. RSVD, CB, and List Rank MF-algorithms that optimize ranks, are compared with the proposed models, viz., CB_Cosine and CB_PL. The reported values of NDCG@5 and NDCG@10 for both the datasets and three variants are the best values obtained by varying the number of latent factors. The values of List rank MF are taken from the reported value in their previous work (Shi et al., 2010a).

Model	**N=10**		**N=20**		**N=50**	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RSVD	0.603	0.623	0.666	0.673	0.667	0.673
CB	0.603	0.623	0.667	0.672	0.667	0.675
List Rank	0.672	0.693	0.682	0.691	0.687	0.684
CB_Cosine	0.674	0.702	0.678	0.702	0.689	**0.716**
CB_PL	**0.681**	**0.710**	**0.682**	**0.709**	**0.678**	0.711

Table 5: Performance comparison in terms of NDCG between proposed model, baseline model (RSVD) and List rank on ml-100k dataset for different training dataset viz., N = 10, 20, and 50

The proposed models result in improvement from the baseline model (table 6) and List Rank by a fair margin in the ml-100k dataset. Since these reported values are cross-validated 10 folds it is highly unlikely that this is purely due to chance. Furthermore, it is also observed on the ml-1m dataset (table 7) that the proposed algorithm outperforms baseline and List Rank, which also indicates the importance of the appropriate loss functions.

Model	N=10		N=20		N=50	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RSVD	0.641	0.643	0.640	0.643	0.671	0.653
CB	0.645	0.649	0.659	0.668	0.703	0.683
List Rank	0.647	0.654	0.683	0.688	0.693	0.692
CB_Cosine	0.655	0.659	0.683	0.681	0.733	0.733
CB_PL	**0.660**	**0.668**	**0.723**	**0.731**	**0.763**	**0.763**

Table 6: Performance comparison in terms of NDCG between proposed model, baseline model (RSVD) and List rank on ml-1m dataset for different training dataset viz., N = 10, 20, and 50

4.6 Discussions and conclusions

This work proposes two algorithms for the ranking task in collaborative filtering using appropriate loss functions. List wise likelihood loss and cosine loss are aptly chosen to act as surrogate losses to learn the ranking task. This chapter has also used the cosine based latent factor model to learn latent features from the dataset which eventually augurs well with both the surrogate losses. The core idea of the chapter lies in arguing that an aptly designed loss function can improve the ranking prediction task which has to be in synchronization with the preference/score prediction.

Interestingly, the proposed algorithms also perform well on ranking tasks when measured on classification metrics. This implies that not only are the positions of the items adjusted in the recommended list more accurately, but better items also appear in the recommendation list as compared to RSVD and CB. Since e-commerce is now shifting to m-commerce, it is even

more necessary to use ranking methods in recommendation systems due to constrained recommendation lists that may appear on mobile platforms.

In the future, extension of the proposed model with other surrogate losses in ranking prediction can be investigated. Another interesting area could be to explore the relevance of ranking methods in solving the problem of novelty and diversity in the recommendation list.

5 The Long tail

5.1 Introduction

The Long Tail (TLT) is composed of a few popular items prevalent among many users, and the rest, which are not so popular among users, lie in the heavy tail . The heavy tail offers the possibility to explore and discover vast numbers of items using RS. Traditionally, in brick-and-mortar stores, there are items which are vastly popular and it is very difficult to search for items which are niche due to shelf space limitation. With the advent of e-commerce, the hits vs. niche concept is getting prominence due to the large enough availability of choices to satisfy even the most idiosyncratic user. The long tail phenomenon is consistent with two fundamentally different theories in principle. The first hypothesis, a more popular one, is that a majority of consumers follow the crowd and only a small proportion have any interest in niche items; the second hypothesis is that everyone is a bit peculiar, consuming both popular and niche items. Based on examining extensive data on user preferences for movies, music, Web search, and Web browsing, Goel et al. find better support for the latter theory. More specifically, their findings call into question the conventional wisdom that specialty products only appeal to a minority of consumers (2010).

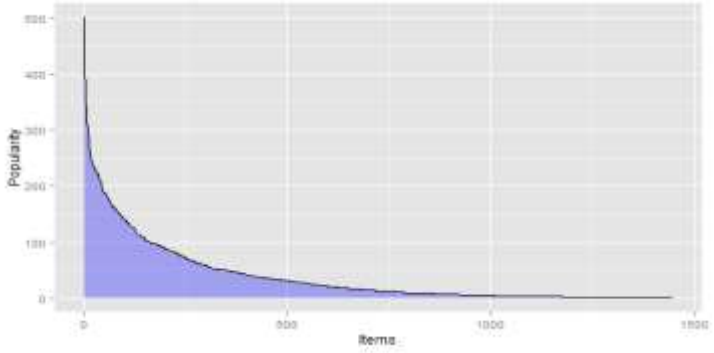


Figure 8: The long tail of items

The key promise of a RS is to help the consumer discover new and relevant items outside their sphere of interest (Hosanagar et al., 2014). It is in the interest of both e-commerce platforms and customers to deploy such a RS which increases long tail items in the recommendation list. Since popular items are anyways popular, recommending such popular and obvious recommendations do not add any significant value for the customer. Recommending not so obvious yet useful recommendations adds value for the customer as well impacts the return on investment (ROI) for e-commerce players (Goel et al., 2010). In order to keep up the promises, RS has developed from accuracy centric DSS to other quality seeking DSS such as novelty and diversity. The qualities such as novelty and diversity are related to the promotion of TLT items. Promoting TLT items have a positive effect on novelty and sales diversity (Vargas & Castells, 2014). There are several approaches that have been proposed in literature to cater to recommendations over such metrics, the majority of which apply re-sorting the top-K items generated by a baseline recommendation algorithm (Adomavicius & Kwon, 2012; Vargas & Castells, 2011).

However, the previous approaches in RS have not taken into account the effect of an individual customer's preferences for novel and diverse items. The previous approaches have assumed the response of every user to be uniform over novelty and diversity; however, it has been observed by empirical investigation that a user's personality influences the degree of novelty and diversity that can be enjoyed by a user in the recommendation list (Chen et al., 2013). In other words, different users have different degrees of liking/taste towards novel and diverse sets of items. Incorporating a user's liking/taste for TLT items in recommendation models is a challenge which has not been attempted till date, to the best of our knowledge.

Keeping in mind the above challenge of modelling a RS, this chapter proposes an innovative and robust model of matrix factorization, state-of-the-art algorithm in rating

prediction (Zhang, Séaghdha, Quercia, & Jambor, 2012), that engenders recommendations based on a user's tastes of TLT items. The matrix factorization technique maps the latent features of both items and corresponding features of users in the joint latent factor space of dimensionality such that the inner products of user-item interactions are modelled in the latent space (Koren & Bell, 2011). In an attempt to incorporate customer taste/liking for novel and diverse sets of items, this work introduces a parameter to capture the degree of liking of each user towards the novelty of the items. Novelty of an item acts as a surrogate measure for the long-tail. Further, as explained by Vargas and Castells, (2014), the promotion of long-tail items positively impacts the diversity of recommendation which has also been validated in this work. The matrix factorization approach has been suitably extended to capture the impact of the incorporated parameter to measure the user's taste of the TLT items. The optimization of the extended matrix factorization model ensures that the optimal liking of every user for long tail items is realized and unseen items are recommended to every user based on his/her interest of long-tail items.

If a firm chooses to aggressively promote niche items, so that diversification in sales help the firm to grow, a suitable strategy could be targeting only those customers who have previously shown an inkling towards such items. For targeting such customers, the parameters can be trained so that a user's disposition can be suitably improved towards long-tail items. Further, the model also focuses on promoting the number of long-tail items and the diversity of the recommendation list in proportion to the user's taste for long tail items. This is accomplished by training the user's taste in such a manner that every user would like long tail items more often. This would facilitate the RS to be flexible and recommend more optimal long tail items in certain situations.

5.2 Related work

"The Long Tail" notion was introduced by Chris Anderson in his book, *The Long Tail*, which described a couple of conditions to exploit the content available in niche segments (2006). These are: (i) make everything available, and (ii) help me find it. Due to e-commerce platforms, the former seems to be fulfilled as costs incurred by e-commerce for distribution and inventory are nearly negligible and therefore most of the items are available online. The focus, therefore, now is on the latter condition and the question to be asked is, do RS, in the current shape, help to find the niche products? There is an on-going debate whether RS generate niche items or popular items (Elberse & Oberholzer-Gee, 2006). It is found in one of the studies that with the use of RS, the more popular items are being recommended in video sales and the benefit as expected from the 'long tail' phenomenon has not been realised (Elberse & Oberholzer-Gee, 2006).

While the choice is available but there is the problem of channelizing niche items to such idiosyncratic users on internet platforms. Recommender systems that incorporate the quality of recommending Niche items to idiosyncratic users can add value to the ever growing e-commerce firms (Anderson, 2006). There have been few works to uncover the potential of recommender systems in this area. Since the field is still growing, there is neither a consensus on how to measure the quality of long tail recommendations nor a state- of- the-art algorithm that is widely acceptable. The literature related to the various works in this area and the corresponding metrics that measure the quality of long tail recommendations is reviewed.

Accuracy is a preferred measure for evaluating RS but now it has been argued that accuracy is not enough to measure the performance of RS (McNee, Riedl, & Konstan, 2006). One can understand this with the help of the following example: Suppose a user provides high rating to a Star Trek movie, inevitably he will receive recommendations of all other Star Trek movies. In this case the RS may be accurate but is it really useful? RS can be useful if it helps users in

discovering new items and therefore increase the sales diversity of e –commerce platforms. Based on literature, diversity can be classified into two ways: individual diversity and aggregate diversity. Individual diversity of recommendations for a user can be measured by calculating average dissimilarity between all pairs of items that have been recommended to a user. Aggregate diversity, on the other hand, is calculated across all users and is measured by the total number of distinct items among top-N items recommended across all users (Adomavicius & Kwon, 2011).

Along with diversity, novelty is another metric that was introduced in order to measure the impact of RS. Novel items are those which a user is unaware of (Konstan et al., 2006). Serendipity and coverage have also been introduced in literature apart from novelty and diversity in order to capture the performance of a RS (Ge, Delgado-Battenfeld, & Jannach, 2010; Herlocker et al., 2004; Shani & Gunawardana, 2011). Serendipity has been defined in the context of RS as the extent to which the items are attractive as well as surprising to a user (Herlocker et al., 2004). Coverage measures the percentage of items that can be predicted using the model of RS and therefore recommended to an active user (Herlocker et al., 2004; Ge et al., 2010).

There are a few research papers describing how diversity has been used in recommender systems. One approach of solving this problem was by the introduction of intra-list similarity metric and topic diversification for recommendation lists (McNee et al., 2006). Intuitively, diversity and accuracy are thought of as a trade-off between each other. This means diversity can be achieved at the expense of accuracy and vice-versa. In another method, the variance based approach was introduced in order to solve the accuracy–diversity tradeoff. Firstly, a k-nearest neighbor approach is adopted to predict the rating of an unseen item by an active user, followed by a ranking of items for a user based on variance of the k neighbor's rating for the

unseen item (Umyarov & Tuzhilin, 2009). Diversity metric in this chapter was taken as the total number of distinct items recommended across all users (Umyarov & Tuzhilin, 2009).

There is another view of the aggregate diversity metric or average diversity metric in the literature. It can be defined as the dissimilarity between recommended items based on the distance between feature vectors of user ratings. In order to solve the accuracy- diversity tradeoff in this case, a new strategy was introduced. It is a two-step process; in the first step recommendation is generated based on popular algorithm (like k-NN or SVD), and in the second step a quadratic optimization function is used in order to filter items having high diversity in the recommendation list (Zhang & Hurley, 2008). Similar to the above approach of optimization between accuracy and diversity, a Genetic Algorithm (GA) was proposed to maximize accuracy, novelty, and diversity simultaneously by hybridizing various popular models in RS (Ribeiro et al., 2012). The basic approach in the method is to search for pareto-optimal hybrids using the evolutionary search technique in GA (Ribeiro et al., 2012).

As a further step towards improving the aggregate diversity of a RS, the re-ranking approach has also been proposed in contrast to the optimization approach. The first step is to optimize ratings based on popular models and then adopting a re-ranking method to generate top-N recommendations in such a way that diversity and novelty are catered to along with accuracy. Several approaches of re-ranking the items have been discussed by Adomavicius and Kwon (2012). At first, a threshold value of the rating score is set after obtaining the rating score based on popular models. The rating scores greater than the threshold are re-ranked based on several approaches while the rating scores less than the threshold are ranked based on the decreasing value of the obtained scores. In order to re-rank the scores greater than a certain threshold, ‘item-popularity-based ranking approach’ is one such method used. Here, the less popular items are recommended to an active user first; popularity is measured as the number

of known ratings of the item being recommended. Another approach of re-ranking is 'reversing the predicted rating values', i.e., ranking the items from lowest to highest to a user, based on the obtained rating score in the first stage. Other approaches such as 'item average rating' re-rank the recommendation list according to an average of all known ratings; 'item absolute likeability' ranks items according to the number of likes (rating greater than some prefixed threshold) by the users in the database; 'item relative likeability' ranks items according to the percentage of likes; 'item rating variance' ranks items according to the rating variance of each item; and lastly, 'neighbors rating variance' ranks items according to the rating variance of neighbors of a particular user for a particular item. Further, aggregate diversity is measured according to three indexes, viz., entropy, Gini coefficient, and Herfindhal index. The proposed re-ranking approach shows the direction in solving the accuracy-diversity dilemma (Adomavicius & Kwon, 2012).

A taxonomy- based diversity measure was also introduced, which captures the distance between items based on the contents as features e.g., genre, language, actor, director, etc. (Ziegler, McNee, Konstan, & Lausen, 2005). The authors also propose a topic diversification algorithm in order to generate a recommendation list that is diverse and more useful for a user. A different perspective on enhancing diversity and diversity measures has been proposed by Zhang, (2009). The author borrows the concept of the concentration index from Economics and applies it to RS. The concentration index is calculated by first plotting a curve of the cumulative proportion of items in a user profile against the cumulative proportion of successful recommendations achieved on these items, and then finding the difference between the area below the diagonal and the area below (Zhang, 2009; Zhang & Hurley, 2009b). In order to generate a diverse recommendation list, three strategies have been laid out by the authors. The first strategy is to optimize an objective function that consists of diversity and accuracy as the maximization function, which comes after a recommendation list based on accuracy is

generated. The second strategy is to cluster items and then match the items of various clusters with the user's profile based on the SUGGEST algorithm (Zhang & Hurley, 2009a; Zhang, 2009).The third strategy is to apply singular value decomposition over item profiles before clustering the items. The next step follows the same as the second strategy (Zhang, 2009). A few other algorithms have also tried to address the diversity-accuracy dilemma. The chapter describes a hybrid method of generating accurate and diverse recommendations, made effective by one of the algorithms (Zhou et al., 2010). The authors introduced a heat-spreading (HeatS) algorithm and combined it with probabilistic spreading (ProbS) using a parameter 'λ' to generate diverse and accurate recommendations.

A metric has been proposed by Ge et al. to capture serendipity which measures the utility of unexpected recommendation. An observation regarding the tradeoff between accuracy, serendipity, and coverage has also been reported in the work (2010). Temporal diversity has also been introduced in literature which posits that the rating pattern of users varies with time, and a time factor needs to be introduced in order to capture the diversity of RS (Lathia, Hailes, Capra, & Amatriain, 2010).

A conference was entirely dedicated to novelty and diversity in Recommender systems (DiveRS) and was held in conjunction with ACM in the year 2011. In the first chapter presented in the conference, Adomavicius & Kwon (2011) proposed a graph-theoretic approach for maximizing aggregate diversity. In contrast to the re-ranking approach which balances diversity and accuracy in the second stage after generating the recommendation list based only on accuracy in first stage, Adomavicius and Kwon (2011) adopted a single stage graph-based approach that can obtain maximum possible diversity. In another chapter presented at the same conference, the focus was on how to improve user satisfaction by generating unexpected recommendations based on the utility theory of Economics. A new concept of unexpectedness

has also been proposed which is different from novelty, diversity, and serendipity (Adamopoulos & Tuzhilin, 2015). Expectedness is defined as the set of items that the user is thinking of as serving his current needs or fulfilling his intentions indicated by visiting the RS (Adamopoulos & Tuzhilin, 2011). The unexpectedness metric has been derived from the definition of expectedness by inculcating some unimodal functions of the distance between items. Authors have argued that unexpected items without their utility do not add value to the RS, so a metric which combines both unexpectedness and utility has been proposed. A fusion-based RS for improving serendipity has also been proposed at the same conference (Oku & Hattori, 2011). Fusion based RS generates a recommendation list of serendipitous items based on mixed features of two user-input items. The fusion of two items is brought about by first representing features of the items and then mixing features of the items by system defined criteria. Representation of the features of an item can be done either by bit representation or set representation. Pre-scoring the items in a recommendation list for a user is carried out in case the fusion-based RS generated many novel items (Oku & Hattori, 2011).

Since it is still a growing field, no consensus has been drawn on the metric formulation of diversity and novelty (Vargas & Castells, 2011). In order to formulate a formal definition of diversity and novelty in the context of RS Vargas and Castells proposed metrics based on different perspectives (2011). Novelty and diversity were defined by taking two notions as a base for building the metric. The two notions are the popularity of items and the similarity of items. Further, the metrics can be modified based on the user-item relationship. Choice, discovery, and relevance are the three user-item relationships which can exist and based on these user-item relationships different novelty and diversity metrics can be formulated (Vargas & Castells, 2011).

5.3 Model development

RS collects information on the preferences of its users for a set of items (e.g., movies, songs, books, travel destination, etc.). The information can be collected explicitly (typically, by extracting users' preferences in the form of star ratings) and/or implicitly (typically, by monitoring purchase history, browsing history, or even mouse clicks) (Koren, 2008). RS may also use demographics of users such as age, location, and gender. There is a growing trend of utilizing social information like followers, followed, and tweets for a personalized recommendation (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013).

In a typical e-commerce setup, there are thousands of users and products. However, explicit or implicit feedback for all the user-item pairs is not available due to the inability of a user to browse all the products available in the e-commerce database. Therefore, the user–item matrix R will be sparse, meaning that most of its entries are missing. Also, there are few popular items among many users, and the rest are in the heavy tail which are not so popular among users. The heavy tail offers the possibility to explore and discover vast numbers of items by using the RS. Traditionally, in brick-and-mortar stores, there are items which are mostly popular and it is very difficult to search items which are niche due to shelf space limitation. With the advent of e-commerce, the Hits vs. Niche concept is getting prominence due to the large availability of choice to satisfy even the most idiosyncratic user.

The goal of a recommender system is to recommend an unseen product based on previous explicit or/and implicit feedback. A typical RS first predicts the ratings of an unseen product by a user based on previous ratings given to products by the user himself and other users in the database. Based on the predicted ratings a recommended list of top-n products whose predicted ratings are highest for the user is generated. The typical collaborative filtering algorithm aims to improve the accuracy of the RS; however, accuracy is only one aspect of RS as described in the above section. There are several improvements in RS which have attempted to focus on

other aspects like novelty and diversity that indirectly have bearings on the long-tail. But these improved algorithms have assumed the uniform response of every user towards the aspect of novelty and diversity.

The aim of the proposed model is to promote long tail items by keeping in mind the accuracy of the RS as well as the response of the user for long-tail products captured in their previous transactions. Therefore, the proposed model differs from both these notions and introduces the concept of optimal promotion of TLT products. For this purpose, the user-item rating matrix R and novelty matrix τ are used to first train the proposed model and later use the parameters of the trained model on unseen items to predict their ratings. The detailed model is described in the next section.

5.3.1 Proposed Models

Notations

With many users and a large number of products in the database it is convenient to represent such a database in the form of a matrix. Consider a user–item matrix $R \in \mathbb{N}_o^{n \times m}$, with n rows and m columns, whose n rows correspond to users and m columns correspond to available products. The cells of this matrix R are ratings provided by a user corresponding to the items. These ratings represent the preferences of a user for items rated by him. Generally, these ratings are in range of 1 to 5, where 1 is the rating provided on least preferred items and 5 is the rating provided for the most preferred item. In the matrix form, every user is identified by its row index $i \in \{1, \ldots, n\}$, every product by its column index $j \in \{1, \ldots, m\}$ and R_{ij} stands for the rating given by user i for item j. In order to capture the novelty of an item at the time of rating by a user, another matrix $\tau \in \mathbb{N}_o^{n \times m}$ is considered which is similar to matrix R, the only difference being the value of the cells in the matrix. The values of the cells in the matrix τ represent the novelty of the items at the time when these items are rated.

Overcoming the Long-tail

Item Novelty

Before developing a model that optimizes the taste of a user for long-tail items, it is necessary to develop a measure of the long-tail. As noted in Vargas & Castells (2014) and Castells et al. (2011), novelty and diversity are stimulated by the promotion of long-tail items, and it would be prudent to take item novelty as a surrogate to depict the long-tail measure of items. Also, item novelty measures have been developed in previous works which are applied on the proposed model.

In the context of information retrieval, the novelty of a retrieval set has been defined as the proportion of known and unknown relevant items in the recommended list with respect to the end-user (Baeza-Yates & Ribeiro-Neto, 1999). Similarly, in the case of an RS item, novelty has been defined as the log of inverse of popularity (Castells et al., 2011; Zhou et al., 2009).

$$\text{novelty(i)} = \log_2\left(\frac{1}{\text{p(i)}}\right) \tag{5.1}$$

Where, p(i) represents probability that item i is liked. Probability of item i to be liked (p(i)) can also be interpreted as the popularity of an item.

$$\text{p(i)} = \frac{\text{\# users prefering item i}}{\text{\#total users}} \tag{5.2}$$

Popularity is the proportion of users who have considered the item as relevant with respect to the total number of users in the database.

The concept of novelty can be implicit when it is being considered as both the number of users considering the item relevant and the total number of users changing with time. This means that with the passage of time the novelty of an item may change. An item which is moderately popular (moderately novel) at the time of its introduction in the e-commerce

database may be very popular (not so novel) after the passage of certain months. So, novelty is a function of time and therefore, to make the representations clearer a subscript $'t'$ with item i is used

$$\text{novelty}(i_t) = \log_2\left(\frac{1}{p(i_t)}\right) \tag{5.3}$$

The novelty values can be represented in a matrix $\tau \in \mathbb{N}_o^{n \times m}$ corresponding to the rating provided by a user for an item. The dimension of this matrix is the same as matrix R and the novelty values are also in the same cells as those of matrix R.

Feature scaling of variables ensures that in machine learning algorithms there is no domination of one variable over another. So, scaling the item's novelty values obtained from the above expression is done before integrating novelty into the proposed model. For scaling, the base of the logarithm has been changed to calculate novelty such that the resultant value lies between 0 and 1. The steps to be followed in feature scaling are:

i. Calculate $\left(\frac{1}{p(i_t)}\right)$ of all the items at each instance of time t.

ii. Obtain minimum of $\left(\frac{1}{p(i_t)}\right)$ calculated in step-i, which is denoted by min $\left(\left(\frac{1}{p(i_t)}\right)\right)$.

iii. Divide each $\left(\frac{1}{p(i_t)}\right)$ by min $\left(\left(\frac{1}{p(i_t)}\right)\right)$.

iv. The maximum value obtained in step-iii is taken as the base of the logarithm

The resultant of the above is a feature scaled value of item novelty ($\text{novelty}(i_t^N)$) for each item at every instant when the rating for an item is provided by a user. A feature scaled novelty matrix, $\tau_{\text{scaled}} \in \mathbb{N}_o^{n \times m}$ is obtained using the resultant feature scaled values (See appendix for a toy example).

Basics of matrix factorization (RSVD Model)

Matrix factorization (MF) is a state-of-the-art algorithm in RS which has been made popular by its successful implementation in the Netflix Prize. Matrix factorization is the technique of decomposing a matrix into several matrices depending upon the suitability of the context. In this section, the basics of regularized singular value decomposition (RSVD), a variant of MF, has been introduced. The beauty of RSVD over other collaborative filtering algorithms is its ability of handling sparser matrices. The basic idea incorporated in RSVD is that users and items may be described by their latent features. Every item can be associated with a feature vector (Q_{ik}) which describes the type of product e.g., convenience vs. specialty, durable vs. non-durable, etc. Similarly, every user is associated with a corresponding feature vector (P_{uk}). The inner product between user feature vector and item feature vector is approximated as the predicted rating given by a user u for an item i. Mathematically, it can be expressed as:

$$\hat{r}_{ui} \approx P_{uk} Q_{ik}^{T} \tag{5.4}$$

Along with latent features, the inherent characteristics of user and item also contribute to the rating of an item by a user. For example, a functional product may always be rated a higher than average rating while an innovative product may always be rated below average. Similarly, a demanding user tends to rate on the lower side while a docile user may rate on the higher side. These inconsistencies are called biases of user and item and have to be captured in a model. Therefore, user bias (b_u) and item bias (b_i) are added to the model, as given in equation (5.1). User bias (b_u) is the observed deviation of a user u from the average rating of all users. Item bias (b_i) is the observed deviation of item i from the average rating for all items. An overall mean of ratings represented by μ is also added to the model to capture the bias of the dataset. The resulting model is given by following equation.

$$\min_{P_*\, Q_*} \sum_{(u,i \in \kappa)} \left(r_{ui} - \mu - b_u - b_i - P_{uk} Q_{ik}^T\right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro} + \|b_u\|^2 + \|b_i\|^2) \quad (5.5)$$

Here, κ is the set of known ratings in matrix R and $\|\cdot\|_{Fro}$ denotes the Frobenius norm. The parameter $\lambda > 0$ is a regularization parameter and is used to avoid over fitting of the model. Over fitting is a common term in a machine learning algorithm which suggests that the model will perform very well in the training set but will perform very badly in a test set on which the model is not trained. Training of the model is done only on the ratings available in the user-item matrix while the missing values in the matrix are skipped during training.

Integrating novelty with matrix factorization (PM-1)

Before delving into formulations that integrate novelty with matrix factorization it is prudent to develop baseline estimates that capture the characteristics of the database. Baseline estimates help in countering the item and user effect (Koren, 2008). The baseline estimates are widely used in matrix factorization formulation. The baseline estimate b_{ui} is formulated using the following equation:

$$b_{ui} = \mu + b_u + b_i \quad (5.6)$$

Further, an optimization model that is formulated by considering the novelty of items with respect to every user is introduced. The model assumes that there is a relationship between the rating of an item rated by a user and the corresponding item novelty. Using this assumption, the various parameters of the model are gradually constructed through an on-going refinement in the formulations. The initial formulation is represented by the following equation, where f denotes a function:

$$R_{ui} = f(\text{novelty}(i_t^N)) \quad (5.7)$$

To formulate a definitive relationship the model assumes that there is a linear relationship between an item's novelty and the rating given by a user. The parameter β_u will be used to formulate a linear relationship which can be interpreted as the weight a user ascribes to items' novelty. Since the weight ascribed to items' novelty would be different for different users therefore, parameter β has a subscript u. In order to capture the influence of novelty and the biases attached with users and items, an additive scheme is formulated that integrates baseline estimates with the normalized item novelty.

The resultant additive prediction scheme for estimating the rating of an unseen item can be solved by estimating the parameters of the following model. The parameters can be estimated by solving the following optimization problem:

$$\underset{b_* \beta_*}{\text{Min}} \sum_{(u,I \in \kappa)} \left(r_{ui} - \mu - b_u - b_i - \beta_u(\text{novelty}(i_t^N)\right)^2 + \lambda_1(\|b_u\|^2 + \|b_i\|^2 + \|\beta_u\|^2) \quad (5.8)$$

The above optimization problem can be solved by using the stochastic gradient descent (SGD) algorithm. The item novelty is pre-computed for unseen items, and the parameters so obtained, using SGD, will predict the rating of an unseen item. This formulation is an innovative idea in RS to optimize the ratings given by a user for an item with the corresponding novelty of the item.

However, optimizing the model solely based on novelty may result in a diminished accuracy of RS. Therefore, integrating this model with matrix factorization can fetch a solution that would result in optimal novelty without compromising on accuracy. A simpler scheme to integrate novelty with MF is to generate an additive model.

This additive model is simpler yet more effective in the prediction of ratings for unseen items with optimal novelty without compromising on accuracy. For estimating the parameters of the above model, the following optimization problem is solved:

$$\min_{b_* \beta_* P_* Q_*} \sum_{(u,i \in \kappa)} \left(r_{ui} - \mu - b_u - b_i - P_{uk} Q_{ik}^T - \beta_u(\text{novelty}(i_t^N))\right)^2 + \lambda_2 (\|b_u\|^2 + \|b_i\|^2 + \|\beta_u\|^2_{Fro} + \|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro}) \quad (5.9)$$

The integrated additive model proposed above is a combination of three stages: In the first stage, $\mu + b_u + b_i$ refers to a general description about user and item without accounting for any other interaction. The second stage, $P_u Q_i^T$ describes the interaction between features of items and corresponding features of users. The final stage, $\beta_u(\text{novelty}(i_t^N))$ calls for item novelty in the model which has to be optimized with respect to the likeness of each user corresponding to the item.

Fattening the long tail (PM-2)

In the previous section, a model is proposed that tries recommending items to idiosyncratic users based on their choice of long tail items. However, it is often valuable to recommend more long-tail items for both users and sellers of an e-commerce platform (Steck et al., 2011). To make this feasible, a new model of RS is developed that endeavours recommending long-tail items to such users who have a positive inclination towards long-tail items. The basic idea of the proposed model is to train the parameters of the model on a given dataset such that the long-tail items are promoted to users having positive interest in them.

For training such a model, one can think of training it in a stratified manner. It means that the same model shall be adaptive to train in such a manner that novel and relevant items are more emphasized but other items are left to train with their implicit significance. This will ensure that the parameters of users are learnt in the scheme so that long-tail items are given

better rating that in turn may be used to rank the items for generating recommendations for each user. One such adaptive scheme can be formulated by extending the model described in the above section. Let us consider the following adaptive formulation:

$$\min_{b_*\,\beta_*\,P_*\,Q_*} \sum_{(u,i\,\in\kappa)} \left(r_{ui} * \alpha - \mu - b_u - b_i - P_u Q_i^T - \beta_u(\text{novelty}(i_t^N))\right)^2 + \lambda_3(\|b_u\|^2 + \|b_i\|^2 + \|\beta_u\|^2 + \|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro}) \quad (5.10)$$

Where, $\alpha = \alpha > 1$; $\forall\, \theta_N > \theta$ and

$\alpha = 1$; $\forall\, \theta_N \leq \theta$; θ_N is a prefixed value of the item novelty ($\text{novelty}(i_t^N)$) for relevant rating (r_{ui}) to be used in determining long-tail and relevant items.

Here α is taken as a discrete numeric constant and can be interpreted as a training factor associated with stratified long-tail and popular items. Stratified training is key to formulate the scheme so that relevant long tail items are boosted while the corresponding parameters of users are adjusted automatically in the learning process of the model. For a simpler scheme, α is chosen as a discrete value greater than 1 for items which are in the long tail and value equals 1 for items other than in the long tail. Long tail items can be defined by a prefixed item novelty threshold value (θ_N). For identified relevant novel items training weight (α) greater than 1 implies that the rating is boosted by α times the original rating. For example, if an item that is relevant and novel has an original rating of '4' may be boosted to '4.8' when the training weight is taken as '1.2', while if the training weight is taken less than '1' (say 0.8), then the novel and relevant items will be diminished to '3.2' from the original rating of '4'. Therefore, the value of training weight α has to be chosen in such a way that it tends to boost the rating rather than diminish it in order to fulfil the objective of promoting long tail items.

Algorithm*:* To generate a recommendation list for each user by promoting optimal long-tail items.

Input:

- R : A matrix of rating, dimension N x M (user item rating matrix)
- κ : Set of know ratings in matrix R
- P_{uk} : An initial vector of dimension N x l (User feature vector)
- Q_{ik} : An initial vector of dimension M x l (item feature vector)
- b_u : Bias of user u.
- b_i : Bias of item i.
- μ : Average rating of all users
- l : Number of latent features to be trained
- τ_{scaled} : A matrix of normalized item novelty values
- β_u : Novelty weight
- e_{ij} : error between predicted and actual rating
- α : Training factor

Parameters:

- η : learning rate
- λ_2 : over fitting regularization parameter

- **Steps :** Number of iterations

Output: A matrix factorization approach to generate a recommendation list promoting optimal long-tail items

Method:

1) Initialize random values to vectors P_{uk}, Q_{ik}, β_u, $b_u$and b_i

2) Fix values of η, α and λ_2.

3) continue till error converges [error(step-1) - error(step) < ε]

error (step)=$\left(r_{ui} * \alpha - \mu - b_u - b_i - P_{uk}Q_{ik}^T - \beta_u(\text{novelty}(i_t^N)\right)^2$

4) for each $R \in \kappa$

Compute $e_{ij} \stackrel{\text{def}}{=} r_{ui} - \mu - b_u - b_i - P_{uk}(\cdot)Q_{ik}^T - \beta_u(\text{novelty}(i_t^N))$

Update training parameters

$b_u \longleftarrow b_u + \eta(2e_{ij} - \lambda_2 b_u)$

$b_i \longleftarrow b_i + \eta(2e_{ij} - \lambda_2 b_i)$

$P_{uk} \longleftarrow P_{uk} + \eta(2e_{ij} Q_{ik}^T - \lambda_2 P_{uk})$

$Q_{ik} \longleftarrow Q_{ik} + \eta(2e_{ij} P_{uk} - \lambda_2 Q_{ik})$

$\beta_u \longleftarrow \beta_u + \eta(2e_{ij} \text{novelty}(i_t^N) - \lambda_2 \beta_u)$

5) endfor

6) endfor

7) return P_{uk}, Q_{ik}, β_u, $b_u$and b_i

8) for each $R \notin \kappa$ predict the ratings for user and item

$$\hat{r}_{ui} = \mu + b_u + b_i + P_{uk}(\cdot)Q_{ik}^{T} + \beta_u(\text{novelty}(i_t^N))$$

9) rank and return top-K predicted ratings for each user in a descending order

5.4 Experimentation and evaluation

Two different datasets are used for the experimental evaluations of the proposed method. The first one is a publicly available Movie Lens dataset (ml-100k). The dataset consists of ratings of movies provided by users with corresponding user and movie IDs. There are 943 users and 1682 movies with a total of 100,000 ratings in the dataset. Had every user rated every movie, the total ratings available should have been 1586126 (i.e. 943×1682); however, only 100,000 ratings are available which means that not every user has rated every movie and the dataset is very sparse (93.7%). This dataset resembles an actual scenario in e-commerce, where not every user explicitly or implicitly expresses preferences for every item. Also, corresponding to every rating by a user for an item, a timestamp is provided in the dataset. The timestamp is useful while computing the novelty of an item at that point of time when it will be used in the proposed models.

The second dataset consists of movie reviews from Amazon. The data spans a period of more than 10 years, including approximately 8 million reviews up to October, 2012. Reviews include product and user information, ratings, timestamp, and a plaintext review. The total number of users is 889,176 and the total number of products is 253,059. In order to use this dataset for experimentation purposes a random sample of the dataset is taken to include 6466 users and 25350 products with only users, items, ratings, and timestamp intact in the data. The

total number of ratings available in the sub- sampled dataset is 54996, which makes the data sparser than ml-100 k dataset (99.67% sparsity).

To distinguish relevant (positive) and non-relevant (negative) items, a rating of '5' is set for relevant items. The categorization of relevant and non-relevant items will be used in calculating accuracy measures like precision. This will also be helpful in calculating the number of relevant long-tail items recommended by a RS.

5.4.1 Evaluation measures

Accuracy measures

In order to evaluate accuracy, the RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) are popular metrics in Recommender Systems. Since RMSE gives more weightage to larger values of errors while MAE gives equal weightage to all values of errors, RMSE is preferred over MAE while evaluating the performance of RS. RMSE have been popular metrics in RS until very recently and many previous works have based their findings on these metrics; therefore, they have been used primarily to exhibit the performance of the proposed models and the RSVD model on various datasets. For a test user item matrix 'Γ' the predicted rating $\hat{r}_{ui}$ for user-item pairs (u, i) for which the true item rating r_{ui} are known, the RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|\Gamma|} \sum_{(u,i \in \Gamma)} (\hat{r}_{ui} - r_{ui})^2}$$

MAE on the other hand is given by

$$\text{MAE} = \frac{1}{|\Gamma|} \sum_{(u,i \in \Gamma)} |\hat{r}_{ui} - r_{ui}|$$

However, when the task is to find good items, the MAE and RMSE metrics might not be appropriate. Therefore, a different accuracy metric that considers the frequency with which a RS makes correct or incorrect decisions (classifications) about whether an item is good or not has been in practice. The top-N good items are presented to the user and accuracy is measured based on the acceptance or rejection of items by the user. Precision is a measure to evaluate the accuracy in such circumstances and is defined as the ratio of relevant items presented, N_{rs}, to the total number of items presented, N_s.

$$\text{Precision} = \frac{\text{Nrs}}{\text{Ns}}$$

Evaluation measures of long-tail

The main notion of this chapter revolves around the items in the long-tail. Since the purpose of the chapter is to optimize TLT items with respect to every user and to promote TLT items to idiosyncratic users the overall frequency and coverage is used as a measure of long-tail items presented to all users by a RS. A simple rule is followed for categorizing long-tail items and popular items in various datasets. Firstly, the popularity of all items in a dataset is calculated by finding the number of users who have rated the items as relevant. Then a cut-off value for the categorization of long-tail and popular items will be determined based on the top 10% of items arranged in decreasing order of popularity. It means that items in the dataset will be arranged in decreasing order of popularity and the cut-off value of popularity is the one which covers the top 10% of the items in popularity rank. Using this cutoff value, those long-tail items can be identified which will be used to evaluate the performance of the recommender system with respect to long-tail. The two measures to evaluate the performance of RS with respect to the long-tail problem are: (i) coverage of long-tail (C) and (ii) frequency of long-tail (ɳ).

Coverage of long–tail is defined as the ratio of number of distinct long-tail items in the recommended list of all users and the total number of long-tail items available in dataset.

$$C = \frac{\#\text{unique long} - \text{tail items in recommended list}}{\#\text{ total long} - \text{tail items in dataset}}$$

Frequency (η) of long-tail items is defined as the summation of the frequencies of all the long-tail items appearing in the recommended lists by the RS for all users.

$$\eta = \sum_{(u \in \Gamma)} \#\text{ of long} - \text{tail items in recommended list of } u^{\text{th}} \text{ user}$$

As noted in Vargas and Castells (2014) and Castells et al. (2011), diversity is also positively stimulated by the promotion of long-tail items, and the same has been used as a performance measure of RS in the present work. The measure of diversity as used in a recommender system has been derived from the measurement of species diversity in ecology. Shannon entropy has been used as a measure of species diversity in ecology (Pielou, 1966). Similarly, in the case of recommender systems, Shannon entropy is a measure of item coverage and diversity of a RS model. It measures what percentages of items are recommended and how evenly they are distributed in the recommendations lists of users (Izadi et al., 2014). The following formula will be used to calculate the Shannon entropy based diversity:

$$\text{Diversity} = -\sum_{i=1}^{n} l_i \log_2 l_i$$

Where l_i represents the percentage of the recommendation list that contains item i and n is the number of items in the recommended list.

Cross validation

Cross validation is a well-established technique in machine learning algorithms used for evaluation purposes. This technique ensures that the evaluation results are unbiased estimates and are not due to chance. The dataset is split into disjoint k-folds; (k-1) folds are used as the training set while the left- out set is used for testing. The procedure is repeated k times so that each time a unique test set can be used for performance evaluation. The measures such as RMSE, MAE, precision, long-tail, and diversity used for the evaluation of RS models will be calculated k times and then averaged to get the resultant unbiased estimate of the performance measures (Bellogin et al., 2011).

5.4.2 Results

This sub-section reveals the value of parameters used for each dataset in the experimental setup and reports the corresponding observations:

For cross validation, 5-fold cross validation is applied on both the datasets. The number of latent factors (*I*) used in RSVD, proposed model-1, and proposed model-2 are kept constant at the value for which RSVD performs best on the cross-validated RMSE accuracy measure for the respective datasets in order to compare with the proposed models on long-tail measures. Since normalized item novelty ($novelty(i_t^N)$) varies between '0' and '1', where close to '0' indicates that the items are very popular at that instance, and close to '1' indicates a novel item, the threshold value (θ_N) is varied '0.1' to '0.9' in steps of '0.1' in the proposed model-2 to capture the variation in number of long-tail items with the varying threshold values of normalized novelty scores. The training factor $'\alpha'$ is also varied so that it adjusts the training of latent factors to promote long tail items in the proposed model-2 between '1' and '2' in steps of '0.2'.

For evaluating precision, relevant (positive) items are those whose rating equals to '5'. In order to evaluate frequency (Ŋ) of long tail items present in the recommended list, categorize the long-tail items in each dataset according to the stated rule in the previous section. In the ml-100k dataset the value of the top 10% popular items are covered above the popularity value 36; so by the predefined 10% rule, 36 is the cut-off value for classifying tail (non-hits) and head (hits) items. Similarly, in the Amazon dataset the cut-off value is found to be 2. A recommendation list of top-k (top 5 and top 10) items has been generated for each user based on RSVD, PM-1 and PM-2 models. Precision, frequency (Ŋ) of long tail items in recommended lists, and diversity of recommended lists are then calculated with top-k recommendations using various models for both datasets.

A comparative summary of results is presented in tables 7 and 8. The comparison between Regularized SVD and PM-1 illustrates the relative values of precision, long-tail (Ŋ) and diversity for both the datasets. The number of latent factors (*I*) in case of RSVD and PM-1 for the ml-100k dataset is taken as 5 since cross validated RMSE calculated is least for RSVD at *I* =5. The cross validated RMSE calculated for Amazon dataset is least at *I* =20, therefore the number of latent factor (*I*) for PM-1 is also chosen as 20.

Model	MAE	RMSE	Precision for k=5	Precision for k=10	Ŋ for k=5	Ŋ for k=10	diversity for k=5	diversity for k=10
RSVD	0.733	0.936	0.391	0.343	983.00	2283.00	4481.323	6125.70
PM-1	0.735	0.938	0.384	0.335	1023.20	2318.80	4564.628	6238.079

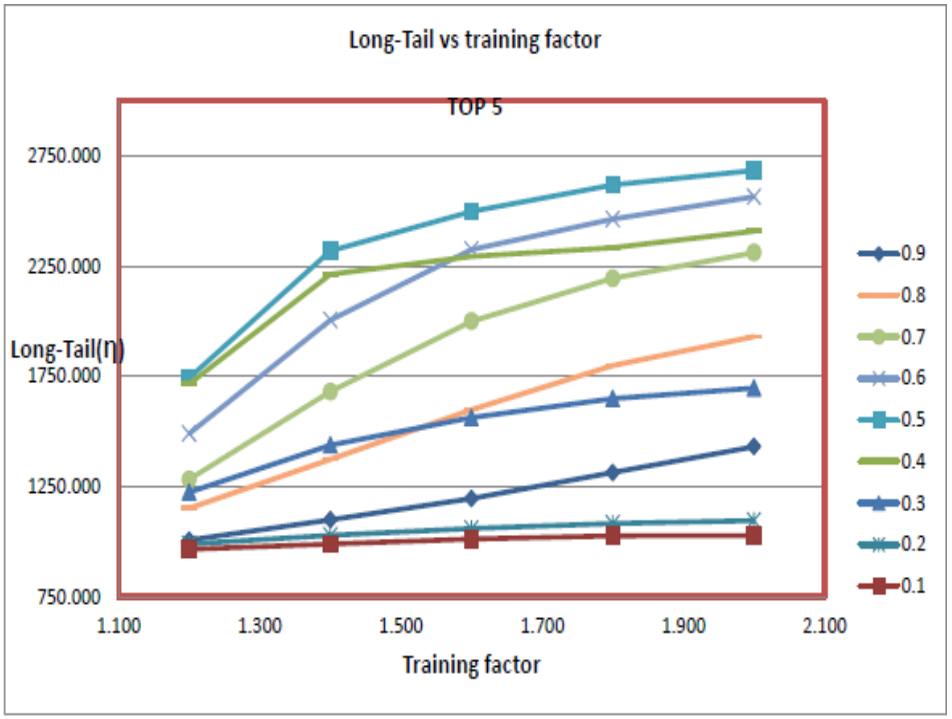
Table 7: Performance measures for ml 100k dataset

Model	MAE	RMSE	Precision for k=5	Precision for k=10	Ŋ for k=5	Ŋ for k=10	diversity for k=5	diversity for k=10
RSVD	0.686	1.057	0.590	0.578	3745.00	4579.60	66715.28	78369.29
PM-1	0.688	1.058	0.589	0.578	3947.00	4679.40	68768.09	78861.49

Table 8: Performance measures for Amazon dataset

Tables 7 and 8 depict performance metrics on the ml-100k dataset and the Amazon dataset, respectively. It suggests that the traditional RSVD generates less number of long-tail items in a recommendation list with slightly better precision value than PM-1. PM-1 demonstrates that with the proposed scheme an optimal number of long-tail items can be promoted to the idiosyncratic users. The proposed model PM-1 promotes long-tail items at the cost of marginally less precision which is in synchronous with previous literatures. As the focus of the proposed model is on both accuracy and optimal long-tail items, the precision of the model is nearly the same for both the datasets. Long-tail and diversity vary for both datasets along the expected lines.

The PM-2 scheme can be used to demonstrate the value of precision, frequency of long-tail (Ⴖ), and diversity by varying the parameters of the training factor (α) and item novelty threshold values (θ_N) for both datasets. The PM-2 scheme that promotes the long-tail items offers a trade-off with precision. There are two ways of promoting long-tail items: (i) by setting the various threshold item novelty values (θ_N) and (ii) by varying the training factor (α) during the training of PM-2 in the datasets. In both the datasets for a fixed θ_N, α will be varied and the observations are graphically presented.



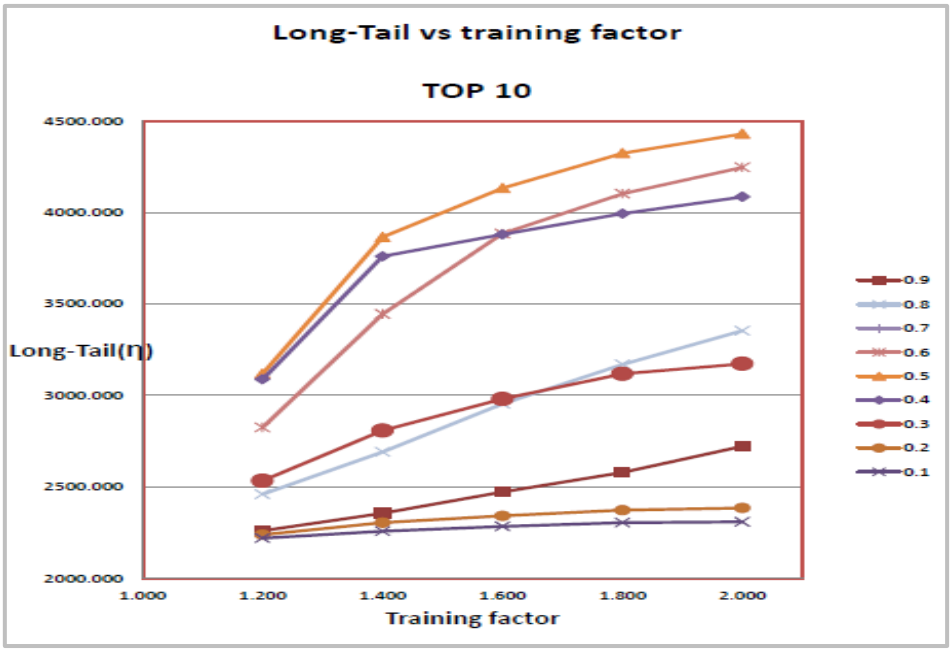


Figure 9: Relation between training factor (α) and long-tail (Ƞ) for ml-100k dataset in case of top-5 and top-10 recommendations

Long-tail (Ƞ) increases with increasing training factor (α)for ml-100k dataset (Figure 9), at various item novelty threshold values (θ_N), which vary from 0.1 to 0.9 for the ml-100k

dataset.. The graph is drawn for top-5 and top-10 recommendation list and it is clearly seen that more of TLT items are invariably recommended with increasing the training factor (α). However, increasing the threshold value (θ_N) does not necessarily result in more TLT items. Maximum number of TLT items are promoted at $\theta_N = 0.5$, while the least number of TLT items are promoted when $\theta_N = 0.1$ in both the recommendation lists. At $\theta_N = 0.1$ almost all items (hits and non-hits) in the dataset are equally boosted, while at $\theta_N = 0.5$ almost all non-hits items in the dataset are boosted, resulting in maximum Ƞ. As θ_N is increased from 0.5 to 0.9 the number of non-hits items in the dataset decreases, resulting in a proportional decline in Ƞ. So, based on the above result, one can say that $\theta_N = 0.5$ is the best threshold value for the partitioning of hits and non-hits items in the ml-100k dataset.

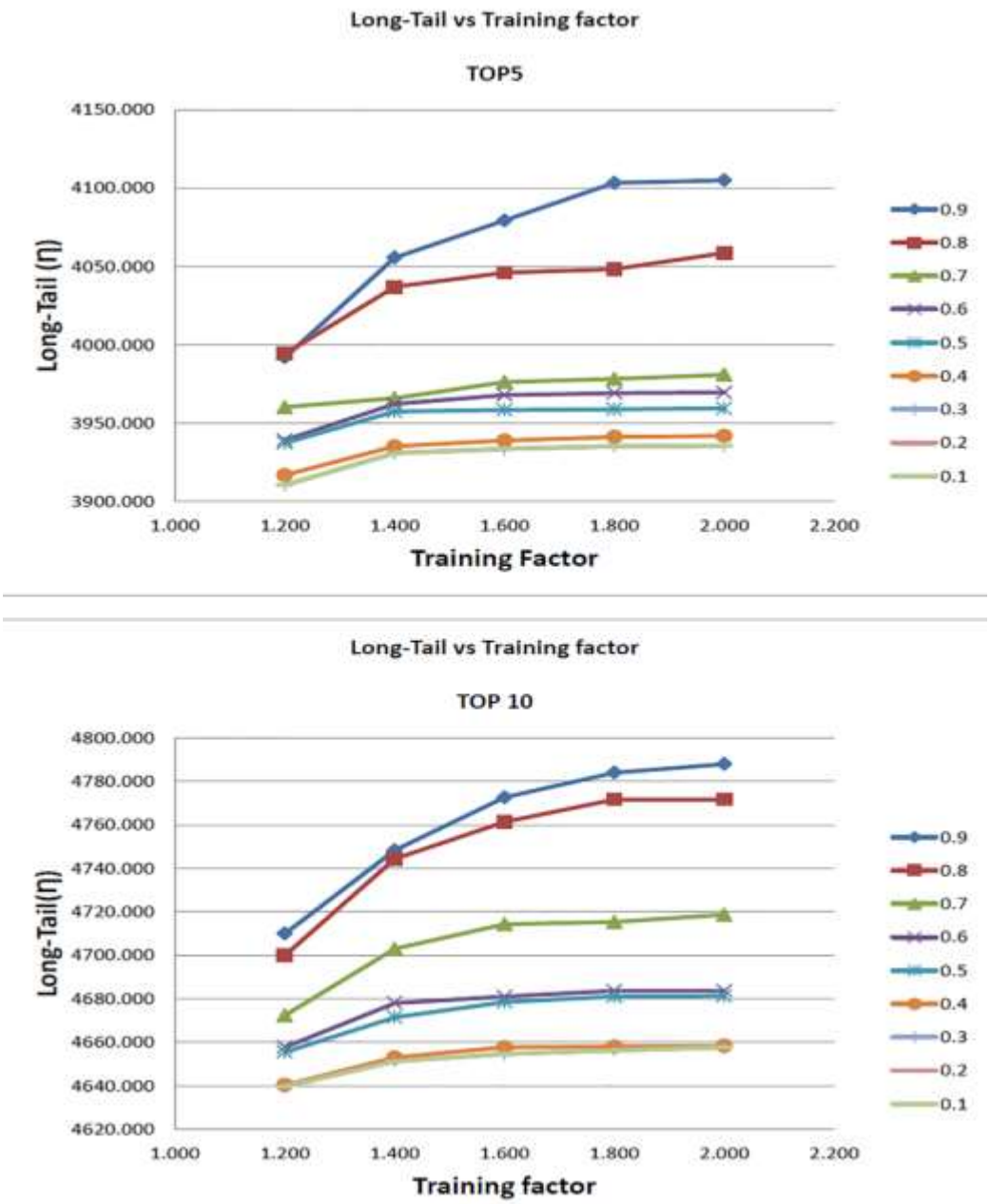


Figure 10: Relation between training factor (α) and long-tail (Ƞ) for Amazon dataset in case of top-5 and top-10 recommendations

In case of the Amazon dataset (Figure 10), the experimentation finds that with the rise of both training factor (α) and item novelty threshold values (θ_N), there is an increasing trend in Ƞ. The increasing trend of Ƞ with increasing α is model driven as is the case in previous datasets, however the increase in θ_N also invariably results in an increase in Ƞ in both the recommendation lists. The reason for this increase is attributed to the heavy presence of TLT items in the tail. At $\theta_N = 0.1$ almost all items (hits and non-hits) in the dataset are equally

boosted, while at $\theta_N = 0.5$ there are more hits items than non-hits items, and at $\theta_N = 0.9$ almost all non-hits items in the dataset are boosted resulting in maximum ⅁. If the strategy of an e-commerce firm is only to promote long-tail items to idiosyncratic users without other constraints, the above graphs alone can help in deciding the parameters of the model to implement in RS. However, solely promoting long-tail items may lead to a decline in precision which may undesirably affect the trust of a customer on RS and may result in trust deficit in the e-commerce firm. Hence, other measures like precision have to be looked upon before deciding the parameters of the model. Therefore, precision-long-tail relationship and long-tail-diversity relationship for both the datasets are also presented, which may give a better insight to e-commerce firms while deciding the parameters of the models based on the strategy of the firms.

There is a trade-off between precision and long-tail on the ml-100k dataset, as shown in Figure 11. It is observed from the figure that for the top-5 recommendation list the highest value of precision occurs at $\theta_N = 0.2, \alpha = 1.4;$ however, ⅁ is very low. Similarly, in the case of top-10 recommendation, the maximum value of precision is at $\theta_N = 0.2, \alpha = 1.6,$ but ⅁ is very low. In the case of both the top-5 and top-10 recommendations the highest value of ⅁ occurs at $\theta_N = 0.5$, $\alpha = 2$ with a low precision value which may be desirable for a firm which gives more weight to the promotion of long-tail rather than to precision. In the case of the top-5 recommendation list, if an e-commerce firm puts almost equal weightage on both long-tail and precision, then $\theta_N = 0.4, \alpha = 2$ could be a more sensible choice for model parameters. Similarly, for the top-10 recommendation list $\theta_N = 0.5, \alpha = 1.4$ could be the practical choice, considering the equal weightage provided to both precision and promotion of long-tail items.

Similar to the ml-100k dataset, the Amazon dataset also depicts an inverse relationship between precision and long-tail (Η͂) at fixed item novelty threshold values (θ_N). It is important to note from figures 11 and 12, that a lower level of precision does not automatically result in a greater number of long-tail items but it is also dependent on another parameter, θ_N. Therefore, it is worth considering all the parameters, viz., θ_N and α for choosing the right model for achieving a desired objective. The analysis done for the ml-100k dataset for choosing parameters of a model depending on varying objectives holds true in the case of the Amazon dataset as well. For the top-5 recommendation list the highest value of precision occurs at $\theta_N = 0.5, \alpha = 1.2;$ however, Η͂ is comparatively lower than $\theta_N = 0.5$. Similarly, in the case of the top-10 recommendation the maximum value of precision is at $\theta_N = 0.5$, but Η͂ is lower. In the case of both the top-5 recommendation and top-10 recommendation, the highest value of Η͂ occurs at $\theta_N = 0.9, \alpha = 2$, with lower precision value. However, the value of precision varies between 0.572 and 0.588 for the top-5 and between 0.569 and 0.581 for the top-10 recommendation lists. The range of variation is very low which suggests that precision is nearly constant for either of the parameters in the model, and therefore, one may need to only look at Η͂ while deciding the parameters of the model.

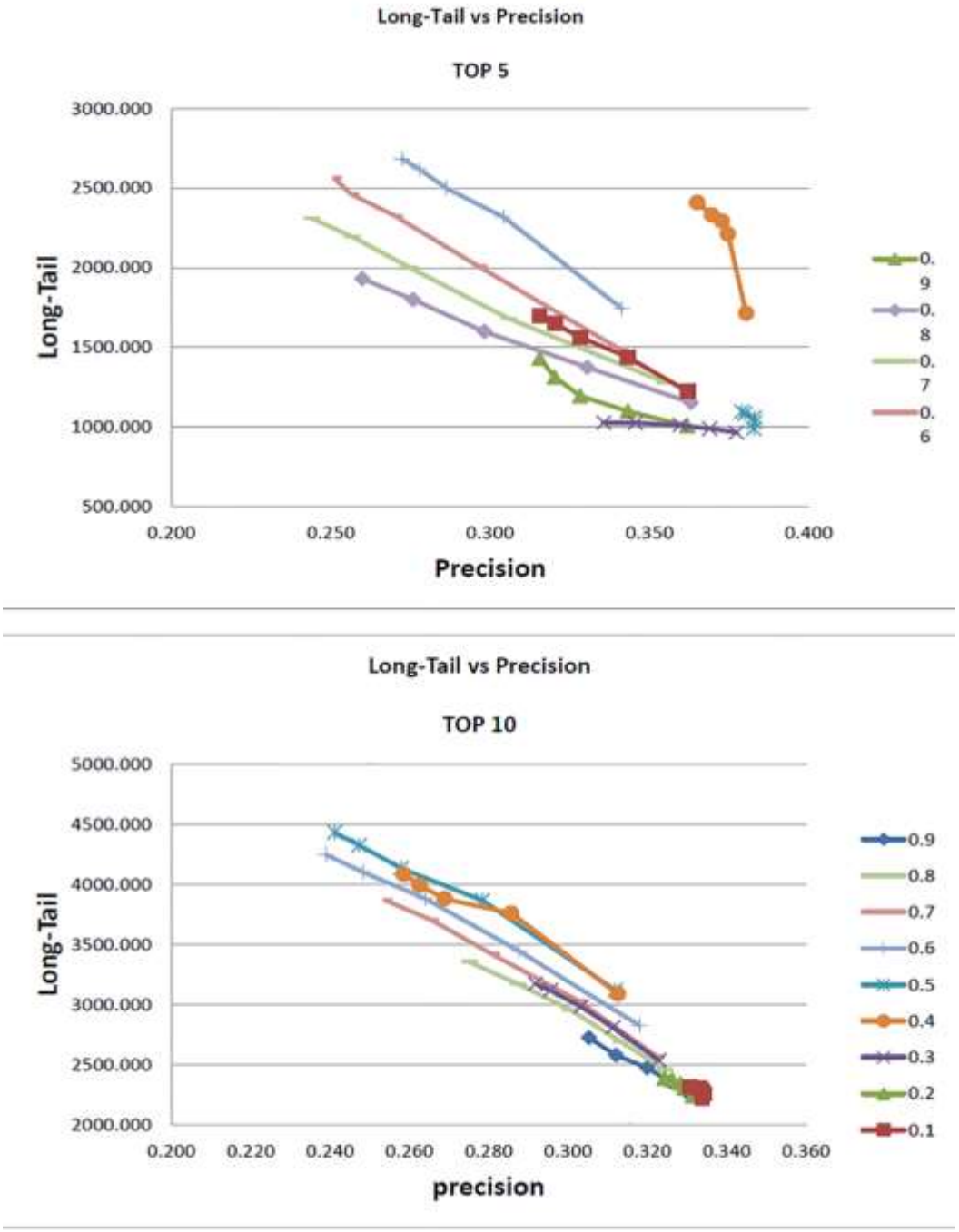


Figure 11: Relation between precision and long-tail (Ƞ) for ml-100k dataset in case of top-5 and top-10 recommendations for various training factor levels

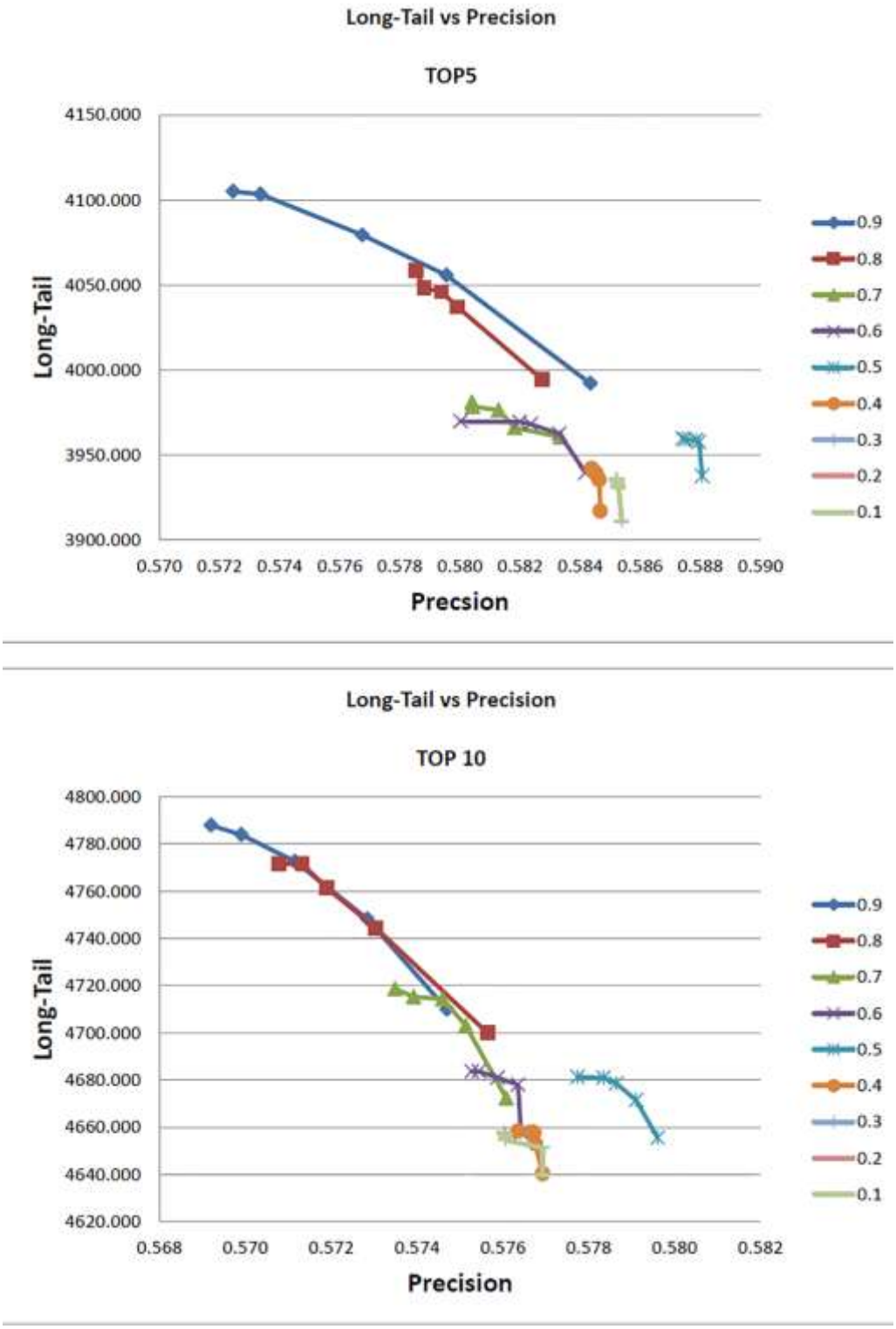


Figure 12: Relation between precision and long-tail (Ƞ) for ml-100k dataset in case of top-5 and top-10 recommendations for various training factor levels

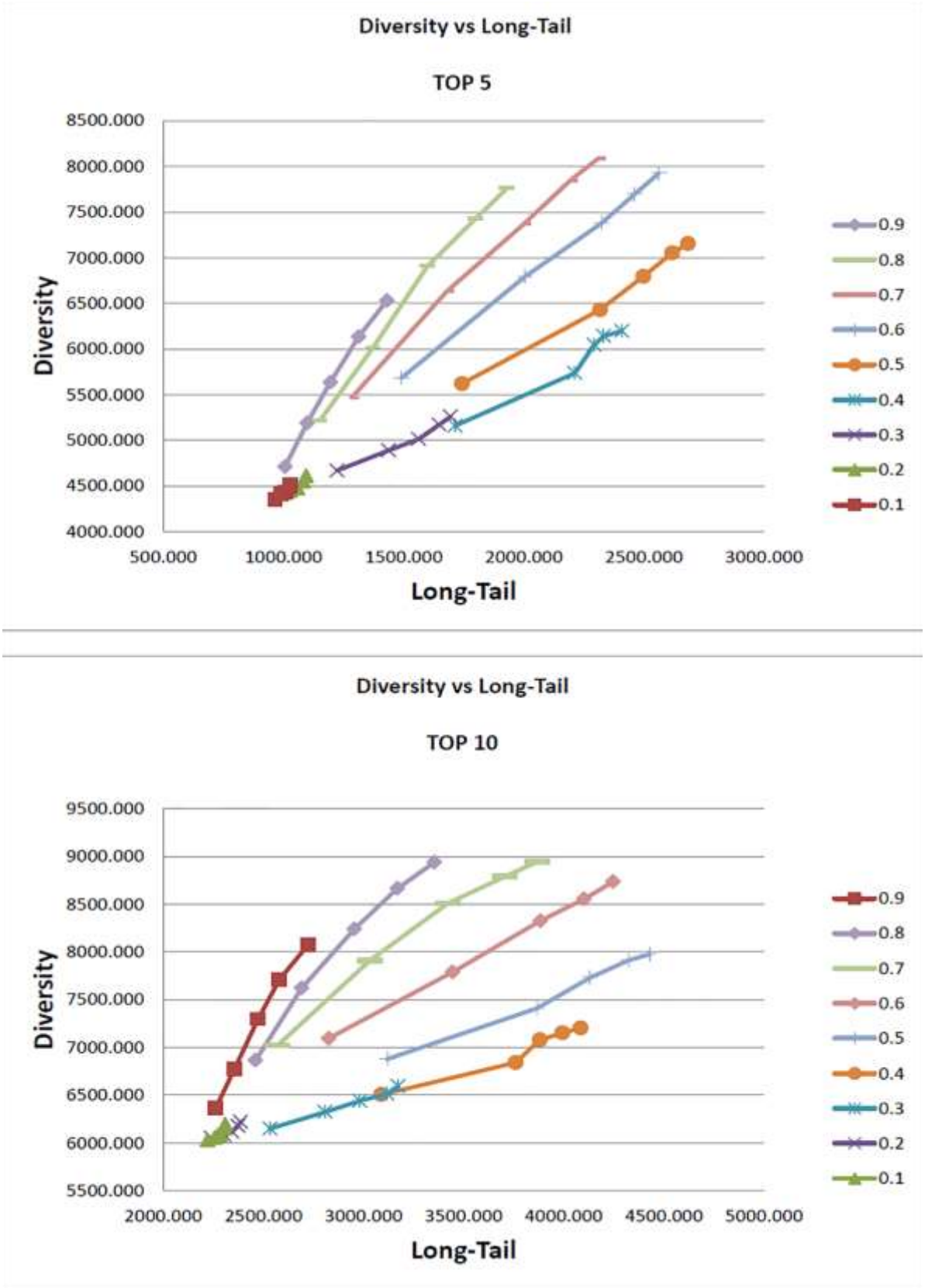


Figure 13: Relation between diversity and long-tail (Ƞ) for ml-100k dataset in case of top-5 and top-10 recommendations for various training factor levels

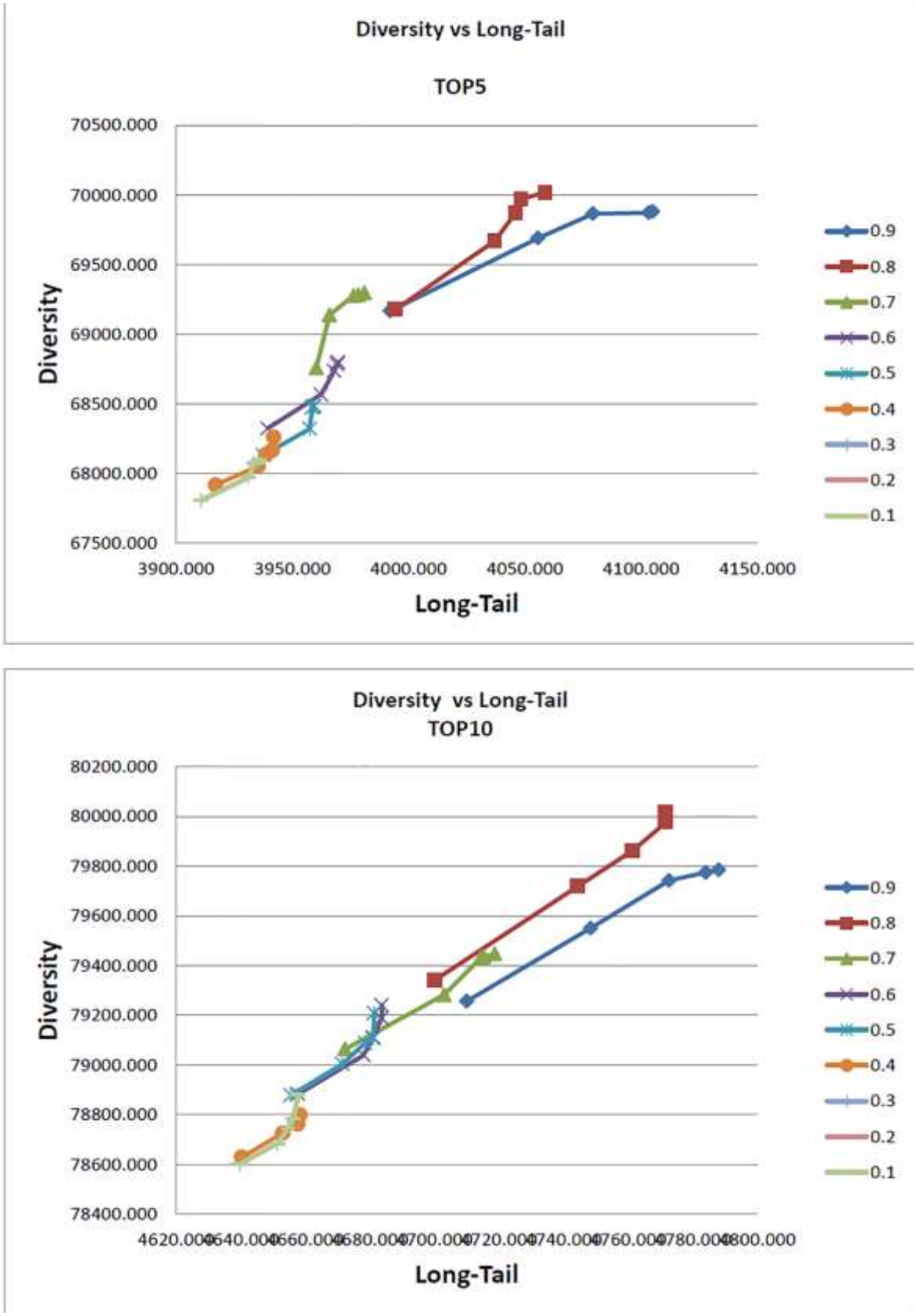


Figure 14: Relation between diversity and long-tail (Ī) for Amazon dataset in case of top-5 and top-10 recommendations for various training factor levels.

It is also worth observing the relationship between long-tail and diversity of the model for both datasets at various levels of θ_N and α. It is observed from figures 13 and 14 that the diversity of the recommendation list is positively stimulated by long-tail for a given θ_N. Diversity can also be another measure to consider before deciding the parameters of the proposed model. One can look at pairwise or overall trade-off between long-tail, diversity, and precision before deciding the parameters of the model, given the preference of the measures.

Depending on the objective and strategy of an e-commerce firm on precise, diverse, and long-tail items, the parameters of the proposed model can be suitably modified. In case of the ml-100k dataset, if the preference is for recommending more diverse items only, then considering the top-5 recommendation list $\theta_N = 0.7, \alpha = 2$ will be chosen, and considering the top-10 recommendation list $\theta_N = 0.8, \alpha = 2$ will be chosen as parameters. Similarly, in the case of the Amazon dataset the suitable parameters for recommending only diverse items considering top-5 and top-10 recommendation lists can be $\theta_N = 0.8, \alpha = 2$ for both the scenarios.

If the objective is to promote diverse as well as long-tail items, giving equal preference to both the performance measures $\theta_N = 0.6, \alpha = 2$ could be a suitable choice inferred from figure 18 in case of the top-5 recommendation list and $\theta_N = 0.7, \alpha = 2$ in case of the top-10 recommendation list for the ml-100 dataset. Likewise, in the case of the Amazon dataset (figure 14) with the above objective, $\theta_N = 0.9, \alpha = 2$ and $\theta_N = 0.9, \alpha = 2$ can be a parameter chosen for top-5 and top-10 recommendation lists.

5.5 Discussions and conclusions

This chapter exhibited through experimentation of the proposed models (PM-1 and PM-2) that it offers a variety of strategic business advantages for online business platforms by

implementing them to fulfil the desired business objective. The proposed model PM-1 guides the users to choose from a recommendation list based on their implicit liking/disliking of long-tail items. Since this will help users to navigate to niche items of their choice, thereby reducing the search time for suitable items, it increases their trust towards such recommender systems. This may in turn generate a 'recommendation-buying' spiral cycle which means that users relying on the recommender system will tend to buy from a recommendation list that will effectively improve the PM-1 with more information available about the users' behaviors (Yoon et al., 2013). The proposed model PM-2, on the other hand, can prove to be an expedient tool to promote long-tail, diverse, and precise items with flexibility to those users who have a better chance to like such items based on their past purchase behavior. Unexpected yet useful recommendations will indeed help e-commerce sellers to utilize their unlimited shelf space which is a major advantage over brick and mortar shops. Additionally, the tail item recommendation in turn, may boost head sales by offering consumers the convenience of "one-stop shopping" for both the mainstream and niche items (Goel et al., 2010). Hence, the return-on investment (ROI) for recommending niche products goes beyond direct revenue, encompassing second-order gains associated with improved consumer satisfaction and repeat patronage. In this way, recommender systems can act as personalized agents to promote cross selling and upselling of products, which are traditional marketing practices in the brick and mortar setup. Thus, given the benefits of both the proposed models, they can be valuable additions to the existing industry practices and research literature.

Recommender systems cannot be utilized maximally unless they recommend unexpected yet useful recommendations. Since there is an increasing focus on utilizing the unlimited shelf-space of e-commerce to benefit even the most unique user, this chapter proposes two novel algorithms based on prior likeness of niche items to idiosyncratic users. The proposed algorithms use latent factor additive models for their implementation. Unlike the algorithms

that have implicit assumption of uniform response to long-tail items by every user, irrespective of his/her liking on the aspect of novelty and diversity, the proposed model develops an algorithm to capture the optimal liking of every user for long-tail items. This is certainly an improvement over previous approaches adopted to address the long-tail problem. In addition, the proposed algorithms are flexible and engender the possibility of modification according to the desired objectives of precision, long-tail, and diversity. Firstly, the proposed model attempts to train the latent parameters of users based on their liking of novel items. The parameters so trained can be interpreted as the users' perception of novel items. The perception of users has been quantified in terms of a numeric value for each user which further determines his disposition towards an unseen item. Secondly, if a firm chooses to aggressively promote niche items so that diversification in sales helps it to grow, a suitable strategy could be to target only those customers who have previously shown an inkling towards such items. For targeting such customers, the parameters can be trained so that a user's disposition can be suitably improved towards long-tail items. The proposed models are a step forward in achieving the goal of unexpected yet useful recommendation.

Apart from trust building and sales diversity there is another view of the utility of the proposed optimal promotion of the long-tail model from a firm's perspective. The margin of popular products is often very less due to competition on both offline and online platforms but the margins of niche products are often on the higher side due to limited competition; therefore, it is logical to optimally recommend long-tail items to targeted customers. This will fatten the long-tail and ensure an increase in the profitability of a firm.

The model can be upgraded in future works so that the discrete novelty levels that have been used to boost the recommendation of long-tail items can be represented in an algebraic function which will make the model more robust. The multiplicative factor can also be

represented in a functional form of novelty and rating such that those parameters of users can be learned that help boosting of accurate and novel items.

6 Context weighted latent factor model

6.1 Introduction

In the era of digital ecosystems, it is worth inquiring if a user is happy or sad, if he is alone or with a companion, etc. This inquest to learn about a user's state can make the digital ecosystem more personalized and trustworthy. Emphasis on personalization in customer relationship management (CRM) is one of the most applied concepts in marketing (Chen & Popovich, 2003; Eirinaki & Vazirgiannis, 2003; Ranjan & Bhatnagar, 2009). One of the personalization systems in the digital arena that sought to suggest a product based on the prior purchase behavior of a user is known as a recommender system (RS). With varying contexts such as time, place, weather, and mood, an individual's preference changes and therefore, providing personalization services with respect to context is indispensable but also challenging in digital world.

RSs are decision support systems that filter the information overload according to a user's needs (Gao et al., 2010). RSs collect data about users' prior preferences and build models to suggest interesting products according to their taste. The preferences are expressed either explicitly by extracting users' preferences in the form of star rating and/or implicitly by monitoring purchase history, browsing history, or even mouse clicks (Koren, 2008). To make RSs adaptive to learn with varying contexts, context- aware recommender systems (CARSs) have been proposed in recent years.

Context in the purview of context- aware computing has been researched since the mid-1990s. Accordingly, context has been defined as any information that portrays the situation of an entity where the entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves

(Dey, 2001). Context awareness in the digital world has to be applied by bagging attributes such as the users' current positions, activities, and their surrounding environments (Dix, Finlay, Abowd, & Beale, 2004). This is essential to better appreciate the user's preferences in such contexts and realizing those preferences by offering personalized and context- aware services (Jung, 2012). The context- aware models can learn the impacts of multidimensional contexts on user-item interactions. Multidimensional contexts are situations where several contexts can play a role in determining a user's choice for an item, e.g., a user may like to watch *action* movies during 'weekdays' and 'when he is alone' but the user may like to watch a *romantic comedy* during the 'weekend' and 'with a companion'. Here, multidimensional contexts such as 'weekday' and 'when he is alone' are simultaneously impacting the choice of a user. One-dimensional contexts are situations where only one context appears to be impacting the user's choice for an item, e.g., a user, irrespective of other contexts, may always like to have *a coffee* 'after dinner'.

In the case of RSs, context has been incorporated in modelling and predicting interesting items to users based on their prior interaction with items and context. It is important to note that the inclusion of context in the modelling of RSs has helped more personalization in previous works relating to RS. Tensor factorization (TF) is considered to be the most accurate model in context- aware recommender systems (CARSs) (Karatzoglou, Amatriain, Baltrunas, & Oliver, 2010). TF extends the two-dimensional matrix factorization problem into an n-dimensional version for incorporating context. However, it also introduces a large number of parameters and these parameters grow exponentially with an increase in the database size (Karatzoglou et al., 2010), making the model very complex. In order to approach the problem in a simple yet effective manner, some of the earlier works have applied two-dimensional matrix factorization formulations to capture the context as well as user-item interaction (Baltrunas, Ludwig, & Ricci, 2011). They integrated the context by introducing a contextual

deviation term on items or item genres. The proposed model extends this concept to capture the impact of the context on both the user and item simultaneously.

When the one-dimensional context is available, the proposed model captures the user-item interaction by optimizing the latent parameters of users and items and optimizes the weights of contexts for capturing 'user-item-context' interaction. This model is termed as the context weighted matrix factorization (CWMF). Since the impact of context on both user and item are captured simultaneously in the model, it is apparent that it would give a better accuracy than the previous models.

For utilizing the proposed model in multidimensional contexts, this work has first decomposed the multidimensional contexts into sets of one-dimensional contexts. Then, recommendation can be generated for interaction among multidimensional contexts and user-item by combining the models learnt on one-dimensional contexts. For combining the set of models learnt in one- dimensional contexts two approaches have been applied, one by computation of mean and another by linear ridge-regression for blending the various models, which are standard practices adopted in RSs (Ekstrand, 2010; Jahrer, Töscher, & Legenstein, 2010; Koren, 2008; 2009; Takács et al., 2009; Vozalis & Margaritis, 2007). This work has examined the accuracy of the proposed context weighted matrix factorization model on the 'InCarMusic' and 'LDOS – CoMoDa' (Movie) datasets that is consistent with previous works (Baltrunas, Ludwig, et al., 2011; Baltrunas, Kaminskas, et al., 2011).

The main contribution of the proposed model is its ability to capture 'user-item-context' interaction with a lesser number of parameters and thereby, reduce its complexity. The proposed model also introduces the concept of a combination of models to generate recommendation in multi-dimensional contexts.

In the remainder of this chapter, an overview of the related work to CARSs is delivered. In the next section, the methodology of the proposed work is illustrated, and subsequently a new algorithm is proposed. Thereafter, the details of experimentation using InCarMusic and LDOS - CoMoDa (Movie) datasets have been presented. The experimentation is followed by results and discussions. Finally, the conclusion sums up the key aspects of the proposed model and guides towards the future direction of research in CARSs.

6.2 Related work

In contrast to the large literature on standard recommender systems, there are only few researches on the context-aware recommender systems. The basic approaches that are followed in CARSs can be divided mainly in three: 1) contextual pre-filtering, 2) contextual post-filtering, and 3) simultaneous modelling of context and user-item interaction (Champiri et al., 2015). As the names suggest, in contextual pre-filtering, the data is pre-processed based on the context of interest before applying the recommender. In contextual post- filtering, the recommender system is applied irrespective of context and then the contexts are post-processed. In the third approach the interactions between user, item, and contexts are simultaneously modelled to process the recommendation system.

One of the earlier works on conceptualization of CARSs dates back to early 2000. Since then there has been substantial progress in developing novel models to handle contexts (Kwon, 2003). The context in the digital world is determined by a user's click behaviors, browsing, and time spent on images during browsing. Adomavicius et al. (2005) defined the problem of CARSs in a formal manner and discussed the solution by taking context as one more dimension in addition to users and items. One of the approaches suggested is a two-step process, wherein the first step, partition of the database is undertaken pertaining to various contexts, which means that similar contexts are grouped together, and then in the

second step, rating prediction is done by a collaborative filtering algorithm for each context (Adomavicius et al., 2005). In the same chapter, the authors also proposed aggregating the hierarchies of context so that on a higher level some contexts can be combined and predicting the user's interest for an item can be done on the higher-level context. Another work in the context of CARSs was done for music recommendations. The authors (Lee & Lee, 2007) adopted a case based reasoning algorithm to provide top 15 recommendations for a user in a particular context. Case based reasoning tries to find top-k similar (method similar to k-NN) data points as the queried data points (Lee & Lee, 2007). Another work that developed an algorithm for context based movie recommendation used the Bayesian network for recommendation and promotion (Ono, Kurokawa, Motomura, & Asoh, 2007). Bayesian networks are flexible methods for modelling complex joint probability distribution of multiple random variables, but that being said, model construction is a challenging task. On the basis of user attributes, movie attributes, and context attributes, a hidden variable called the impression variable is conditioned using the Bayesian network for training the recommendation system(Ono et al., 2007). Another way of using Bayesian is the Bayesian classifier which has also been applied in context based RS (De Pessemier, Deryckere, & Martens, 2010). Generally, the algorithms in context- aware systems have considered high level contexts like location, time, or social aspects but the dynamic user's content evolution is difficult to model. In response to this, an algorithm called the mobile bandit algorithm is proposed to tackle dynamically changing user's content (Eroglu, Toprak, Urgan, Ozge, Denizbasi, Akoglu, Ozpolat, & Akoglu, 2012). Another dynamically changing context for a user is modelled in music recommendation. Based on the sequence of music, a hidden factor of context is determined using sequential pattern mining, and then using a contextual score, music recommendation is provided to a user (Hariri, Mobasher, & Burke, 2012).

In order to incorporate context during consumption of a product in the digital world a tag of cloud approach was proposed by De Pessemier, Deryckere, & Martens (2009). The concept of this chapter lies in a content based filtering approach in information retrieval a user; where a user's profile is generated based on ratings and tags given by him, and subsequently it can be used to predict ratings for unseen items (De Pessemier et al., 2009). Inferring a context by itself is challenging, if not stated explicitly by a user. One of the chapters used decision trees for inferring higher level context from the browsing history, location, and time (Hong, Suh, Kim, & Kim, 2009). By inferring a high level context, an association rule is applied to generate the recommendation list (Hong, Suh, & Kim, 2009). A similar attempt to build a profile lattice to capture context was made by Kwon and Kim in 2009. A mobile platform based recommender system using a social network that utilizes temporal and spatial context was proposed by Ramaswamy et al. They use a temporal and spatial function in order to derive a utility function which is dependent on time and location (2009). A novel method of automatically inferring context or activities such as running, walking, sleeping, studying, and shopping through low level sensor data from mobile phones using the Bayesian probabilistic model has been illustrated by Wang et al. The authors also train a statistical model of music content corresponding to each activity and then integrate context and music content to generate music recommendation (2012).

The matrix factorization approach has also been employed in contextual movie recommendation (Shi, Larson, & Hanjalic, 2010b). A joint matrix factorization was developed by the authors where mood specific similarity is calculated first and then the joint matrix factorization technique is implemented (Shi et al., 2010b). An N-dimensional matrix factorization, also known as tensor factorization can be used in the decomposition which takes user, item, and context as the 3dimensions for the decomposition of the rating matrix. An additive model that integrates context with matrix factorization has been proposed which

shows that it is as effective as tensor factorization but is less complex (Baltrunas, Ludwig et al., 2011). An interesting work which emphasises the concept of ubiquitous computing also uses matrix factorization in order to recommend music depending upon the context while travelling in a car (Baltrunas, Kaminskas et al., 2011). A fast factorization machine has also been proposed in the literature which claims to be less complex than a multiverse recommendation system due to its learning in linear time (Rendle, Gantner, Freudenthaler, & Schmidt-Thieme, 2011). It models the interaction between contexts, items, and users in a linear form and then tries to learn the parameters which are used in the rating prediction with contexts (Rendle et al., 2011). Another approach in recommender systems is the learning to rank approach which tries to optimize the rank of items in the recommendation list rather than the rating of items provided by a user. In CARSs, one such algorithm is TFMAP which uses mean average precision (MAP) in proxy to rank the recommended list and then employs tensor factorization (Shi et al., 2012). Yet another research work using matrix factorization in CARSs is of Odić, Tkalčič, Tasič, & Košir, which proposed three types of schemes for integrating context (2013). The first model utilizes only the users' bias varying with context, the second utilizes only items' bias varying with context, and finally users' features only varying with context. The authors also stress on finding the important contexts statistically relevant to the dataset, so that irrelevant contexts can be ignored while training and further assist in providing recommendation (Odić et al., 2013). In the case of multi-dimensional context modelling, an algorithm becomes a challenging task. The approach suggested is to first decompose the multi-dimensional contexts to several one-dimensional contexts by averaging the preferences score by keeping other contexts fixed. Then, the algorithm predicts the preference score for an unseen item and unseen contexts by applying SVD on each decomposed data and finding the coefficient of contextual influence which is then linearly added (Wang, Meng, & Zhang, 2013).A non-linear Gaussian process factorization machine

which assumes the relation between user, item, and context as non-linear is also proposed and evaluated on 5 benchmark datasets (Nguyen et al., 2014).

This work differs from previous approaches in at least two ways. First, this work models context differently than other works. This work considers context as a component of weight assigned to each user-item latent interaction. This is a better approach as it reduces complexity when compared to the tensor factorization methods. Secondly, multi-dimensional context has been handled by combining the one-dimensional contexts. Combining one-dimensional contexts in the model in this way limits the variables to be learnt at a time.

6.3 Model development

6.3.1 Problem definition

Unlike the standard RS problem where only user ($u \in U$) and item ($i \in I$) are used for modelling RSs based on the interaction of users with items, the context-aware recommender system considers user ($u \in U$), item ($i \in I$), and contexts ($c \in C$) for modelling interaction among them for generating recommendations. One important point to note is that there may be simultaneous impacts of multiple contexts at their respective levels that may affect the choice of a consumer. It means that contexts C can be represented by a combination of $C_1 \times C_2 \times C_3 \times C_4 \ldots \ldots \times C_k$. Also, a context can have several levels, e.g., weather can be a context and it has “rainy”, “hot & dry”, “hot & humid”, “cold” etc. as levels of the context.

In a standard RS problem for rating prediction, the task is to learn a function $y : U \times I \rightarrow R$. Here, R is the set of ratings for a user-item combination and y is a function that maps users and items. Similarly, in CARSs, the task is to predict ratings by learning a function $z : U \times I \times C \rightarrow R_C$, where, R_C denotes the set of the ratings for a user-item-context combination; z maps users, items, and contexts.

6.3.2 Notations

With many users, large number of products, and many contexts, each context with multiple levels available in the database, it is convenient to represent such database in the form of a multi-dimensional tensor. Generally, these ratings are in the range of 1 to 5 integers, where 1 is the rating provided for least preferred items and 5 is the rating provided for the most preferred items. Every user is identified by an index, $a \in \{1, \dots, m\}$, every item by its index, $b \in \{1, \dots, n\}$ and every context by its index, $k \in \{1, \dots, p\}$. Within the context, there can be multiple levels; therefore, l represents the levels of the contexts being used. The following table summarizes the notations to be used in this chapter.

Notations	Meaning
$\mathbf{u_a}$	a^{th} User where, $a \in \{1, \dots, m\}$
$\mathbf{i_b}$	b^{th} Item where, $b \in \{1, \dots, n\}$
$\mathbf{C_{kl}}$	k^{th} Context with level '*l*' where, $k \in \{1, \dots, p\}$
$\mathbf{r_{u_a i_b C_{kl}}}$	Rating provided by user 'a' for item 'b' in context 'k' with level '*l*'

Table 9: Basic notations

6.3.3 Models

In this section, this chapter first introduces the concept of matrix factorization technique that has been applied to solve standard RS problems, and later uses this as a basis to develop a novel algorithm to solve the context- aware problem in RS.

Matrix factorization

The matrix factorization (MF) technique, where interaction of user features with item features is modelled to the ratings given by a user to an item, has been taken as the basis in the proposed model, viz., context weighted matrix factorization (CWMF). Matrix factorization trains the parameters of the model which are obtained by solving the following optimization problem (Koren, 2008):

$$\mathcal{L} = \min_{P_* \, Q_*} \sum_{(u,i \in \kappa)} \left(r_{ui} - \mu - b_u - b_i - P_{uk} Q_{ik}^T\right)^2 + \lambda(\|P_{uk}\|^2{}_{Fro} + \|Q_{ik}\|^2{}_{Fro} + \|b_u\|^2 + \|b_i\|^2) \qquad (6.1)$$

Here, κ is set of known ratings and $\|\cdot\|_{Fro}$ the Frobenius norm. The parameter $\lambda > 0$ is a regularization parameter and is used to avoid over-fitting of the model. Every item can be associated with a feature vector (Q_i) which describes the type of product, e.g., convenience vs. specialty, durable vs. non-durable, etc. Similarly, every user is associated with a corresponding feature vector (P_u). The inner product between user feature vector and item feature vector is approximated as the predicted rating given by a user u for an item i. User bias (b_u) is the observed deviation of a user u from the average rating of all users. Item bias (b_i) is the observed deviation of item i from the average rating for all items. An overall mean of ratings represented by μ is also added to the model to capture the bias of the dataset. The above optimization problem can be solved by alternatively performing a stochastic gradient descent of latent features of users and items (Koren, 2008; Larson, 2010).

Context weighted matrix factorization (CWMF)

This chapter will first develop a model using MF that is suitable for the one-dimensional context and later extend this to the multidimensional context. For employing MF to capture interaction of user-item simultaneously with a context, a parameter called context weight is introduced. The context weight measures the impact of a context corresponding to each user-item interaction. The parameter '$\beta_{C_{kl}}$' denotes the ascribed weight to context 'C_{kl}' in a user-item-context interaction. If a context 'C_{kl}' is not available for a user-item interaction it is treated as null or zero. So, the resultant matrix factorization scheme that takes care of user-item and context is given by the following loss function

$$\mathcal{L} = \min_{\beta_* P_* Q_*} \sum_{(u,i,c \in \kappa)} \left(r_{u_a i_b C_{kl}} - \hat{r}_{u_a i_b C_{kl}}\right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|Q_{ik}\|^2_{Fro} + \|b_u\|^2 + \|b_i\|^2 + \left\|\beta_{C_{kl}}\right\|^2) \quad (6.2)$$

Where, the predicted rating is given by $\hat{r}_{u_a i_b C_{kl}} = \mu + b_u + b_i + (P_{uk} Q_{ik}^T)\beta_{C_{kl}}$ and λ is the regularization parameter to avoid over-fitting. It is to be noted that $\lambda\left\|\beta_{C_{kl}}\right\|^2$ is added for regularization along with other parameters of MF as expressed in equation (6.1). The regularization parameter is obtained by cross-validation (Baltrunas, Ludwig, et al., 2011; Yu et al., 2009).

Given a set of training data κ , optimization to local minima can be achieved by alternatively performing gradient descent of latent features and weights of contexts in the above loss function, as mentioned in the previous sub-section.

Let us denote the difference between predicted and actual rating by error (e):

$$e = r_{u_a i_b C_{kl}} - \hat{r}_{u_a i_b C_{kl}} \quad (6.3)$$

The gradients of loss function with respect to P_u, Q_i and $\beta_{C_{kl}}$ after substitution of equation (6.3) are given below:

$$\frac{\partial \mathcal{L}}{\partial P_{uk}} = 2(eQ_{ik}^T \beta_{C_{kl}} + \lambda P_{uk}) \quad (6.4)$$

$$\frac{\partial \mathcal{L}}{\partial Q_{ik}} = 2(eP_{uk} \beta_{C_{kl}} + \lambda Q_{ik}) \quad (6.5)$$

$$\frac{\partial \mathcal{L}}{\partial \beta_{C_{kl}}} = 2(eP_{uk} Q_i^T + \lambda \beta_{C_{kl}}) \quad (6.6)$$

By randomly initializing the parameters of loss function as depicted in equation (6.2), the local minima of the function are reached by descending against the gradients with a learning

parameter (η). The learning of parameters is stopped as soon as the decrease in the sum of the square of the overall error in the set of training data κ dips below a prefixed, infinitely small, positive threshold value, **ε**. The following pseudo code illustrates the CWMF method in detail:

Algorithm:

Input:

- R : A matrix of rating, dimension M x N (user item rating matrix)
- κ : Set of known ratings in matrix R
- P_{uk} : An initial vector of dimension M x F (User feature vector)
- Q_{ik} : An initial vector of dimension N x F (item feature vector)
- b_u : Bias of user u.
- b_i : Bias of item i.
- μ : Average rating of all users
- F : Number of latent features to be trained
- e : error between predicted and actual rating
- $\beta_{C_{kl}}$: the parameter of context to be trained

Parameters:

- α_1 : learning rate
- λ : over fitting regularization parameter (obtained using cross-validation)
- **Steps:** Number of iterations

Output: A matrix factorization approach to generate recommendation list

Method:

5) Initialize random values to matrix Q_i, $P_u$and vectors, $\beta_{C_{kl}}$, $b_u$and b_i

6) Fix value of α_1 and λ.

7) do till error converges [error(step-1) - error(step) < ε]

$$\textbf{error (step)}= \sum_{(u,i,c \in \kappa)}\left(r_{u_a i_b C_{kl}} - \mu - b_u - b_i - (P_{uk}Q_{ik}^T)\beta_{C_{kl}}\right)^2 + \lambda(\|P_{uk}\|^2_{Fro} + \|b_i\|^2 + \left\|\beta_{C_{kl}}\right\|^2 + \|Q_{ik}\|^2_{Fro} + \|b_u\|^2)$$

8) for each $R \in \kappa$

Compute $e \stackrel{\text{def}}{=} r_{u_a i_b C_{kl}} - \mu - b_u - b_i - (P_u Q_i^T)\beta_{C_{kl}}$

Update training parameters

$$b_u \longleftarrow b_u + \alpha_1(2e - \lambda b_u)$$

$$b_i \longleftarrow b_i + \alpha_1(2e - \lambda b_i)$$

$$P_u \longleftarrow P_u + \alpha_1(2eQ_{ik}^T\beta_{C_{kl}} - \lambda P_{uk})$$

$$Q_i \longleftarrow Q_i + \alpha_1(2eP_{uk}\beta_{C_{kl}} - \lambda Q_{ik})$$

$$\beta_{C_{kl}} \longleftarrow \beta_{C_{kl}} + \alpha_1(2eP_{uk}Q_i^T - \lambda\beta_{C_{kl}})$$

9) endfor

10) end

11) return P_{uk}, Q_{ik}, $\beta_{C_{kl}}$ b_u, and b_i

12) for each $R \notin \kappa$ predict the ratings for user and item

$$\hat{r}_{u_a i_b C_{kl}} = \mu + \boldsymbol{b_u} + \boldsymbol{b_i} + (\boldsymbol{P_{uk}Q_{ik}^T})\boldsymbol{\beta_{C_{kl}}}$$

6.3.4 Implementation in multidimensional contexts

The proposed context weighted matrix factorization (CWMF) is easily generalizable to multidimensional contexts. To implement the proposed algorithm in a multi-dimensional context, the first step is to decompose the multidimensional context to a one-dimensional

context for user- item sets. For each decomposed one-dimensional user-item-context a CWMF model is built and combined to generate prediction for the multidimensional context.

User	Item	Rating	Time	Day type	Season
U1	I1	5	Morning	Weekend	Summer
U1	I1	3	Evening	Weekday	summer
U1	I1	?	Afternoon	Weekend	winter

Table 10: Illustration of multidimensional context

The above table shows a part of a typical dataset of multi-dimensional contexts. The multi-dimensional contexts such as 'Time', 'Day type', and 'Season' impact the rating of a user for an item. One can see that in the given example, the task is to build a model so that it can predict the rating of user U1 for an item I1 when the contexts are 'Time', 'Day type', and 'Season' with the level of corresponding context as 'Afternoon', 'Weekend', and 'Winter'.

In order to implement the proposed CWMF model in the multidimensional context, it has first been decomposed to the multi-dimensional context into a one-dimensional context for a user item set. The above example can be decomposed into following three sub-datasets:

User	Item	Rating	Day type
U1	I1	5	Weekend
U1	I1	3	Weekday
U1	I1	?	Weekend

Table 12 (a)

User	Item	Rating	Time
U1	I1	5	Morning
U1	I1	3	Evening
U1	I1	?	Afternoon

Table 12 (b)

User	Item	Rating	Season
U1	I1	5	Summer
U1	I1	3	summer
U1	I1	?	winter

Table 12 (c)

Table 11: Decomposed multidimensional context dataset into one-dimensional contexts.

For obtaining the impact of multi-dimensional context, the proposed CWMF model can be applied to each of the decomposed sub-datasets, as illustrated above in a section of context weighted matrix factorization,, by blending or combining the parameters of generated models for each context. The blending of different algorithms has helped in improving the accuracy of standard recommender systems in previous works (Ekstrand, 2010; Jahrer et al., 2010; Koren, 2008, 2009; Takács et al., 2009; Vozalis & Margaritis, 2007). The main idea of blending is that the prediction vectors $\widehat{\boldsymbol{R}}_{C_k}$obtained from *k* algorithms ($\mathrm{CWMF}_1, \mathrm{CWMF}_2, \ldots., \mathrm{CWMF}_\mathrm{k}$) for every context $C_k \in \{C_1, C_2 \ldots, C_k\}$ are combined by a function $\varphi: R_{C_k} \rightarrow R_C$ so that the accuracy of the overall model is improved. This model uses the blending function as mean and linear ridge regression for combining the various models that predict the ratings in the multidimensional context.

Ridge regression can be obtained by solving the least square error with ridge regression constant (Jahrer et al., 2010). Given the input (regressor) vector **x**, the prediction is $\varphi(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{w}$. Weights $\boldsymbol{w}$ can be calculated with ridge regression, $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{R_C}$. Here, $\boldsymbol{I}$ is an identity matrix, '$\boldsymbol{R_C}$' is the output (predictor) vector and λ is a ridge constant. Ridge regression can be applied for the blending of different algorithms in various contexts by taking the regressor vector **x** as $\widehat{\boldsymbol{R}}_{C_k}$(output of CWMF_k) and the predictor vector $\boldsymbol{R_C}$ (rating of user-item-context interaction set).

6.4 Experiments and Evaluation

This section covers the experimental setup of the proposed context weighted matrix factorization which will be used to compare the state-of-the-art algorithms on a publicly available dataset. The dataset has already been used in reporting observations of context-aware algorithms in previous works. This work uses the proposed algorithms on the dataset, InCarMusic, which has non-overlapping context. This means that only one context exists for a given user-item pair at a particular time. The other dataset, the LDOS - CoMoDa dataset (Movie) has multidimensional contexts, where many contexts simultaneously impact the rating a user gives to a movie. The experimentation compares the accuracy levels of the proposed algorithm and previous algorithms applied on this dataset. The baseline algorithm is chosen as matrix factorization, which is a state-of-the- art algorithm in standard RS (Shi et al., 2010b).

Datasets

The dataset InCarMusic is collected from a mobile application (Android) offering music recommendations to the passengers of a car. The data was first used in developing a new algorithm for the context-aware recommender system (Baltrunas, Kaminskas, et al., 2011). There are 8 contextual factors and several contextual levels in each contextual factor. The lists of contextual factors and contextual levels are summarised in table 13. The next dataset, LDOS – CoMoDa, is a movie recommendation dataset and has 12 contextual factors and several contextual levels for each context. A summary of contextual factors and its levels are summarised in table 14.

Contextual factors	Contextual levels
DrivingStyle	relaxed driving, sport driving,

Landscape	coast line, country side, mountains/hills, urban
Mood	active, happy, lazy, sad
Natural phenomena	afternoon, day time, morning, night
Road Type	city, highway, serpentine
Sleepiness	awake, sleepy
Traffic Conditions	free road, lots of cars, traffic jam
Weather	cloudy, rainy, snowing, sunny

Table 12: InCarMusic dataset

Contextual factors	Contextual levels
time	Morning, Afternoon, Evening, Night
daytype	Working day, Weekend, Holiday
season	Spring, Summer, Autumn, Winter
location	Home, Public place, Friend's house
weather	Sunny / clear, Rainy, Stormy, Snowy, Cloudy
social	Alone, My partner, Friends, Colleagues, Parents, Public, My family
endEmo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
dominantEmo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
mood	Positive, Neutral, Negative
physical	Healthy, Ill
decision	User decided which movie to watch, User was given a movie
interaction	first interaction with a movie, n^{th} interaction with a movie

Table 13: LDOS – CoMoDa dataset

6.4.1 Evaluation measures

In order to evaluate accuracy, the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are popular metrics in Recommender systems. Since RMSE penalizes larger errors than MAE, it is preferred over MAE while evaluating the performance of RS (Shani & Gunawardana, 2011). RMSEs have been popular metrics in RS until very recently and many previous works have based their findings on these metrics; therefore, they have been used primarily to exhibit the performance of the proposed models and the MF model on the two datasets. For a test user item matrix 'Γ' the predicted rating $\hat{r}_{ui}$ for user-item pairs (u, i) for which the true item rating r_{ui} are known, the RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|\Gamma|} \sum_{(u,i \in \Gamma)} (\hat{r}_{ui} - r_{ui})^2}$$

MAE, on the other hand, is given by

$$\text{MAE}= \frac{1}{|I|} \sum_{(u,i \in I)} |\hat{r}_{ui} - r_{ui}|$$

Cross validation

Cross validation is a well-established technique in machine learning algorithms used for evaluation purposes. This technique ensures that the evaluation results are unbiased estimates and are not due to chance. The dataset is split into disjoint k-folds; (k-1) folds are used as a training set while the left-out set is used for testing. The procedure is repeated k times so that each time a unique test set can be used for performance evaluation. The measures such as RMSE and MAE for evaluation of RS models will be calculated k times and then averaged to get the resultant unbiased estimate of the performance measures (Bellogin et al., 2011).

6.4.2 Results

In order to compare the proposed model with the previous models, rigorous experimentations on 'InCarMusic' and 'LDOS – CoMoDa' datasets are conducted. For experimentation, the dataset is processed in a 5-fold cross-validation to assess the MAE and RMSE of the proposed model. The learning parameter (η) and hyper-parameter (λ) for the InCarMusic dataset are 0.01 and 0.05 respectively, while for the LDOS – CoMoDa dataset the learning parameter (η) and hyper-parameter (λ) are 0.01 and 0.1 respectively. The values of the hyper-parameters are obtained by cross validation (Baltrunas, Ludwig, et al., 2011; Parambath, 2013; Yu et al., 2009). The number of latent features (F) for InCarMusic and LDOS – CoMoDa datasets are varied from 5 to 50 in steps of 5.

As already mentioned in previous sections, InCarMusic dataset has non-overlapping contexts but the LDOS – CoMoDa dataset has overlapping multidimensional contexts. So, in the former dataset, every contextual factor can be a unique context and hence the proposed model can be applied directly for predicting ratings. The latter dataset has simultaneous multidimensional contexts; therefore, firstly decomposing the multidimensional context in the unique context and secondly, modelling each context impacting a user-item interaction and combining them by taking mean or ridge regression can make the rating prediction task viable.

For applying the proposed CWMF model on the InCarMusic dataset the five-fold cross validated MAE and RMSE are obtained by varying latent features (F) from 5 to 50 in steps of 5. The MAE and RMSE are plotted in figure 15. The best obtained MAE and RMSE from the CWMF model will be used for comparison from various models. The lowest MAE and RMSE ensue at the number of latent features (F) = 10.

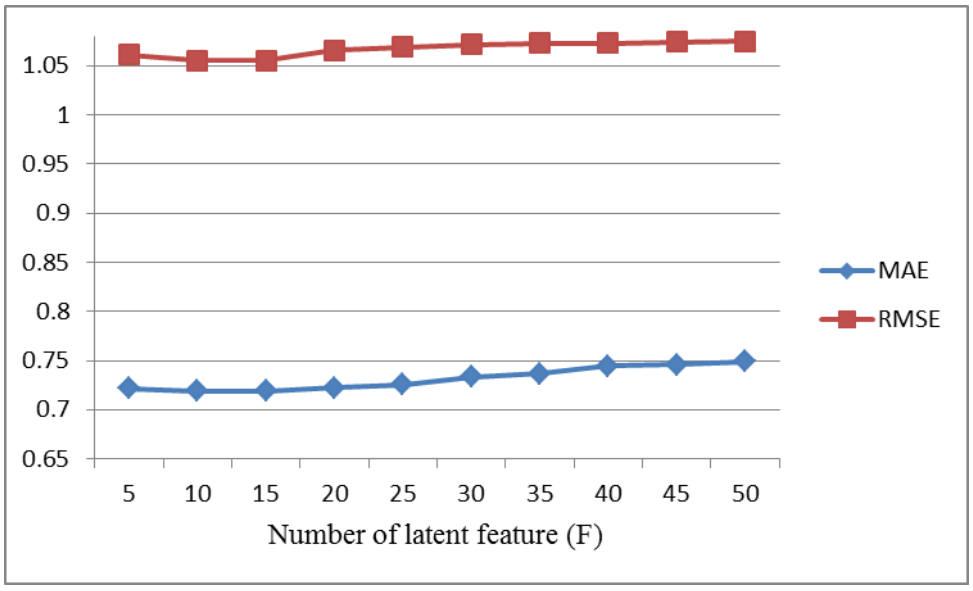


Figure 15: Variation of RMSE and MAE with change in number of latent features for CWMF model on InCarMusic dataset

The MAE values are as reported in the chapter for comparison with the previous model on the InCarMusic dataset (Baltrunas, Ludwig, et al., 2011). The previous approaches that applied the context- aware model on the dataset are referred to as the context- aware matrix factorization (CAMF), and it has three variants, namely, CAMF-C, CAMF-CC and CAMF-CI. The CAMF-C model's context is considered as the deviation in ratings due to global influence, CAMF-CI model's context as the deviation in ratings due to the impact of context on items, and CAMF-CC model's context as the deviation in rating due to impact on the genre of the music. The computed average MAE of 5-fold cross-validation using the proposed model is compared with the MAE reported in the chapter (Baltrunas, Ludwig, et al., 2011). The proposed model is also compared with MAE of the model obtained by using matrix factorization (without context) and item average (Item-Avg), which are the baselines for comparison. The MAE of all the above models in comparison with the proposed model for both the InCarMusic datasets are shown in table 15 and figure 16.

Performance measure	Dataset	Item-Avg	MF	CAMF-C	CAMF-CI	CAMF-CC	CWMF
MAE	InCarMusic	1.150	0.922	0.910	0.899	0.918	0.719

Table 14: Comparison of MAE on InCarMusic dataset with baseline models (Item-avg and MF), previous models (CAMF-C, CAMF-CI, CAMF-CC) and Proposed CWMF model

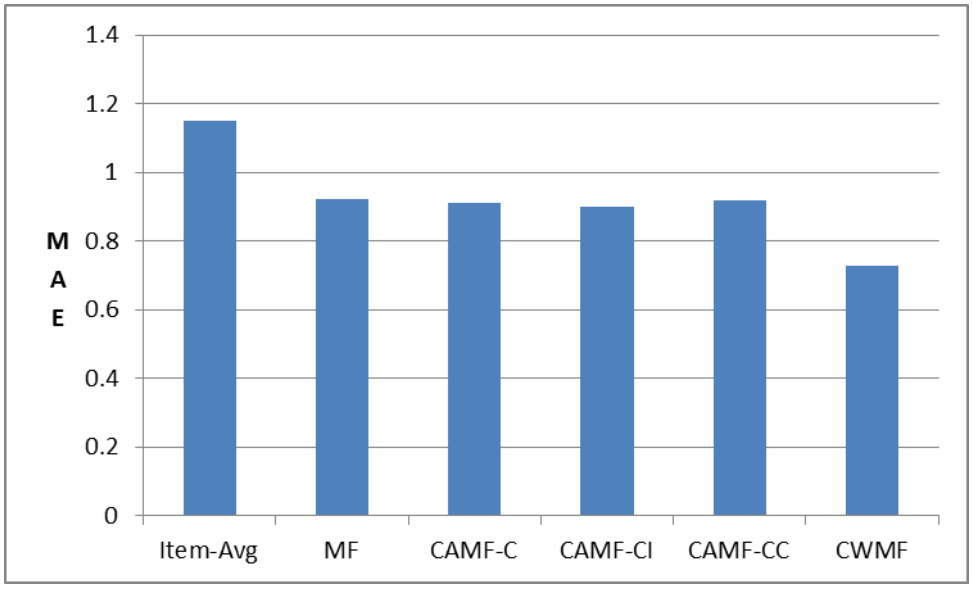


Figure 16: Mean Average Error of proposed CWMF and popular prediction models

As one can see from table 15, CWMF outperforms the previous model when compared in terms of MAE. While CAMF-CI, the best performing CAMF model, shows approximately 2.5% improvement in MAE over MF, the proposed CWMF shows an improvement of approximately 20% over MF and 18% over CAMF-CI on the InCarMusic dataset. This is certainly a remarkable improvement of the proposed model over other popular models as applied in CARSs.

In the case of the LDOS – CoMoDa dataset, the average of 5-fold cross validated MAE and RMSE is reported by considering each unique context and then these models are combined using mean and ridge regression to generate prediction in multi-dimensional contexts. On this dataset, the five-fold cross validated MAE and RMSE are obtained by varying the number of latent features (F) from 5 to 50 in steps of 5. The MAE and RMSE are plotted in figure 17.

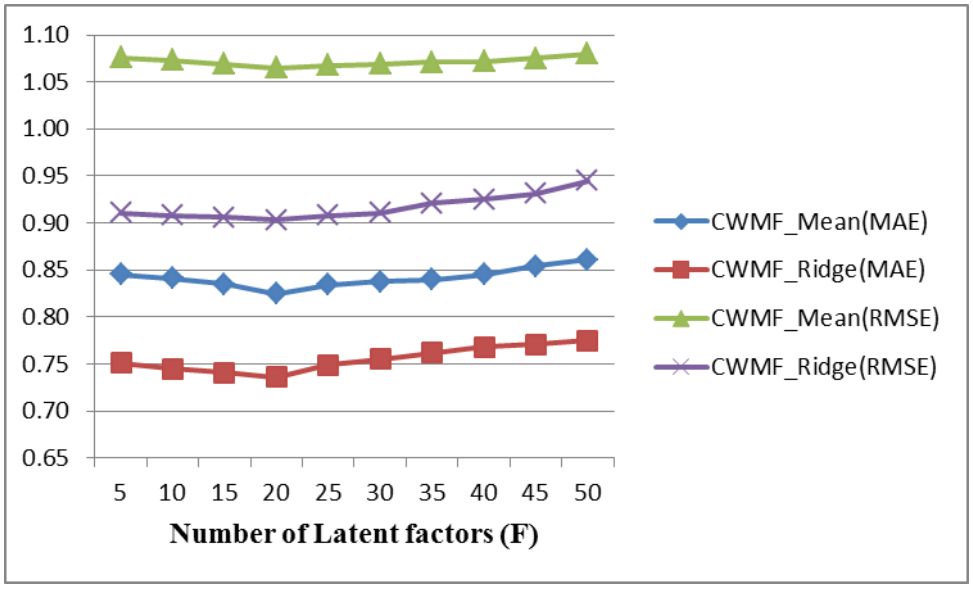


Figure 17: variation of RMSE and MAE with change in number of latent features for CWMF model on LDOS CoMoDa dataset

The best obtained MAE and RMSE from the CWMF model will be used for comparison with various models. The lowest MAE and RMSE happens at number of latent features (F) = 20. The MAE and RMSE values obtained by individual contexts and combinations for F=20 are reported in table 16.

Contextual factors	MAE	RMSE
time	0.865	1.124
daytype	0.869	1.128
season	0.853	1.103
location	0.871	1.134
weather	0.873	1.128
social	0.883	1.151
endEmo	0.800	1.033
dominantEmo	0.815	1.051
mood	0.878	1.127
physical	0.880	1.139
decision	0.882	1.138
interaction	0.887	1.132
Mean	0.825	1.065
Ridge regression	0.736	0.903

Table 15: Average of five-fold cross validated MAE and RMSE values on decomposed contexts of LDOS-CoMoDa dataset using proposed CWMF model

Table 16, as given above, shows the different values of prediction performance (MAE and RMSE) obtained by taking each contextual factor, one at time, into account and applying

the proposed CWMF model. The context that shows the best prediction performance on the LDOS CoMoDa dataset is 'emotions' (endEmo, dominantEmo) among all the other contexts. This finding goes in line with another work (Zheng, Mobasher, & Burke, 2013) on the same dataset, which developed a model in CARSs to predict the ratings based only on emotions. Since the prediction of rating is based on account of all the contexts, the different prediction models are combined using mean and ridge regression.

Now, to compare the proposed model with the previous models on the same dataset, the measures reported in previous works have been applied (Zheng, Burke, & Mobasher, 2013; Odić et al., 2013). Therefore, only RMSE, not MAE, has been used to compare the proposed model with existing models. CAMF-C, CAMF-CI, and CAMF-CU have also been applied on this dataset that improve the RMSE over MF and item-average (Zheng, Burke, et al., 2013). It is to be noted that instead of CAMF-CC, that models the genre-context interaction, CAMF-CU has been applied in the LDOS-CoMoDa dataset. CAMF-CU models the interaction of the user and context for prediction of ratings. Another approach referred to as contextualized user bias matrix factorization (CUB-MF), which models user bias influenced by the context, performs better on this dataset than MF and item-average (Odić et al., 2013). The comparison of RMSE using baseline models (Item-Avg and MF), previous models (CAMF-C, CAMF-CI, CAMF-CU, CUB-MF) and Proposed CWMF model is shown in table 17 and figure 23.

Performance measure	Dataset	Item -Avg	MF	CAMF -C	CAM F-CI	CAM F-CU	CUB-MF*	CWMF +Mean	CWMF +Ridge
RMSE	LDOS – CoMoDa	1.01	0.99	1.012	1.032	0.932	0.885	1.065	0.905

Table 16: Comparison of RMSE on LDOS-_CoMoDa dataset with baseline models (Item-avg and MF), previous models (CAMF-C, CAMF-CI, CAMF-CU, CUB-MF) and Proposed CWMF model

*CUB-MF first categorizes relevant and non-relevant contexts and then predicts the ratings. The RMSE value reported is the best obtained on this dataset among relevant contexts; however, the RMSE varies from 0.885 to 1.01 with different relevant contexts.

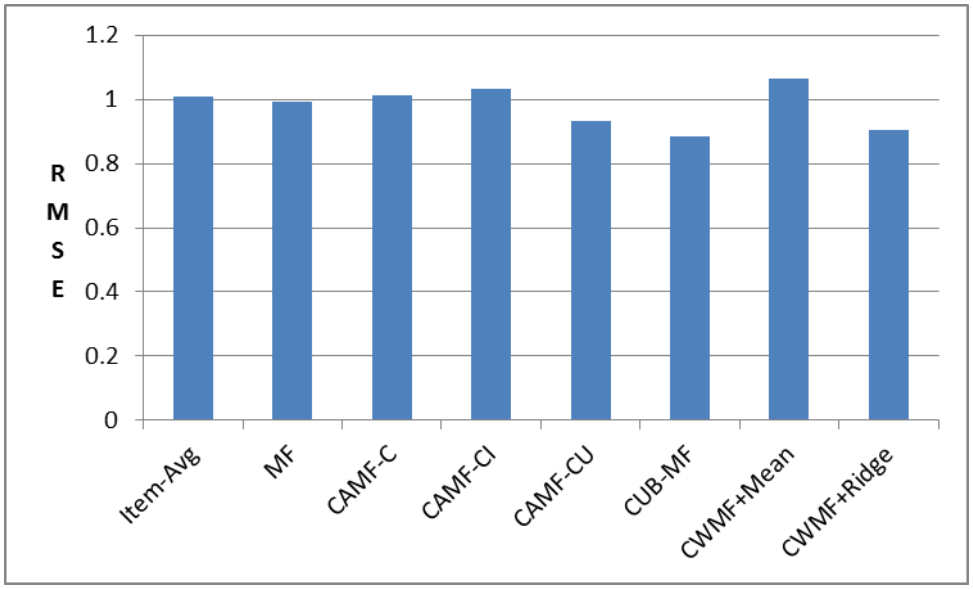


Figure 18: Root Mean Square Error (RMSE) of proposed CWMF and popular prediction models

The proposed CWMF model has been applied on the multidimensional context by a combination of linear ridge regression and mean. Mean is used for the combining of models as it is the simplest method of combination and has been used in RS literature. The RMSE obtained by combination using mean does not improve the predictive performance but in fact worsens it more than the baseline models. However, a combination using linear ridge regression improves the prediction performance by approximately 8.89% over the MF model and approximately 3% over the CAMF-CU model.

The best prediction model on this dataset is the CUB-MF model, and the proposed CWMF model lags by approximately 2%. One of the major drawbacks of the CUB-MF model is that finding relevant context can be a challenging task until there are sufficient data points in each context to find the relevant context. Another caveat in applying the CUB-MF

model is that even the obtained relevant context can give a range of RMSE values for different relevant contexts, which vary between 0.885 (for emotion) and 1.01(for interaction). The proposed CWMF predicts the ratings based on all the contexts and the combination using ridge regression ensures that all the contexts of the dataset are adjusted to the weights according to the dataset automatically, without finding the relevant context. This ability to predict better ratings for unseen items considering all the contexts gives an advantage with previous models in CARSs.

6.5 Discussions and conclusions

This chapter has proposed an optimization based context weighted matrix factorization approach to personalize decision support systems. The notion of context awareness is the main feature of this work. The chapter captures the interaction amongst user, item, and context in a novel and meaningful way, which is a key ingredient of this work. The proposed CWMF model has been developed to capture the influence of a single-dimension context on a user's choice for an item. The proposed model has also been extended to model the interplay of multiple contexts on the user's choice for an item by combining models obtained on decomposed single-dimension contextual data. The combination of different models based on single-dimension context is obtained by using blending techniques like the computation of mean and linear ridge regression. The proposed algorithm is based on a well-established theoretical grounding of matrix factorization, linear regression, and a combination of models applied in various domains demonstrating robustness.

This chapter primarily focuses on predicting ratings of items which are not yet seen by a user in varying context. The performance measures like MAE and RMSE captures the error subtlety in rating prediction task and are widely used in RS literature. In this chapter, the objective has been to model context, user, and item interaction at the finer, granular level of

context, but this can well be extended to model the impact of various context-context interactions on user choice for an item. In future work, the proposed model can be extended to formulate interactions between the contexts that may impact a user's choice. In this chapter, the prediction for multi-dimensional context has been shown using a computation of mean and linear ridge regression; however, a more sophisticated technique of ensembles like boosting and bagging can also be implemented for a combination of models. It is recommended in the future direction to develop algorithms that recommend the top-n items for every user based on his purchase behaviour in different contexts.

7 Conclusion and scope of future research

This dissertation revolves around three broad themes in the area of recommendation systems. The first theme, chapters 3 and 4, relates to maximizing the precision of top-n recommendations and further extending it to the precise ranking of recommended items. The second theme, chapter 5, takes into account the dilemma of accuracy and diversity/novelty in the recommendation task. Lastly, the third theme, chapter 6, proposes a new model of incorporating contexts in the field of recommender systems.

7.1 Conclusion

The analysis of real-world rating datasets reveals useful data patterns which aided in formulating new recommendation approaches. The first theme of this dissertation proposes a cosine based latent factor model that seeks to improve the prediction of top-n recommendations. The major contribution in the first theme is the incorporation of bounded loss function in the form of cosine loss function, which improves the precision of the recommendations. The cosine loss function naturally smooths any outbound prediction which results in a robust prediction of the recommendation list. When measured on two benchmark datasets the proposed model outperforms state-of- the- art RSVD models in terms of precision. The proposed model, when applied for rating prediction, by computing k-similar neighbours based on similarity of latent factors, matches the accuracy obtained using a state-of-the-art RSVD model on both the datasets. The usefulness of the cosine loss function has also been extended in ranking to check for its robustness. In the ranking prediction task, the cosine loss function determines the latent factors of users and items, and list wise loss functions, viz., cosine based permutation probability and list wise loss, a.k.a. the Plackett-Luce model. List wise likelihood loss and cosine loss are aptly chosen to act as surrogate loss to learn for the ranking task. The cosine based latent factor model learns latent features from

the dataset which eventually augurs well with both the surrogate losses. Interestingly, the proposed algorithms on ranking tasks also perform well when measured on classification metrics. Therefore, the ranking task improves not only the positions of the items in the recommended list, but more relevant items also appear in the recommendation list as compared to the models used for rating prediction and classification task. Since e-commerce is now shifting to m-commerce, the use of ranking methods in recommendation systems are witnessing a growing interest due to the constrained recommendation list that may appear on mobile platforms.

The second theme of this dissertation proposes a unique model in handling the accuracy and diversity/novelty dilemma. Diversification of recommendation is imminent as it helps to escape from the 'filter bubble'(Knijnenburg, Sivakumar, & Wilkinson, 2016). The 'filter bubble' is a popular term used for recommenders which may isolate us from a diversity of viewpoints, contents, and experiences, and thus make us less likely to discover and learn new things. Diversifying the recommendation can prevent this but the main shortcoming of diversification is that it depends on the system's interpretation of diversity rather than the users'(Knijnenburg et al., 2016). For applying diversification from users' perspective, the second theme takes into account the disposition of all users towards a diverse/novel set of items and then suitably recommends each user with a unique set of accurate and diverse/novel items. There are a few algorithms which try to balance between accuracy and diversity/novelty; however, the proposed approach is different in two aspects, 1) the concept of varying degree of novelty that a user can possess has been introduced in this work, and 2) a flexible approach that tunes accuracy and diversity/novelty has also been introduced in the proposed model. The proposed models demonstrated on two benchmark datasets reveal that diversity and accuracy are inversely related and so are accuracy and novelty. The findings are in line with the previous research works in this area (ref). The models have utility in

promoting a diverse/novel set of items as desired without any noticeable loss in accuracy. The proposed models offer the scope of deployment to leverage strategic business advantage for online business platforms. The proposed model, PM-1, guides the users to choose from a recommendation list based on the implicit taste of the long-tail items to users. Since this will help users to navigate to niche items of their choice, thereby reducing the search time for suitable items, it increases their trust towards such recommender systems. This may in turn generate a 'recommendation-buying' spiral cycle which means that users relying on a recommender system will tend to buy from the recommendation list that will effectively improve the PM-1 with more information available about the users' behaviors. The proposed model PM-2, on the other hand, can prove to be an expedient tool to promote long-tail, diverse, and precise items with flexibility to those users who have a better chance to like such items based on their past purchase behavior. Unexpected, yet useful recommendations will indeed help e-commerce sellers to utilize their unlimited shelf space, which is a major advantage over brick and mortar shops.

The third theme incorporates context in the recommender systems models. Contexts are important facets in shaping the improved user experience (Knijnenburg, Willemsen, Gantner, Soncu, & Newell, 2012). The proposed model incorporates contexts by weighing corresponding context with respect to each user-item interaction. The concept of weighing each context results in lesser complexity of the model than in a tensor factorization model. The proposed CWMF on two benchmark datasets reveals that the accuracy is as good as in a tensor factorization model. CWMF has been primarily designed for unidimensional context where it is assumed that only one relevant context exists for each user-item interaction. However, the proposed model has been extended to the multi-dimensional context by applying an ensemble of various unidimensional contexts. To substantiate the results the

experimentation has been carried out on both types of datasets, viz., unidimensional context and multidimensional context,.

7.2 Scope of future research

Traditionally, RS have focussed on developing models that are accurate but this has recently taken a major shift towards enhancing user experience. To enhance user experiences, the emphasis of RS is on the diversity of choices, incorporating contextual characteristics, interactive recommender system, etc. While user experience is an important feature of RS, future research can reveal the business value of diverse and accurate recommendations empirically. A few researches have investigated the social and economic impact of RS in general; however, a specific experimental study of the impact of accurate items based on ranking or the dilemma of accurate versus diverse set of items can be investigated in detail. A related extension could be to investigate the suitability of accurate, diverse/novel recommendations in various domains of e-commerce; for example, the movie domain may be better served by applying accurate recommendations but the clothes and jewellery domain may be better served by applying diverse recommendations.

Further, this dissertation has applied the time factor as a dimension in diversity seeking recommendation which can also be extended to the context- aware recommendation system. The basic concept underlying here is the evolution of customers with time, as is discussed in previous research works. With regards to generating contexts, they can be enriched in a non-obtrusive way by exploiting new sensors, particularly with the advancement of smart phones or tablets. In addition, developing richer interaction capabilities for contextual recommender systems are needed. The current black box nature of recommender systems prevents users to provide input into the recommendation process in an interactive and iterative manner. As the

predication of the current task or interest of the user is a challenge, there is a need to develop mixed approaches that enable users to help steer this process.

References

Adamopoulos, P., & Tuzhilin, A. (2015). On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology (TIST), 5(4),* 54.

Adolphs, C., & Winkelmann, A. (2010). Personalization research in E-commerce-a state of the art review (2000-2008). *Journal of Electronic Commerce Research, 11(4),* 326.

Adomavicius, G., & Kwon, Y. (2008). Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach. In Proceedings of the *18th workshop on information technology and systems (WITS'08),* Paris: *CONF, Citeseer*.

Adomavicius, G., & Kwon, Y. (2011). Maximizing aggregate recommendation diversity: A graph-theoretic approach. In Proceedings of the *1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011)* (pp. 3–10). *CONF, Citeseer*.

Adomavicius, G., & Kwon, Y. (2012). Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering, 24(5),* 896–911.

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating

contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS), 23(1)*, 103–145.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17(6)*, 734–749.

Agarwal, D., & Chen, B.C. (2009). Regression-based latent factor models. In Proceedings of the *15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 19–28). *CONF, ACM*.

Ahmed, A., Kanagal, B., Pandey, S., Josifovski, V., Pueyo, L. G., & Yuan, J. (2013). Latent factor models with additive and hierarchically-smoothed user preferences. In Proceedings of the *Sixth ACM international conference on Web search and data mining* (pp. 385–394). *CONF, ACM*.

Anand, D., & Bharadwaj, K. K. (2011). Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. *Expert Systems with Applications, 38(5)*, 5101–5109.

Anderson, C. (2007). *The long tail: how endless choice is creating unlimited demand*. US: Random House Business Books.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM Press.

Balakrishnan, S., & Chopra, S. (2012). Collaborative ranking. In Proceedings of the *Fifth ACM international conference on Web search and data mining* (pp. 143–152). *CONF, ACM.*

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., … Schwaiger, R. (2011). Incarmusic: Context-aware music recommendations in a car. In *International Conference on Electronic Commerce and Web Technologies* (pp. 89–100). *CONF, Springer*.

Baltrunas, L., Ludwig, B., & Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In Proceedings of the *Fifth ACM conference on Recommender systems* (pp. 301–304). *CONF, ACM*.

Bauer, J., & Nanopoulos, A. (2014). Recommender systems based on quantitative implicit customer feedback. *Decision Support Systems, 68*, 77–88.

Bell, R. M., & Koren, Y. (2007). Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)* (pp. 43–52). *CONF, IEEE*.

Bellogin, A., Castells, P., & Cantador, I. (2011). Precision-oriented evaluation of recommender systems: an algorithmic comparison. In Proceedings of the *Fifth ACM conference on Recommender systems* (pp. 333–336). *CONF, ACM*.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. In *ICML* (Vol. 98, pp. 46–54). *CONF*.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3(Jan)*, 993–1022.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132.

Bose, I., & Chen, X. (2009). Quantitative models for direct marketing: A review from

systems perspective. *European Journal of Operational Research, 195*(1), 1–16..

Brynjolfsson, E., Hu, Y., & Simester, D. (2011). Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science, 57(8),* 1373–1386.

Brynjolfsson, E., Hu, Y., & Smith, M. D. (2010). Research commentary-long tails vs. superstars: The effect of information technology on product variety and sales concentration patterns. *Information Systems Research, 21*(4), 736–747.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In Proceedings of the *22nd International Conference on Machine Learning* (pp. 89–96).

Burke, R. R. (2002). Technology and the customer interface: What consumers want in the physical and virtual store. *Journal of the Academy of Marketing Science, 30*(4*),* 411–432.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. In Proceedings of the *24th International Conference on Machine Learning* (pp. 129–136).

Castells, P., Vargas, S., & Wang, J. (2011). Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. (pp. 109-116) *CONF, ACM*

Champiri, Z. D., Shahamiri, S. R., & Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications, 42(3),* 1743–1758.

Chen, I. J., & Popovich, K. (2003). Understanding customer relationship management (CRM)

People, process and technology. *Business Process Management Journal, 9(5),* 672–688.

Chen, L., Wu, W., & He, L. (2013). How personality influences users' needs for recommendation diversity? In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 829–834).

Chen, W., Liu, T.-Y., Lan, Y., Ma, Z.-M., & Li, H. (2009). Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems* (pp. 315–323).

Cheung, K.-W., Kwok, J. T., Law, M. H., & Tsui, K.-C. (2003). Mining customer product ratings for personalized marketing. *Decision Support Systems, 35(2),* 231–243.

Cho, Y. H., Kim, J. K., & Kim, S. H. (2002). A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications, 23(3),* 329–342.

Chung, C.-Y., Hsu, P.-Y., & Huang, S.-H. (2013). βP: A novel approach to filter out malicious rating profiles from recommender systems. *Decision Support Systems, 55(1),* 314–325.

Clemons, E. K., & Nunes, P. F. (2011). Carrying your long tail: Delighting your consumers and managing your operations. *Decision Support Systems, 51(4),* 884–893.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of the *Fourth ACM conference on Recommender systems* (pp. 39–46).

Dali, L., Fortuna, B., & Rupnik, J. (2010). Learning to Rank for Personalized News Article Retrieval. In *WAPA* (pp. 152–159). Dao, T. H., Jeong, S. R., & Ahn, H. (2012). A novel

recommendation model of location-based advertising: Context-aware collaborative filtering using GA approach. *Expert Systems with Applications, 39(3)*, 3731–3739.

Das, A. S., Datar, M., Garg, A., & Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In Proceedings of the *16th international conference on World Wide Web* (pp. 271–280). *CONF, ACM*.

De Pessemier, T., Deryckere, T., & Martens, L. (2009). Context aware recommendations for user-generated content on a social network site. In Proceedings of the *Seventh European conference on European interactive television conference* (pp. 133–136). *CONF, ACM*.

De Pessemier, T., Deryckere, T., & Martens, L. (2010). Extending the Bayesian classifier to a context-aware recommender system for mobile devices. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference* on (pp. 242–247). *CONF, IEEE*.

Deng, X., & Wang, X. (2009). Mining rank-correlated associations for recommendation systems. In. *WISM 2009. International Conference* on (pp. 625–629). *CONF, IEEE*.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5(1)*, 4–7.

Doukidis, G. I., Pramatari, K., & Lekakos, G. (2008). OR and the management of electronic services. *European Journal of Operational Research, 187(3)*, 1296–1309.

Drosou, M., & Pitoura, E. (2009). Diversity over continuous data. *IEEE Data Eng. Bull., 32(4)*, 49–56.

du Boucher-Ryan, P., & Bridge, D. (2006). Collaborative recommending using formal concept analysis. *Knowledge-Based Systems, 19(5)*, 309–315.

Eirinaki, M., & Vazirgiannis, M. (2003). Web mining for web personalization. *ACM Transactions on Internet Technology (TOIT), 3(1)*, 1–27.

Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction, 4(2)*, 81–173.

Elberse, A., & Oberholzer-Gee, F. (2006). Superstars and underdogs: An examination of the long tail phenomenon in video sales. *Harvard Business School Working Paper, No. 07-015*

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research, 4(Nov)*, 933–969.

Gan, M., & Jiang, R. (2013). Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities. *Decision Support Systems, 55(3)*, 811–821.

Gao, M., Liu, K., & Wu, Z. (2010). Personalisation in web computing and informatics: Theories, techniques, applications, and future research. *Information Systems Frontiers, 12(5)*, 607–629.

Ge, M., Delgado-Battenfeld, C., & Jannach, D. (2010). Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In Proceedings of the *Fourth ACM conference on Recommender systems* (pp. 257–260). *CONF, ACM.*

Ghosh, R., & Lerman, K. (2010). Predicting influential users in online social networks. *arXiv Preprint arXiv*:*1005.4882*.

Goel, S., Broder, A., Gabrilovich, E., & Pang, B. (2010). Anatomy of the long tail: Ordinary people with extraordinary tastes. In Proceedings of the *Third ACM international*

conference on Web search and data mining (pp. 201–210).

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35(12)*, 61–70.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4(2)*, 133–151.

Goldstein, D. G., & Goldstein, D. C. (2006). Profiting from the long tail. *Harvard Business Review*, *84*(6), 24–28.

Gunawardana, A., & Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research, 10*(Dec), 2935–2962.

Guo, G., Zhang, J., & Yorke-Smith, N. (2013). A novel Bayesian similarity measure for recommender systems. In *IJCAI:*

Hariri, N., Mobasher, B., & Burke, R. (2012). Context-aware music recommendation based on latent topic sequential patterns. In Proceedings of the *Sixth ACM conference on Recommender systems* (pp. 131–138).

Harrington, E. F. (2003). Online ranking/collaborative filtering using the perceptron algorithm. In *ICML (Vol. 20*, pp. 250–257).

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In Proceedings of the *22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 230–237).

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating

collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS), 22*(1), 5–53.

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In Proceedings of the *SIGCHI conference on Human factors in computing systems* (pp. 194–201). Addison-Wesley Publishing Co.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS), 22*(1), 89–115.

Hofmann, T., & Hartmann, D. (2005). Collaborative filtering with privacy via factor analysis. In Proceedings of the *2005 ACM Symposium on Applied Computing* (pp. 791–795).

Hofmann, T., Puzicha, J., & Jordan, M. I. (1999). Learning from dyadic data. *Advances in Neural Information Processing Systems*, 466–472.

Hong, J., Suh, E., & Kim, S.-J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications, 36*(4), 8509–8522.

Hong, J., Suh, E.-H., Kim, J., & Kim, S. (2009). Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications, 36*(4), 7448–7457.

Hosanagar, K., Fleder, D., Lee, D., & Buja, A. (2013). Will the global village fracture into tribes? *Recommender Systems and Their Effects on Consumer Fragmentation. Management Science, 60*(4), 805–823.

Hosseini-Pozveh, M., Nematbakhsh, M., & Movahhedinia, N. (2009). A multidimensional approach for context-aware recommendation in mobile commerce. *arXiv Preprint arXiv:0908.0982*.

Hostler, R. E., Yoon, V. Y., & Guimaraes, T. (2012). Recommendation agent impact on consumer online shopping: The Movie Magic case study. *Expert Systems with Applications*, *39*(3), 2989–2999.

Hu, R., & Pu, P. (2011). Helping users perceive recommendation diversity. In *Workshop on Novelty and Diversity in Recommender Systems, DiveRS*. Chicago.

Huang, J., Zhong, N., & Yao, Y. (2014). A unified framework of targeted marketing using customer preferences. *Computational Intelligence*, *30*(3), 451–472.

Izadi, M., Javari, A., & Jalilii, M. (2014). Unifying inconsistent evaluation metrics in recommender systems. In *RecSys conference, REDD workshop*.

Jahrer, M., Töscher, A., & Legenstein, R. (2010). Combining predictions for accurate recommender systems. In Proceedings of the *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 693–702).

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 422–446.

Jawaheer, G., Weller, P., & Kostkova, P. (2014). Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *4*(2), 8.

Joachims, T. (2002). Optimizing search engines using click through data. In Proceedings of the *eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 133–142).

Jung, J. J. (2012). ContextGrid: A contextual mashup-based collaborative browsing system. *Information Systems Frontiers*, *14*(4), 953–961.

Kagie, M., Van Der Loos, M., & Van Wezel, M. (2009). Including item characteristics in the probabilistic latent semantic analysis model for collaborative filtering. *Ai Communications*, *22*(4), 249–265.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the *Fourth ACM Conference on Recommender Systems* (pp. 79–86).

Karatzoglou, A., Baltrunas, L., & Shi, Y. (2013). Learning to rank for recommender systems. In Proceedings of the *7th ACM Conference on Recommender Systems* (pp. 493–494).

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In Proceedings of the *Tenth International Conference on Information and Knowledge Management* (pp. 247–254).

Kelleher, J., & Bridge, D. (2004). An accurate and scalable collaborative recommender. *Artificial Intelligence Review*, *21*(3–4), 193–213.

Kim, Y. S., Yum, B.J., Song, J., & Kim, S. M. (2005). Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Systems with Applications*, *28*(2), 381–393.

Knijnenburg, B. P., Sivakumar, S., & Wilkinson, D. (2016). Recommender systems for self-actualization. In Proceedings of the *10th ACM Conference on Recommender Systems* (pp. 11–14).

Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., & Newell, C. (2012). Explaining the user experience of recommender systems. *User Modeling and User-*

Adapted Interaction, 22(4–5), 441–504.

Konstan, J. A., McNee, S. M., Ziegler, C.-N., Torres, R., Kapoor, N., & Riedl, J. (2006). Lessons on applying automated recommender systems to information-seeking tasks. In *Association for the Advancement of Artificial Intelligence (AAAI)* (Vol. 6, pp. 1630–1633).

Konstan, J. A., & Riedl, J. (2012). Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction, 22*(1–2), 101–123.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the *14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining* (pp. 426–434).

Koren, Y. (2009). The Bellkor solution to the Netflix grand prize. *Netflix Prize Documentation, 81*, 1–10.

Koren, Y., & Bell, R. (2011). Advances in collaborative filtering. In *Recommender systems handbook* (pp. 145–186). CHAP: Springer.

Koren, Y., & Sill, J. (2011). OrdRec: An ordinal model for predicting personalized item rating distributions. In Proceedings of the *Fifth ACM conference on Recommender systems* (pp. 117–124).

Krestel, R., Fankhauser, P., & Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In Proceedings of the *Third ACM conference on Recommender systems* (pp. 61–68).

Kumar, R., Raghavan, P., Rajagopalan, S., & Tomkins, A. (1998). Recommendation systems: A probabilistic analysis. In Proceedings of *Foundations of Computer Science, 1998.*

39th Annual Symposium on (pp. 664–673).

Kwon, O. (2006). The potential roles of context-aware computing technology in optimization-based intelligent decision-making. *Expert Systems with Applications, 31*(3), 629–642.

Kwon, O. B. (2003). "I know what you need to buy": Context-aware multimedia-based recommendation system. *Expert Systems with Applications, 25*(3), 387–400.

Kwon, O., & Kim, J. (2009). Concept lattices for visualizing and generating user profiles for context-aware service recommendations. *Expert Systems with Applications, 36*(2), 1893–1902.

Lathia, N., Hailes, S., Capra, L., & Amatriain, X. (2010). Temporal diversity in recommender systems. In Proceedings of the *33rd international ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 210–217).

Lee, J., Bengio, S., Kim, S., Lebanon, G., & Singer, Y. (2014). Local collaborative ranking. In Proceedings of the *23rd International Conference on World Wide Web* (pp. 85–96).

Lee, J. S., & Lee, J. C. (2007). Context awareness by case-based reasoning in a music recommendation system. In *International Symposium on Ubiquitous Computing Systems* (pp. 45–58). Springer.

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology, 58*(7), 1019–1031.

Liu, D.-R., & Liou, C.-H. (2011). Mobile commerce product recommendations based on hybrid multiple channels. *Electronic Commerce Research and Applications, 10*(1), 94–

104.

Liu, J., Wu, C., Xiong, Y., & Liu, W. (2014). List-wise probabilistic matrix factorization for recommendation. *Information Sciences, 278*, 434–447.

Liu, N. N., & Yang, Q. (2008). Eigenrank: A ranking-oriented approach to collaborative filtering. In Proceedings of the *31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 83–90).

Liu, Q., Chen, E., Xiong, H., Ding, C. H. Q., & Chen, J. (2012). Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42*(1), 218–233.

Liu, X., & Aberer, K. (2014). Towards a dynamic top-n recommendation framework. In Proceedings of the *8th ACM Conference on Recommender systems* (pp. 217–224).

Lussier, D. A., & Olshavsky, R. W. (1979). Task complexity and contingent processing in brand choice. *Journal of Consumer Research, 6*(2), 154–165.

Maes, P. (1999). Smart commerce: The future of intelligent agents in cyberspace. *Journal of Interactive Marketing, 13*(3), 66.

Marlin, B. M. (2003). Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems* (p. None).

McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems* (pp. 1097–1101).

Meeds, E., Ghahramani, Z., Neal, R. M., & Roweis, S. T. (2006). Modeling dyadic data with

binary latent factors. In *Advances in Neural Information Processing Systems* (pp. 977–984).

Miyahara, K., & Pazzani, M. J. (2002). Improvement of Collaborative Filtering with the Simple Bayesian Classifier 1. *IPSJ, 43*(11)

Mulvenna, M. D., Anand, S. S., & Büchner, A. G. (2000). Personalization on the Net using Web mining: Introduction. *Communications of the ACM, 43*(8), 122–125.

Nguyen, T. V, Karatzoglou, A., & Baltrunas, L. (2014). Gaussian process factorization machines for context-aware recommendations. In Proceedings of the *37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 63–72).

Niemann, K., & Wolpers, M. (2013). A new collaborative filtering approach for increasing the aggregate diversity of recommender systems. In Proceedings of the *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 955–963).

Ning, X., & Karypis, G. (2011). Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining* (pp. 497–506).

Odić, A., Tkalčič, M., Tasič, J. F., & Košir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers, 25*(1), 74–90.

Oestreicher-Singer, G., & Sundararajan, A. (2010). Recommendation networks and the long tail of electronic commerce. Available at *SSRN 1324064*. https://www.ssrn.com/

Oku, K., & Hattori, F. (2011). Fusion-based recommender system for improving serendipity.

In *Proceedings of the Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011)*, at the *5th ACM International Conference on Recommender Systems (RecSys 2011)* (p. 19).

Olafsson, S., Li, X., & Wu, S. (2008). Operations research and data mining. *European Journal of Operational Research, 187*(3), 1429–1448.

Ono, C., Kurokawa, M., Motomura, Y., & Asoh, H. (2007). A context-aware movie preference model using a Bayesian network for recommendation and promotion. In *International Conference on User Modeling* (pp. 247–257). Springer.

Pahikkala, T., Stock, M., Airola, A., Aittokallio, T., De Baets, B., & Waegeman, W. (2014). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 517–532). Springer.

Palmisano, C., Tuzhilin, A., & Gorgoglione, M. (2008). Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering, 20*(11), 1535–1549.

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications, 39*(11), 10059–10072.

Park, S.-H., & Han, S. P. (2013). From accuracy to diversity in product recommendations: Relationship between diversity and customer retention. *International Journal of Electronic Commerce, 18*(2), 51–72.

Park, Y.-J., & Tuzhilin, A. (2008). The long tail of recommender systems and how to

leverage it. In Proceedings of the *2008 ACM Conference on Recommender Systems* (pp. 11–18).

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In Proceedings of *KDD Cup workshop* (Vol. 2007, pp. 5–8).SIGKDD.

Pennock, D. M., Horvitz, E., Lawrence, S., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In Proceedings of the *Sixteenth Conference on Uncertainty in Artificial Intelligence* (pp. 473–480). Morgan Kaufmann Publishers, Inc.

Pielou, E.C. (1966) The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology. 13*, 131-144.

Prahalad, C. K. (2004). Beyond CRM: CK Prahalad predicts customer context is the next big thing. *American Management Association Mw World*.

Pu, P., Chen, L., & Kumar, P. (2008). Evaluating product search and recommender systems for E-commerce environments. *Electronic Commerce Research, 8*(1–2), 1–27.

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2008). Query-level loss functions for information retrieval. *Information Processing & Management, 44*(2), 838–855.

Ramaswamy, L., Deepak, P., Polavarapu, R., Gunasekera, K., Garg, D., Visweswariah, K., & Kalyanaraman, S. (2009). Caesar: A context-aware, social recommender system for low-end mobile devices. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware* (pp. 338–347). IEEE.

Ranjan, J., & Bhatnagar, V. (2009). A holistic framework for mCRM-data mining perspective. *Information Management & Computer Security, 17*(2), 151–165.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the *twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452–461). AUAI Press.

Rendle, S., Gantner, Z., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In Proceedings of the *34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 635–644).

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the *22nd International Conference on Machine Learning* (pp. 713–719).

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of the *1994 ACM Conference on Computer Supported Cooperative Work* (pp. 175–186).

Ribeiro, M. T., Lacerda, A., Veloso, A., & Ziviani, N. (2012). Pareto-efficient hybridization for multi-objective recommender systems. In Proceedings of the *Sixth ACM Conference on Recommender Systems* (pp. 19–26).

Russell, S., & Yoon, V. (2008). Applications of wavelet data reduction in a recommender system. *Expert Systems with Applications, 34*(4), 2316–2325.

Said, A., Kille, B., Jain, B. J., & Albayrak, S. (2012). Increasing diversity through furthest neighbor-based recommendation. Proceedings of the *WSDM, 12 Conference Series*.

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In Proceedings of the *25th International Conference on Machine Learning* (pp. 880–887).

Salakhutdinov, R., & Mnih, A. (2011). Probabilistic matrix factorization. In *NIPS* (Vol. 20, pp. 1–8).

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In Proceedings of the *24th International Conference on Machine Learning* (pp. 791–798).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000a). Analysis of recommendation algorithms for e-commerce. In Proceedings of the *2nd ACM Conference on Electronic Commerce* (pp. 158–167).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000b). Application of dimensionality reduction in recommender system-a case study (RPRT). *DTIC Document*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science* (pp. 27–28). Citeseer.

Shafer, S. A. N., Brumitt, B., & Cadiz, J. J. (2001). Interaction issues in context-aware intelligent environments. *Human-Computer Interaction, 16*(2), 363–378.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender Systems Handbook* (pp. 257–297). CHAP, Springer.

Shani, G., Heckerman, D., & Brafman, R. I. (2005). An MDP-based recommender system. *Journal of Machine Learning Research, 6*(Sep), 1265–1295.

Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth." In Proceedings of the *SIGCHI Conference on Human Factors in Computing Systems* (pp. 210–217). Press/Addison-Wesley Publishing Co.

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., & Oliver, N. (2012). TFMAP: Optimizing MAP for top-n context-aware recommendation. In Proceedings of the *35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 155–164).

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., & Hanjalic, A. (2012). CLiMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In Proceedings of the *Sixth ACM conference on Recommender Systems* (pp. 139–146).

Shi, Y., Larson, M., & Hanjalic, A. (2010a). List-wise learning to rank with matrix factorization for collaborative filtering. In Proceedings of the *Fourth ACM conference on Recommender Systems* (pp. 269–272).

Shi, Y., Larson, M., & Hanjalic, A. (2010b). Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In Proceedings of *The Workshop on Context-Aware Movie Recommendation* (pp. 34–40).

Steck, H. (2010). Training and testing of recommender systems on data missing not at random. In Proceedings of the *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 713–722).

Steck, H. (2011). Item popularity and recommendation accuracy. In Proceedings of the *Fifth ACM Conference on Recommender Systems* (pp. 125–132).

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques.

Advances in Artificial Intelligence, 2009, 4.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research, 10(Mar)*, 623–656.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: Optimizing non-smooth rank metrics. In Proceedings of the *2008 International Conference on Web Search and Data Mining* (pp. 77–86).

The University of Chicago's Underpowered Campaign to Achieve Racial Diversity. (2000). *The Journal of Blacks in Higher Education*, (29), 36–37. http://doi.org/10.2307/2678832

Vahidov, R., & Ji, F. (2005). A diversity-based method for infrequent purchase decision support in e-commerce. *Electronic Commerce Research and Applications, 4*(2), 143–158.

Valizadegan, H., Jin, R., Zhang, R., & Mao, J. (2009). Learning to rank by optimizing NDCG measure. In *Advances in Neural Information Processing Systems* (pp. 1883–1891).

Vargas, S., & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In Proceedings of the *Fifth ACM Conference on Recommender Systems* (pp. 109–116).

Vargas, S., & Castells, P. (2014). Improving sales diversity by recommending users to items. In Proceedings of the *8th ACM Conference on Recommender Systems* (pp. 145–152).

Vargas, S., Castells, P., & Vallet, D. (2011). Intent-oriented diversity in recommender systems. In Proceedings of the *34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1211–1212).

Vozalis, M. G., & Margaritis, K. G. (2007). Using SVD and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences, 177*(15), 3017–3037.

Wang, L., MENG, X., & ZHANG, Y. (2013). Applying HOSVD to alleviate the sparsity problem in context-aware recommender systems. *Chinese Journal of Electronics, 22*(4), 773–778.

Wang, X., Rosenblum, D., & Wang, Y. (2012). Context-aware mobile music recommendation for daily activities. In Proceedings of the *20th ACM International Conference on Multimedia* (pp. 99–108).

Weimer, M., Karatzoglou, A., Le, Q. V., & Smola, A. (2007). Maximum margin matrix factorization for collaborative ranking. *Advances in Neural Information Processing Systems, 1–8*.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: Theory and algorithm. In Proceedings of the *25th International Conference on Machine Learning* (pp. 1192–1199).

Yang, X., Guo, Y., Liu, Y., & Steck, H. (2014). A survey of collaborative filtering based social recommender systems. *Computer Communications, 41*, 1–10.

Yoon, V. Y., Hostler, R. E., Guo, Z., & Guimaraes, T. (2013). Assessing the moderating effect of consumer product knowledge and online shopping experience on using recommendation agents for customer loyalty. *Decision Support Systems, 55*(4), 883–893.

Yu, K., Zhu, S., Lafferty, J., & Gong, Y. (2009). Fast nonparametric matrix factorization for

large-scale collaborative filtering. In Proceedings of the *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 211–218).

Yu, S., Yu, K., Tresp, V., & Kriegel, H.-P. (2006). Collaborative ordinal regression. In Proceedings of the *23rd International Conference on Machine Learning* (pp. 1089–1096).

Zhang, F. (2009). Improving recommendation lists through neighbor diversification. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference* (Vol. 3, pp. 222–225).

Zhang, M. (2009). Enhancing diversity in top-n recommendation. In Proceedings of the *third ACM Conference on Recommender Systems* (pp. 397–400).

Zhang, M., & Hurley, N. (2008). Avoiding monotony: improving the diversity of recommendation lists. In Proceedings of the *2008 ACM Conference on Recommender Systems* (pp. 123–130).

Zhang, M., & Hurley, N. (2009a). Novel item recommendation by user profile partitioning. In Proceedings of the *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01* (pp. 508–515).

Zhang, M., & Hurley, N. (2009b). Statistical modeling of diversity in top-n recommender systems. In Proceedings of the *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01* (pp. 490–497).

Zhang, Y. C., Séaghdha, D. Ó., Quercia, D., & Jambor, T. (2012). Auralist: Introducing serendipity into music recommendation. In Proceedings of the *fifth ACM International Conference on Web Search and Data Mining* (pp. 13–22).

Zheng, Y., Mobasher, B., & Burke, R. (2014). CSLIM: Contextual SLIM recommendation algorithms. In Proceedings of the *8th ACM Conference on Recommender Systems* (pp. 301–304).

Zheng, Y., Mobasher, B., & Burke, R. D. (2013). The role of emotions in context-aware recommendation. *Decisions@ RecSys, 2013*, 21–28.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., & Zhang, Y.-C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the *National Academy of Sciences*, *107*(10), 4511–4515.

Zhou, T., Su, R.-Q., Liu, R.-R., Jiang, L.-L., Wang, B.-H., & Zhang, Y.-C. (2009). Accurate and diverse recommendations via eliminating redundant correlations. *New Journal of Physics*, *11*(12), 123008.

8 Appendix A

8.1 Singular Value Decomposition

A rectangular matrix $X \in \mathbb{R}^{m \times n}$ can be decomposed into three matrices in following way:

$$X = USV^T,$$

Where $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, $S \in \mathbb{R}^{m \times n}$. The matrices U, V are orthogonal, so $UU^T = U^TU = I$ and $VV^T = V^TV = I$; here I is an identity matrix, and S is a diagonal matrix with singular values of X. One of the common uses of SVD is to reduce it to rank k by performing truncating top k diagonal values of S.

$$X \approx U_k S_k {V_k}^T,$$

Which means $U_k \in \mathbb{R}^{m \times k}$, $V_k \in \mathbb{R}^{k \times n}$, $S_k \in \mathbb{R}^{k \times k}$. For illustration SVD is demonstrated using following example:

Examples with SVD

Let us start with a 2×3 matrix

$$X = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

To find U calculate XX^T , here X^T represents transpose of X

$$X^T = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

So

$$XX^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

Now calculate Eigenvalues and corresponding Eigenvectors of XX^T. To find Eigenvectors of XX^T find vectors v_1 and v_2.

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

To solve this, rewrite the equation as

$$11v_1 + v_2 = \lambda v_1$$

$$1v_1 + 11v_2 = \lambda v_2$$

Rearranging the above equation

$$(11-\lambda)v_1 + v_2 = 0$$

$$v_1 + (11-\lambda)v_2 = 0$$

Solving λ by setting determinants of the coefficients of the above equation

$$\begin{vmatrix} 11-\lambda & 1 \\ 1 & 11-\lambda \end{vmatrix} = 0$$

Or,

$$(11-\lambda)(11-\lambda)-1=0$$

$$\lambda^2-22\lambda+120=0$$

$$(\lambda-10)(\lambda-12)=0$$

$$\lambda=10\,,\lambda=12$$

These are Eigenvalues of the matrix X; in order to find out Eigenvectors plug in the value of λ in the original equation.

For $\lambda=10$ equation becomes

$$(11-10)v_1+v_2=0 \text{ and}$$

$$v_1+(11-10)v_2=0$$

Or, $$v_2=-v_1$$

The above conditions are satisfied for many integers, but take the smallest magnitude of a vector which is $[1 \quad -1]$ corresponding to $\lambda=10$.

Similarly, for $\lambda=12$

$$(11-12)v_1+v_2=0 \text{ and}$$

$$v_1+(11-12)v_2=0$$

Or, $$v_2=v_1$$

So, the Eigenvectors for the above condition gives $[1 \quad 1]$ corresponding to $\lambda=12$

Now arrange the Eigenvectors in a column of matrix with Eigenvectors corresponding to largest Eigenvalue coming in the first column, while Eigenvectors corresponding to smaller Eigenvalue are arranged in subsequent columns. Here, Eigenvectors corresponding to Eigenvalue $(\lambda) = 12$ will appear in first column and Eigenvectors corresponding to Eigenvalue $(\lambda) = 10$ will appear in second column.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Finally create an orthogonal matrix by applying Gram-Schmidt ortho- normalization process to the column vectors.

$$\vec{u}_1 = \frac{\vec{v}_1}{|\vec{v}_1|} = \frac{[1,1]}{\sqrt{1^2 + 1^2}} = \frac{[1,1]}{\sqrt{2}} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$$

$$\vec{w}_2 = \vec{v}_2 - \text{projection}_{\vec{u}_1}(\vec{v}_2) = \vec{v}_2 - \vec{u}_1 \cdot \vec{v}_2 * \vec{u}_1$$

$$= [1, -1] - \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right] \cdot [1, -1] * \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$$

$$= [1, -1] - 0 * \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$$

$$= [1, -1]$$

By normalizing $\vec{w}_2$ obtain $\vec{u}_2$

$$\vec{u}_2 = \frac{\vec{w}_2}{|\vec{w}_2|} = \frac{[1, -1]}{\sqrt{1^2 + 1^2}} = \frac{[1, -1]}{\sqrt{2}} = \left[\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right]$$

This gives

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

The calculation of V is similar to U, however find Eigenvectors of X^TX

$$X^TX = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

Now find Eigenvalues of X^TX by

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$10v_1 + 0v_2 + 2v_3 = \lambda v_1$$

$$0v_1 + 10v_2 + 4v_3 = \lambda v_2$$

$$2v_1 + 4v_2 + 2v_3 = \lambda v_3$$

Rearranging the above,

$$(10 - \lambda)v_1 + 0v_2 + 2v_3 = 0$$

$$0v_1 + (10 - \lambda)v_2 + 4v_3 = 0$$

$$2v_1 + 4v_2 + (2 - \lambda)v_3 = 0$$

Using determinant

$$\begin{vmatrix} (10 - \lambda) & 0 & 2 \\ 0 & (10 - \lambda) & 4 \\ 2 & 4 & (2 - \lambda) \end{vmatrix} = 0$$

Or, $$(10-\lambda)\begin{vmatrix}(10-\lambda) & 4\\ 4 & (2-\lambda)\end{vmatrix}+2\begin{vmatrix}0 & (10-\lambda)\\ 2 & 4\end{vmatrix}=0$$

Or, $$(10-\lambda)[(10-\lambda)(2-\lambda)-16]+2[0-(20-2\lambda)]=0$$

Or, $$\lambda(\lambda-10)(\lambda-12)=0$$

So, Eigenvalues for X^TX are $\lambda=0,\lambda=10$ and $\lambda=12$. Substituting λ back to the original equation find Eigenvectors.

For $\lambda=12$ equations become

$$(10-12)v_1+0v_2+2v_3=0$$

$$0v_1+(10-12)v_2+4v_3=0$$

$$2v_1+4v_2+(2-12)v_3=0$$

Solving this get $v_1=1, v_2=2$ and $v_3=1$

For $\lambda=10$ equations become

$$(10-10)v_1+0v_2+2v_3=0$$

$$0v_1+(10-10)v_2+4v_3=0$$

$$2v_1+4v_2+(2-10)v_3=0$$

Solving this get $v_1=2, v_2=-1$ and $v_3=0$

For $\lambda=0$ equations become

$$(10-0)v_1+0v_2+2v_3=0$$

$$0v_1+(10-0)v_2+4v_3=0$$

$$2v_1 + 4v_2 + (2-0)v_3 = 0$$

Solving this get $v_1 = 1, v_2 = 2$ and $v_3 = -5$

Now arranging Eigenvectors in the form of matrix according to the size of Eigenvalue

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 0 & -5 \end{bmatrix}$$

Doing Gram-Schmidt ortho-normalization of column vectors

$$\overrightarrow{V_1} = \frac{\overrightarrow{v_1}}{|\overrightarrow{v_1}|} = \frac{[1,2,1]}{\sqrt{1^2+2^2+1^2}} = \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right]$$

$$\vec{a}_2 = \vec{v}_2 - \text{projection}_{\vec{V}_1}(\vec{v}_2) = \vec{v}_2 - \vec{V}_1 \cdot \vec{v}_2 * \vec{V}_1$$

$$= [2,-1,0] - \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right] \cdot [2,-1,0] * \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right]$$

$$= [2,-1,0] - 0 * \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right]$$

$$= [2,-1,0]$$

$$\overrightarrow{V_2} = \frac{\overrightarrow{a_2}}{|\overrightarrow{a_2}|} = \frac{[2,-1,0]}{\sqrt{2^2+(-1)^2+0^2}} = \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, \frac{0}{\sqrt{5}}\right]$$

$$\vec{a}_3 = \vec{v}_3 - \text{projection}_{\vec{V}_1}(\vec{v}_3) - \text{projection}_{\vec{V}_2}(\vec{v}_3) = \vec{v}_3 - \vec{V}_1 \cdot \vec{v}_3 * \vec{V}_1 - \vec{V}_2 \cdot \vec{v}_3 * \vec{V}_2$$

$$= [1,2,-5] - \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right] \cdot [1,2,-5] * \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right] - \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, \frac{0}{\sqrt{5}}\right] \cdot [1,2,-5] * \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, \frac{0}{\sqrt{5}}\right]$$

$$= [1,2,-5] - 0 * \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right] - 0 * \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, \frac{0}{\sqrt{5}}\right] = [1,2,-5]$$

Normalizing $[1,2,-5]$

$$\overrightarrow{V_3} = \frac{\overrightarrow{a_3}}{|\overrightarrow{a_3}|} = \frac{[1,2,-5]}{\sqrt{1^2 + 2^2 + (-5)^2}} = \left[\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-5}{\sqrt{30}}\right]$$

Therefore,

$$V = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

Transposing V obtain V^T

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

For calculating S of matrix X find the square root of non-zero Eigenvalues of XX^T or X^TX. Note that non-zero Eigenvalues will be equal for both resulting matrices. These are called singular values of matrix X. The diagonal values in S are singular values of matrix X, columns in U are called left singular vectors and the columns in V are called right singular vectors.

$$S = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

The singular decomposition of matrix X is given by.

$$X = USV^T,$$

$$X = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

8.2 Novelty calculation

The below toy example illustrates the calculation of novelty values of items at different instants of time. It is important to note that e-commerce platforms report the UNIX timestamp of a transaction log. Therefore, for all practical purposes, the timestamp corresponding to each rating is used in model building.

User ID	Item ID	Rating	Timestamp
U1	I1	5	874965758
U1	I3	3	876893171
U2	I2	5	878542960
U2	I4	2	876893119
U3	I1	3	889751712
U4	I2	3	875071561
U5	I3	5	875072484

Table 17: A toy example for user-item rating and timestamp representation

User/items	I1	I2	I3	I4
U1	5		3	
U2		5		2
U3	3			
U4		3		

U5			5	

Table 18: Representation of user-item rating matrix

In the above toy example, assuming that there are only 5 users in the database, item I1 has been rated by user U1 and U3 at t=874965758 and 889751712 respectively. To calculate featured scaled novelty of I1 the following steps may be applied:

1. Calculate popularity at t=874965758 and 889751712

$$p(\mathrm{I1}_{874965758}) = \frac{\#\ users\ prefering\ item\ \mathrm{I1}\ till\ time\ t1}{\#total\ users} = \frac{1}{5}\ \&$$

$p(\mathrm{I1}_{889751712.}) = \frac{2}{5}$

2. Min { $(\frac{1}{p(I1_t)})$, $(\frac{1}{p(I2_t)})$ } ={5, 2.5} =2.5

3. Divide $(\frac{1}{p(I1_t)})$ by min { $(\frac{1}{p(I1_t)})$, $(\frac{1}{p(I2_t)})$ } = $\frac{5}{2.5}$=2 and $(\frac{1}{p(I2_t)})$ by min { $(\frac{1}{p(I1_t)})$, $(\frac{1}{p(I2_t)})$ }= $\frac{2.5}{2.5}$=1

4. Base is corresponding maximum value of $(\frac{1}{p(i_t)})$ =5

5. $novelty(I1_{874965758}) = log_{base}\left(\frac{1}{p(\mathrm{I1}_{874965758})}\right) = log_5 5 = 1$

$$and\ novelty(\mathrm{I2}_{889751712.}) = log_{base}(\frac{1}{p(\mathrm{I2}_{889751712.})}) = log_5 2.5 = 0.57$$

Note that the featured scaled novelty values obtained in such a manner are normalized between 0 and 1. Similar steps can be followed for other items in database. A novelty matrix

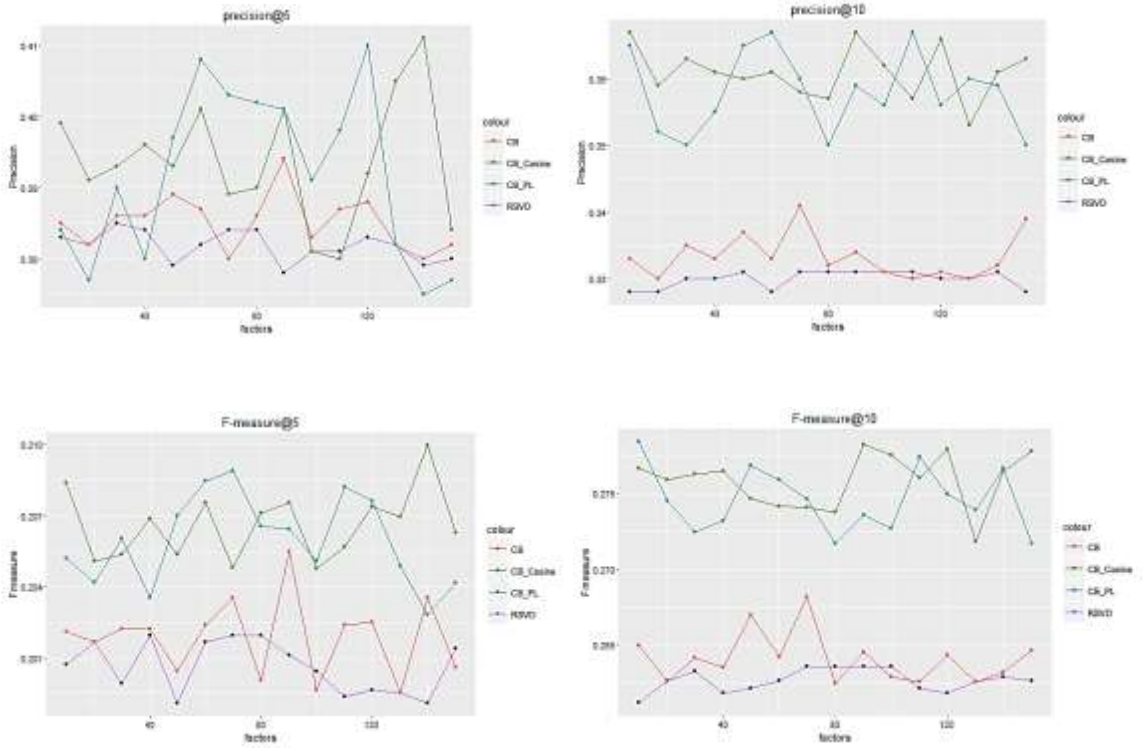
can thus be obtained after calculating novelty values in the above manner for all items at each instant of time.

	I1	I2	I3	I4
U1	1 (874965758)		0.57 (876893171)	
U2		0.57 (878542960)		1 (876893119)
U3	0.57 (889751712)			
U4		1 (875071561)		
U5			1 (875072484)	

Table 19: Resultant Novelty matrix

Having obtained the novelty matrix, the next step is to integrate it with matrix factorization.

9 Appendix B



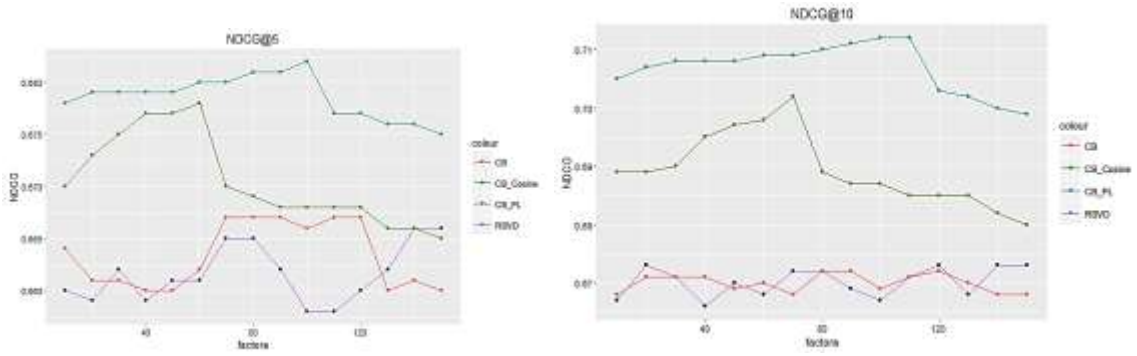


Figure 19: Impact of number of latent factors (D) on various performances metric for ml-100k (N=20)

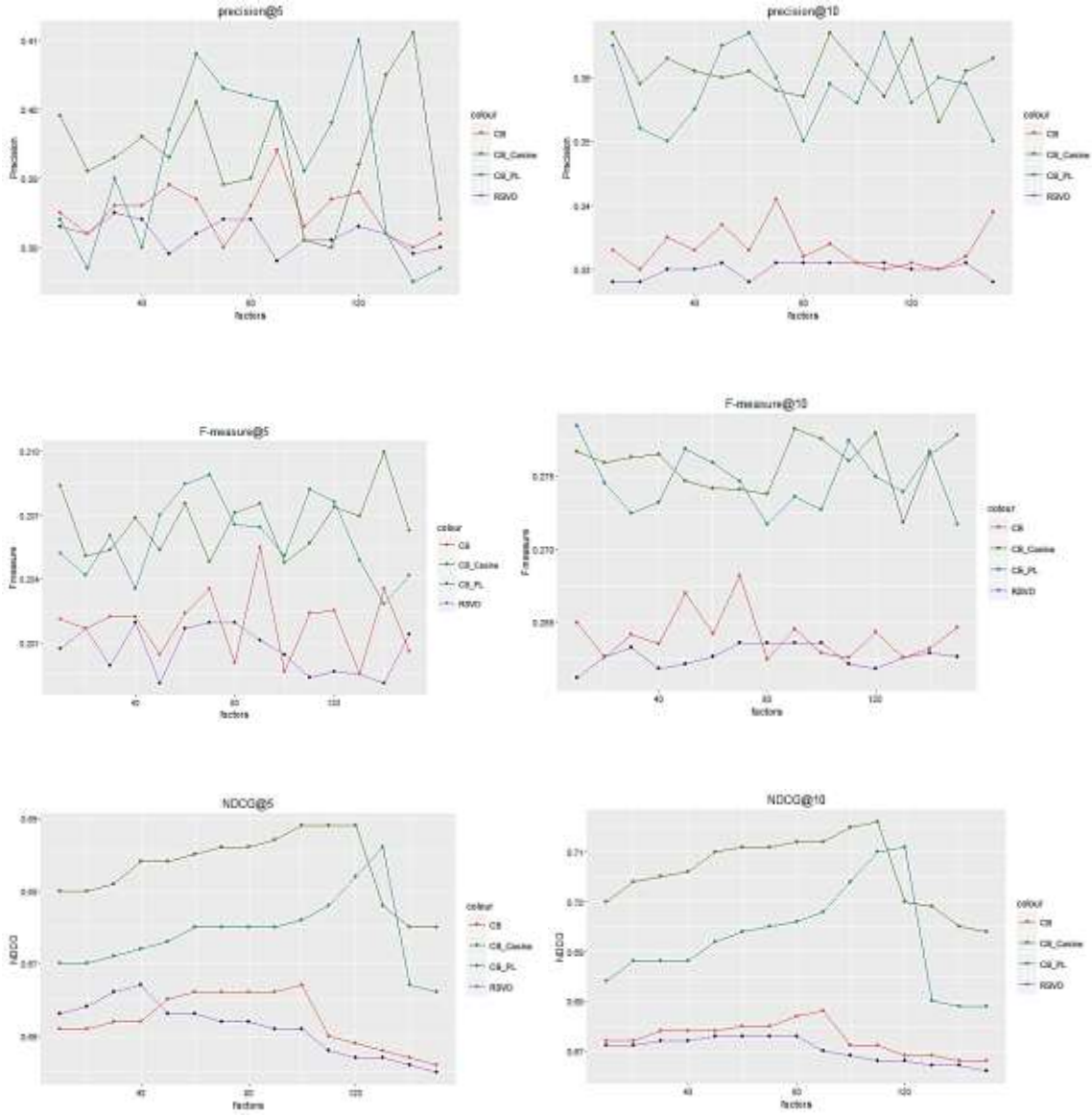


Figure 20: Impact of number of latent factors (D) on various performances metric for ml-100k (N=50)

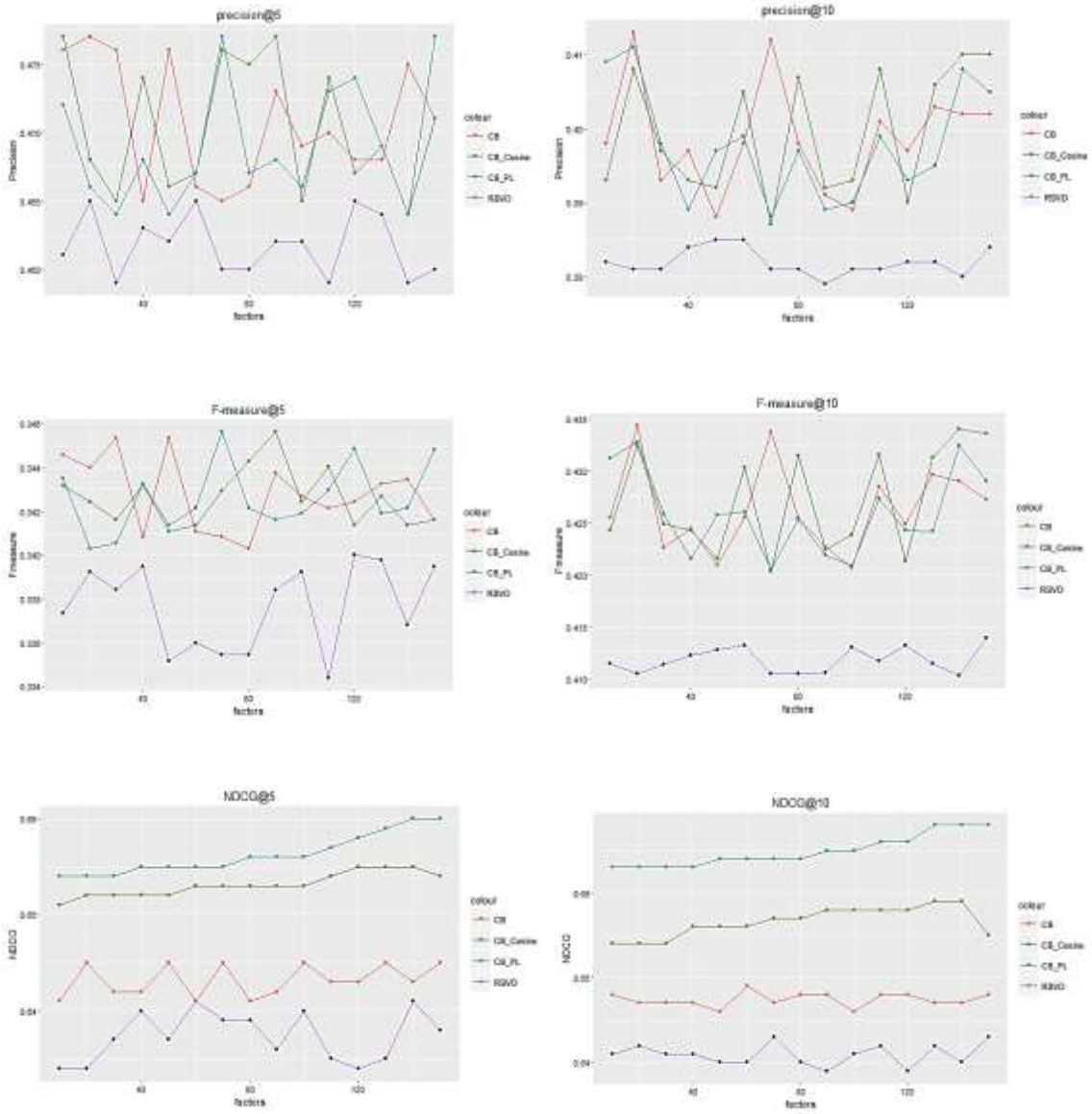
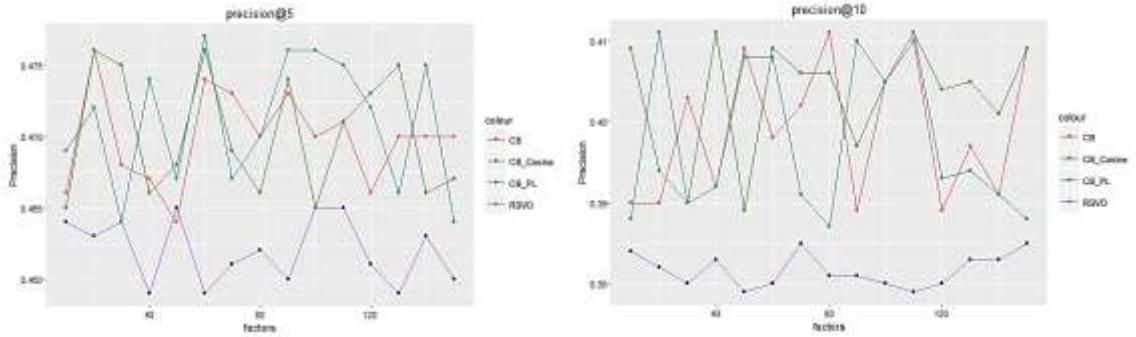


Figure 21: Impact of number of latent factors (D) on various performances metric for ml-1m (N=10)



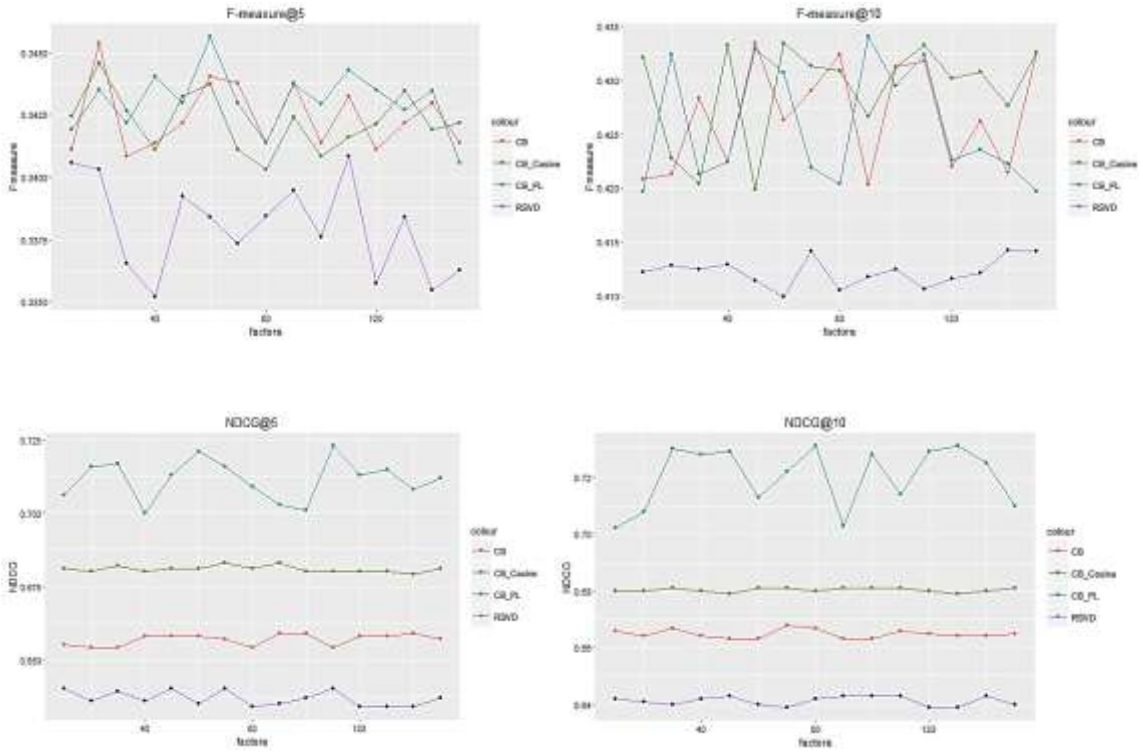
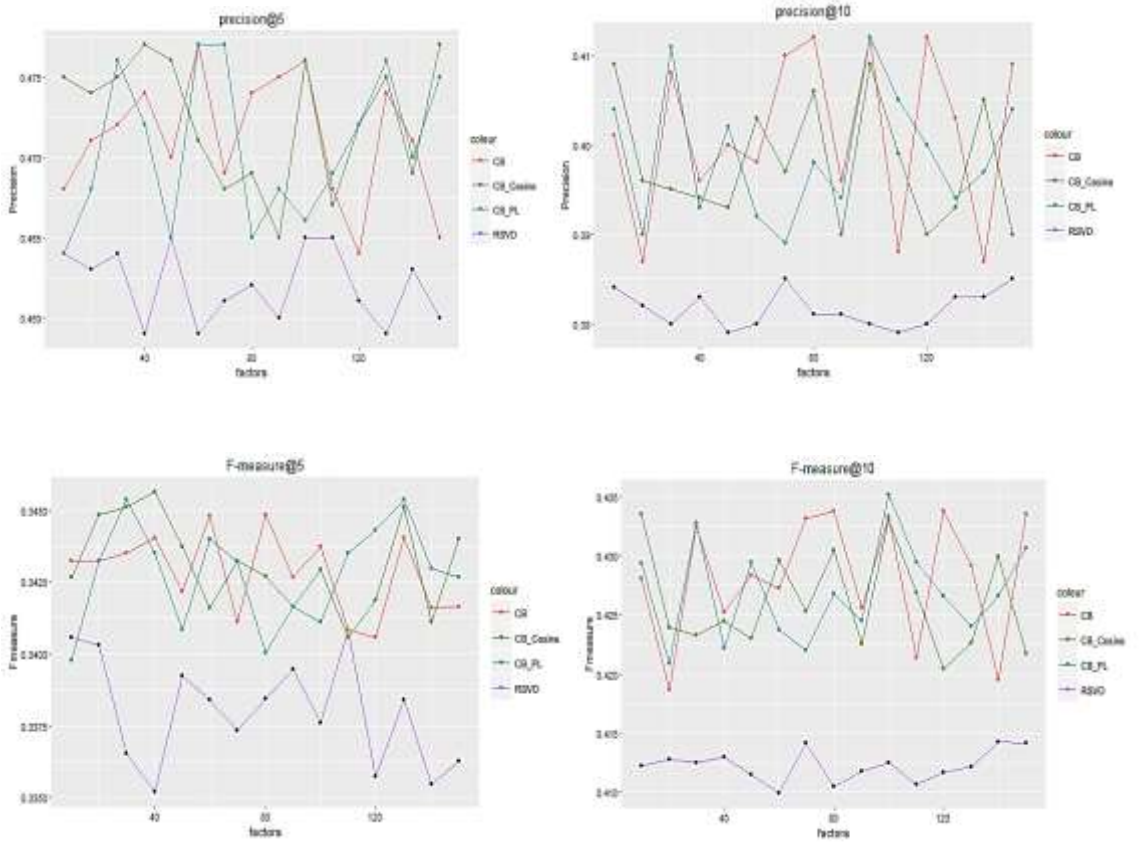


Figure 22: Impact of number of latent factors (D) on various performances metric for ml-1m (N=20)



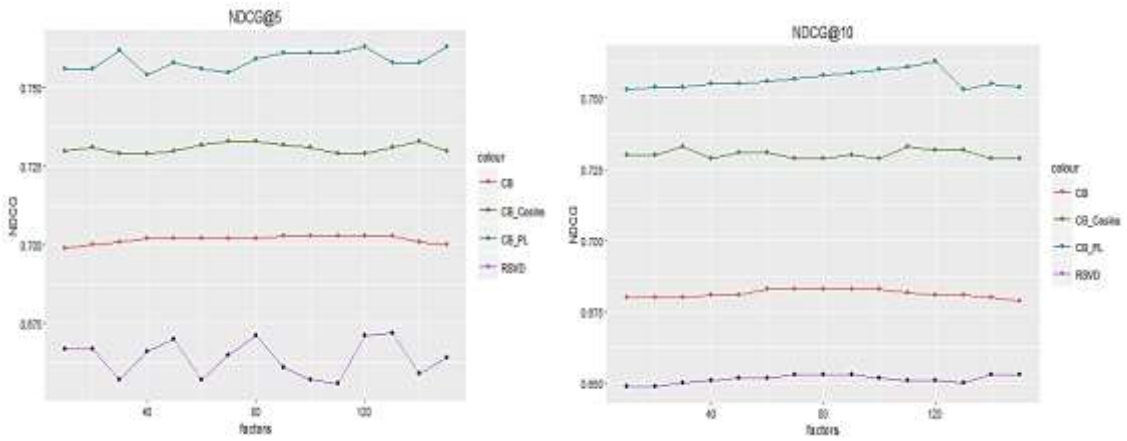


Figure 23: Impact of number of latent factors (D) on various performances metric for ml-1m (N=50)