**PERSONALISED RANKING WITH SINGLE SOURCE
IMPLICIT INFORMATION FOR RECOMMENDATION
TASKS, A SIMILARITY BASED MONTE CARLO
BAYESIAN PERSONALISED RANKING**

by

Parisa Lak

B.Eng, Tehran Markaz Azad University, Tehran, Iran 2004

MBA, Sharif University of Technology, Tehran, Iran, 2009

MMSc, Ryerson University, Toronto, Canada, 2013

A dissertation presented to Ryerson University in partial fulfilment
of the requirements for the degree of Doctor of Philosophy in the
program of Mechanical and Industrial Engineering

Toronto, Ontario, Canada, 2018

©Parisa Lak, 2018

Declaration of Authorship

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

PERSONALISED RANKING WITH SINGLE SOURCE IMPLICIT INFORMATION
FOR RECOMMENDATION TASKS, A SIMILARITY BASED MONTE CARLO
BAYESIAN PERSONALISED RANKING

Parisa Lak

PhD, Mechanical and Industrial Engineering

Ryerson University, 2018

Abstract

Background: A recommender algorithm's main goal is to learn user preferences from the user-system interactions and provide a list of relevant items to the user. In information retrieval literature this problem is formulated as learning to rank (LtR) problem. Bayesian Personalized Ranking (BPR) [1] is one of the popular LtR approaches based on pair-wise comparison using single source implicit information.

Aim: In this work, we aim to design a recommender system algorithm that generates accurate recommendations. The system should only use a single source implicit user preference information. This is possible through a good approximation of the posterior probability in BPR optimization function.

Method: We proposed a Similarity based Monte Carlo approximate solution for the posterior probability in BPR. We used four datasets from different recommendation application domains to evaluate the performance of our proposed algorithm. The input data was pre-processed to match with the requirements of the algorithm.

Result: The result of the analysis shows a significant improvement in terms of mean average precision (MAP) for our proposed algorithm compared with the BPR and another alternative extension to BPR.

Conclusion: We conclude that the proposed approximate solution is successful in providing the most informative samples to approximate BPR posterior probability. This is confirmed by the significant improvement of the accuracy of the provided ranked list of items for the users.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Ayse Bener for the continuous support during my PhD study and related research, for her motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this dissertation.

Besides my advisor, I would like to thank the rest of my dissertation committee: Dr. Arthur Ryman, Dr. Cory Searcy, and Dr. Andriy Miranskyy, for their insightful comments and encouragement, and also for the hard question which incited me to widen my research from various perspectives.

My sincere thanks also goes to all industry partners who provided me with the opportunity to work on real-life projects and gain hands-on experiences. Special thanks to IBM Canada Lab for their support and dedication and providing the opportunity for me to work closely with the IBM Watson Analytics team. Many thanks goes to Martin Petitsclerc for his support and insightful feedback throughout our collaborative project.

Also, I would like to thank all the post doctoral fellows who helped me during the past four years, specially Dr. Shariyar Murtaza, Dr. Carl Barrelet, and Can Kavaklioglu. Many thanks to my lab mates for their continuous support and encouragement throughout these years.

Last but not the least, I would like to thank my family, my parents and my brother and sister for supporting me spiritually throughout completing and writing this dissertation and my life in general.

*To my parents, Dr.Parviz Lak and Mrs.Jaleh Zahed, for
their ongoing support and to my lovely brother and
sister, Pedram and Parinaz, for their words of
encouragement*

Contents

Declaration of Authorship	ii
Abstract	iii
Acknowledgements	iv
List of Figures	viii
List of Tables	x
Abbreviations	xi
1 Introduction	1
2 Problem Statement	5
3 Background	9
3.1 Recommender Systems	9
3.2 Major Challenges in Recommender Systems	14
3.3 Implicit User Preferences	17
3.4 Learning to Rank for Recommender Systems	20
3.5 Bayesian Personalized Ranking- Review	21
4 Methodology	28
4.1 Bayesian Inference	32
4.1.1 Bayesian Probability	33
4.1.2 Bayesian Machine Learning	34
4.1.3 Approximate Inference	35
4.1.4 Markov Chain Monte Carlo Sampling	36
4.2 Bayesian Personalized Ranking	39
4.3 Scoring Function with Matrix Factorization	46
4.4 Problem Formulation	50

4.5	Proposed Similarity based Monte Carlo Bayesian Personalized Ranking (SMC-BPR) approach	52
4.6	Alternative Strategies	61
4.7	Recommender System Evaluation	65
4.8	Datasets	68
5	Experiments and Results	83
5.1	Aim of the study	83
5.2	Selection of the response variable	84
5.3	Choice of factors and levels	86
5.4	Choice of experiment design	86
5.5	Performing the experiment	90
5.6	Analysis of the data and discussion of the results	90
5.7	Threats to Validity	99
6	Conclusion and Future Directions	102
6.1	Contributions	104
6.1.1	Theoretical and Methodological Contributions	104
6.1.2	Practical Implications	105
6.2	Future Direction	107
	Bibliography	108

List of Figures

4.1	Schematic representation of data from implicit information and the transformation solutions. (a) consider all missing as not preferred (b) pairwise comparison approach	30
4.2	Score Prediction based on Matrix Factorization	47
4.3	Illustration of Sigmoid function. This function goes to its minimum, being zero, as x decreases to negative values.	54
4.4	Illustrative example of IBM Watson Analytics starting point recommendations	71
4.5	Overview of datasets extracted from IBM Watson Analytics Logstash	73
4.6	a	75
4.7	b	75
4.8	Item popularity (sparsity) for WA dataset. Figure (a) shows the probability of selection for the items in the WA dataset. Figure (b) shows the log of the selection frequency for each item in the dataset	75
4.9	a	77
4.10	b	77
4.11	Item popularity (sparsity) for ML-100K dataset. Figure (a) shows the probability of rating each item in the ML-100K dataset. Figure (b) shows the log of the selection frequency for each item in the dataset	77
4.12	a	79
4.13	b	79
4.14	Item popularity (sparsity) for Frappe dataset. Figure (a) shows the probability of selection for the items in the Frappe dataset. Figure (b) shows the log of the selection frequency for each item in the dataset	79
4.15	a	81
4.16	b	81
4.17	Item popularity (sparsity) for Jester dataset. Figure (a) shows the probability of selection for the items in the Jester dataset. Figure (b) shows the log of the selection frequency for each item in the dataset	81
5.1	An example of Mean Average Precision@10 computation for two users	85

5.2	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for WA dataset in terms of MAP on each training epoch for test instances	91
5.3	The comparison of the three algorithms, Blue: BPR, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for ML-100K dataset in terms of MAP on each training epoch for test instances	92
5.4	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Frappe dataset in terms of MAP on each training epoch for test instances	93
5.5	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Jester dataset in terms of MAP on each training epoch for test instances	94
5.6	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for WA dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars. . .	95
5.7	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for ML dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars. . .	96
5.8	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Frappe dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars. . . .	97
5.9	The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Jester dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars. . . .	98

List of Tables

3.1	Typical RS data representation example	15
3.2	Summary of BPR Extensions	26
3.3	Summary of BPR Extensions	27
4.1	Description of the datasets used in this study. IBM WA is the unique dataset from obfuscated user interaction with IBM Watson Analytics data visualisation recommendation application. ML-100K is the movie rating dataset made available by Grouplens. Frappe dataset is user mobile application interaction dataset and jester is a joke rating dataset	70
4.2	Sparsity Rate for each dataset	70
5.1	Description of the pre-processed datasets used in this study.	87
5.2	Summary of minimum user activity criterion for each dataset.	87
5.3	Summary of Regularisation Rate and Learning Rate for each dataset.	88
5.4	Summary of the number of features for each dataset.	88
5.5	Summary of the number of samples per training epoch for each dataset.	89
5.6	Summary of the number MCMC iterations for each dataset.	89
5.7	The p -value and t -value for the t -test analysis to show the significant difference in the performance of SMC-BPR-MF compared with the baselines BPR-MF and AS-BPR-MF in terms of MAP. This table shows the result for the reported results in this study.	95

Abbreviations

RS	R ecommender S ystems
IR	I nformation R etrieval
BPR	B ayesian P ersonalized R anking
MF	M atrix F actorization
MAP	M ean A verage P recision
AP	A verage P recision
ROC	R eciever O perating C haracteristic Curve
AUC	A rea U nder the R OC C urve
CF	C ollaborative F iltering
CB	C ontent B ased
SVD	S ingular V alue D ecomposition
LtR	L earning to R ank
KNN	K N earest N eighbor
NLL	N egative L og L ikelihood
WA	W atson A alytics

Chapter 1

Introduction

The World Wide Web interconnect so many online services, which yields to the interconnection of massive amount of information. Users searching for the next movie to watch, the best camera to buy, the next vacation to book, or the best restaurants to go to are offered with thousands of options. The huge pool of options makes it overwhelming for the users to pick the one right option. Recommender systems (RS) have been proposed and now are largely diffused to help users find the most appropriate items from the large number of available options (e.g. movie, camera, vacation, restaurant). Recommender Systems are intelligent applications that are able to identify and suggest the products, information or services (i.e. items) that best fit the user needs and preferences. In this work, we aim to design a successful Recommender System algorithm that can perform with the minimum amount of user feedback information.

One of the most important factors that determines the success of a RS technology is its accuracy in prediction of the right set of items to suggest to the user [2]. Various algorithms and models are proposed and tested in the literature to satisfy this requirement [1, 3–11]. Each algorithm is designed based on the available information from users and items, as well as the user preference information. The user preference information is either explicitly provided by the users themselves (e.g. in the form of ratings) or it should be implicitly derived from their interaction(s)

with the applications, such as a purchase action in an e-commerce application. The latter is referred to as implicit user preference information.

The implicit user preference information is the most available and most reliable source of user feedback information [12]. The Recommender System algorithms using the implicit source of information sometimes follow the methodologies from Learning to Rank (LtR) literature [1, 8, 13, 14]. Learning to Rank for top-n item recommendations, either follow the point-wise prediction approach, pair-wise comparison approach or list-wise comparison approach. All of these Learning to Rank approaches has been previously used to design recommender systems [1, 4, 14, 15]. The use of pairwise ranking approach have shown to work well when the input data is implicit information [16–21]. Bayesian Personalized Ranking (BPR) approach is one of the most popular pairwise comparison approaches suggested to design a Recommender System [17–23].

BPR provides a Bayesian optimization framework for pairwise comparison Learning to Rank problem to deal with implicit information [1]. However, BPR has some limitations that comes from the approximate solution suggested for the posterior probability. In a pairwise comparison framework, all items should be compared against each other, which causes a large number of computations, specially when the item space is large. This problem frequently occurs in Bayesian Machine Learning approaches that is the basis of BPR approach. The solution to this problem is the use of either deterministic approximations, such as Variational Bayes and Expectation Propagation, or the use of approximate inference methods based on numerical sampling [24]. BPR suggests a bootstrap sampling as an approximate solution to reduce the number of computations. However, as the number of items increases the search space becomes larger and finding the right sample that contributes most to the learning process becomes hard. We propose an approach to mitigate this problem and detect the most informative samples for the learning algorithm.

Specifically, to design our recommender system algorithm, we propose an extension to the Bayesian Personalized Ranking (BPR) approach that performs superior to

the original approach in terms of the accuracy of the provided ranked list of items. In fact, we propose an approximate inference method for the posterior probability in the BPR approach. A similarity based Monte Carlo Markov Chain method is adopted to approximate the posterior probability in BPR. The proposed approach is referred to as Similarity based Monte Carlo Bayesian Personalized Ranking (SMC-BPR).

We evaluate the performance of our proposed approach with a factorization model as the underlying scoring function. The proposed recommender system algorithm, Similarity based Monte Carlo BPR Matrix Factorization (SMC-BPR-MF), performs significantly better than the baseline algorithm, being the factorization based Bayesian Personalized Ranking (BPR-MF). The stability of the proposed algorithm based on hyper parameters depends on the input data sparsity as was also observed for other variations of the BPR algorithm.

We also compared the performance of our proposed algorithm with an alternative extension to the algorithm, being the Bayesian Personalized Ranking with adaptive sampling (we refer to this algorithm as AS-BPR-MF) presented in [18]. Similar result is observed for the comparison of our proposed algorithm with the alternative algorithm (i.e. AS-BPR-MF). In that, our algorithm significantly outperforms the alternative algorithm as well. The result is consistent for all four datasets used in this study in terms of Mean Average Precision (MAP) measure. Mean Average Precision is a measure that evaluates the quality of a ranked list of items based on historical data. The across study evaluation of all experiments with different experiment settings (variations in hyper parameters) also provides evidence that our proposed algorithm performs significantly better than the two baseline algorithms, in terms of the accuracy of the list of recommended items.

This thesis is organised as follows. In Chapter 2, we specify the problem in hand and we outline the research question that we intend to address in this work. Chapter 3 provides a review of literature on recommender systems and its dependencies. This is along with the review of the literature on the main algorithm used in this research that is Factorization based Bayesian Personalized Ranking (BPR-MF).

We then outline our methodology in Chapter 4. In this chapter we first provide an overview of Bayesian Inference and Bayesian Machine Learning. We then provide a detailed explanation of the BPR method followed by the proposed methodology that extends the BPR-MF algorithm. In Chapter 5, we provide an exploratory analysis of the datasets used in this study as well as a detailed experiment design. We then provide the analysis of the data as well as the discussion of the result in this chapter. Chapter 6 provides a brief summary and the conclusion of our results along with the future directions of our work.

Chapter 2

Problem Statement

Recommender systems (RS) are one of the tools that provide individuals with the most relevant suggestions, using information filtering strategies [25]. Some examples for RS are movie recommendations provided by Netflix, product recommendations by Amazon, travel destination recommendations by Trip advisor and Restaurant recommendations by Yelp. All applications provide a list of potential items (e.g. movies, products) that their user may wish to utilise [2]. The motivation behind this study comes from the increasing need in the design and implementation of such systems in various applications. Recommender systems learn user preferences from their historical behaviour/preferences and then provide recommendations based on the prediction of users future preferences [2].

In most applications users historical preferences are not provided explicitly in the form of likes and dislikes or ratings [26]. For example, in an e-commerce application, a user might not be willing to provide rating for a product. However, the system should be able to detect his/her preferences based on his/her purchase behaviour, without requesting for an extra information. In these situations, user preference information should be implicitly derived from users interaction with the application [12]. There is a growing trend towards the collection and use of implicit information, instead of relying on traditional explicit evaluation of items through ratings, in the recommender system domain [16].

Implicit information is extracted from user interaction log in a target application (e.g. movie recommendation website or e-commerce application). Therefore, this type of user preference information is one of the most readily available source of information to be used as the input for a recommender system. This information can be in the form of click on an item, purchase of an item or the time spent on one item. At least one of the aforementioned information is available in most user interaction logs [12]. We refer to the availability of one of these sources of information as the availability of a *single source implicit information*.

One drawback of using a single source implicit type of information in recommender systems is that it only provides one class information [12]. This means that we are only able to detect the positive feedback from the users (e.g. clicks, purchase), and not the negative ones (i.e. the items that are not preferred by the user) [12]. Most studies in the literature consider the lack of positive feedback as negative feedback [8, 27–29]. However, this assumption may not always be correct [1]. For example, lack of click on an item may be because the user has never seen that item. One approach that deals with this drawback is the pairwise comparison approach, adopted from Learning to Rank literature for the design of the recommender systems [30].

In the pairwise comparison framework, user preferences are modelled as the relative preference of one item over the other. Recommender system algorithms are proposed in the literature that follow the pairwise comparison framework based on implicit information. Some examples are Bayesian Personalized Ranking (BPR) [1] and Supervised Semantic Indexing (SSI) [19]. The use of Bayesian Personalized Ranking (BPR) approach [1] is omnipresent in the pairwise comparison based recommender system design [17–23]. BPR is a Bayesian optimization approach that deals with single source implicit information, using a pairwise comparison framework.

BPR also provides an approximate inference solution for the pairwise comparison framework for recommender systems. In a pairwise comparison framework, all items should be compared against each other, which causes a large number of

computational operations. This problem exacerbates when the item space is large. This is a common incident in Bayesian Machine Learning approaches, where the posterior probability becomes hard to compute. The solution to this problem is the use of either deterministic approximations, such as Variational Bayes and Expectation Propagation, or the use of approximate inference methods based on numerical sampling [24]. BPR suggests a bootstrap sampling as an approximate solution to reduce the number of computations. However, as the number of items increases the search space becomes larger and finding the right sample that contributes most to the learning process becomes harder. This is a limitation with the BPR approach that was also clearly raised in [18] and [21] previously.

To tackle this limitation in BPR, some works propose the use of additional information to refine the parameters of BPR [17, 22, 31, 32]. For instance, some use this additional information to group items [22, 31] before the random sampling solution. Some others use the extra information to refine the learning processes [17, 32], without tackling the approximate inference limitation directly. However, additional information might not always be available or it might be hard to extract for different recommendation tasks. Some extensions to BPR, propose non-uniform sampling strategies to perform better approximations [18, 21]. However, these extensions generate additional limitations, such as the introduction of additional hyper parameters and additional computation steps. We consider some of these approaches as the baseline in our experiments.

In our work, we propose a similarity based Monte Carlo Markov Chain method to better approximate the posterior probability in BPR. Our proposed method, the Similarity based Monte Carlo - BPR (SMC-BPR), in essence searches for a decision boundary [33], separating the positive and negative items for a given user. This is similar to finding the correct hypothesis class with a certain level of confidence in Probability Approximately Correct (PAC) learning [33]. As the number of items increase detecting this boundary becomes a harder problem to solve exactly, which forces researchers to come up with efficient approximate methods. SMC-BPR presents an alternative approximate solution to find the most informative

negative items to better estimate the decision boundary between the positive and negative items in BPR framework.

To summarise, the general question that we address in this work, from the practical point of view is:

How to generate accurate personalised ranked list of items, using a single source implicit information?

To address this practical question, we proposed an extension to Bayesian Personalized Ranking approach to generate more accurate recommendations. The BPR approach dealt with a single source implicit information, but had some limitations based on Bayesian inference frameworks. We can summarise our research question from theoretical point of view as:

RQ. *How to improve the accuracy of Bayesian Personalised Ranking approach for Recommender Systems using a single source implicit information?*

Our proposed solution to address this question, mainly focuses on providing a better approximate inference solution for the posterior probability in BPR to generate more accurate recommendations.

Chapter 3

Background

In this chapter we explain the concept of recommender system (RS) and its dependencies in Section 3.1. The evaluation metrics for RS is reviewed in Section 4.7. We also outline the challenges in RS domain and review literature on various available solutions in Section 3.2. We then explain the use of implicit source of information to predict user preferences with recommender systems 3.3. In Section 3.4 we review the Learning to Rank approaches for Recommender systems. Section 3.5 provides an overview on Bayesian Personalized Ranking (BPR), which is the main recommender system approach in this research and its extensions.

3.1 Recommender Systems

In the era of big data and information overload, “Recommender Systems”(RS) are designed to simplify the selection tasks for products and services. Some examples for the selection tasks may be what product to buy, which music to listen to, which movie to watch or what type of travel plan to choose. The diversity of RS applications and the variety of data types and recommendation techniques arises a significant body of work both in academia and in industry on the advancements of these systems during the last two decades [2, 7]. In RS there are three key knowledge source: **Users**, **Items** and **Transactions**[2].

Users of RS may have different goals, while using these systems. For instance, in a movie recommendation application a user may look for the most popular movie regardless of its genre, while the other user may consider watching a horror movie from 1985 that was only watched by a number of professional movie critics. In order to provide personalized recommendations, RS exploits a range of information about the user. The type and format of this information is dependent on the recommendation technique and is either explicitly provided by the user or extracted based on the their interaction with the RS application [2].

Items in RS applications are the objects that are recommended. Items are different in terms of complexity and value. Some items are complex in terms of their descriptions such as financial investments, jobs and insurance policies, while others are containing low complexity and value such as books, news, web pages and movies [34]. To design an appropriate RS system the nature of item to recommend should be one of the key elements while deciding about the recommendation technique.

Recorded interaction between user and the RS application are generally referred to as **Transactions**. The transaction information can be extracted from the log-like data that stores the important information generated during the user interaction with the RS application. These interactions provide historical user preferences. The historical preferences are either provided by the user as an explicit information (e.g. movie ratings) or implicit information (e.g. clicks) (see Section 3.3). Ratings are the most common form of transactions recorded from user interactions with RS systems [2]. Ratings can be provided in variety of forms such as numerical ratings (e.g. in scale of 1-5), ordinal ratings (e.g. strongly agree, agree, neutral, disagree, strongly disagree), Binary ratings and Unary ratings (e.g. clicks) [35]. In the latter case, normally, the absence of such ratings is considered as having no information from the user [3].

In a recommender system application, the design, graphical user interface as well as the core recommendation techniques are used alongside each other to provide

customised and useful item suggestions to each user interacting with the application. The user may or may not accept the recommendation provided by the RS and also may or may not provide a feedback for the future recommendations[36]. In any case, all user interactions with the application can be stored in a database to be used for generating new recommendations in the next user-system interactions. RS according to their core technology (i.e. recommendation technique) may use a range of properties recorded in that database. Some RS techniques may be more knowledge dependent than the others. For instance in a movie recommendation domain, a recommendation technique might need contextual information on when and where the movie was evaluated. On contrary, some techniques might be knowledge poor and might only need the user interaction information without any further details [3]. The latter techniques might generate poorer performances, however they do not require any user effort to provide recommendations. In this research, we focus on the advancement of the core technology using a knowledge poor approach. The implementation and the design of the user interface and recommendation presentation may be addressed as a future works.

Recommendation techniques can be divided into three main categories: (1) collaborative filtering based technique, (2) content based technique, and (3) hybrid technique [5].

Collaborative Filtering (CF): This type of recommendation techniques are designed to mimic a real world peer recommendation behavior. Essentially, this techniques evaluates the other users interests and provides an active user (i.e. the user looking for recommendation) with the list of items he/she may wish to utilize. The rationale behind this technique is that if the active user agreed with some users u in the past, then other recommendations coming from this user u should be relevant to the active user [10]. This technique is one of the most popular ones in the RS research domain [4, 10, 37–39]. In collaborative filtering users are modeled based on their provided opinion on items. More specifically, the user profile is either directly made using users assigned ratings to items, or the system derives a vector of factor values based on the provided ratings. In the latter case users differ in how each factor weights in their profile [2].

The two main methods used in designing collaborative filtering based RS are neighbourhood and model based methods. Neighbourhood-based systems use the stored ratings directly in the prediction, while the model-based approaches use these ratings to learn a predictive model. The neighbourhood-based models can be either user based or item based [10]. In a user based neighbourhood modeling approach, the user similarities are evaluated based on their rated items. Similar users are the ones with similar patterns in rating. The active user in this method is presented with the items that are rated high by similar users [40–42]. In the item based approach, the rating prediction for user u on an item i is based on the other users rating on the items that has similar patterns in rating to i [43, 44].

In the model based approach the rating patterns are used to build a model that learns from the rating patterns to find the latent characteristics of users and items. This model is later used to predict the rating that might be assigned by user u on item i . There are numerous model based approaches used to design RS such as Boltzmann Machines [45], Support Vector Allocation[46], Latent Semantic Analysis [47], Latent Dirichlet Allocation[48] and Singular Value Decomposition [6, 49, 50]. In this research, we extensively describe a method that uses model based approaches. We specifically use Matrix Factorization that is a form of Singular Value Decomposition (SVD) family (see Section 4.3). These family of methods transform users and items to the same latent factor space [4]. The latent space is then used to find the predicted ratings that a user might assign to an item. SVD type models can handle additional information from the data and are able to deal with implicit source of information (See Section 3.3).

Content Based (CB): The content based filtering, as its name implies, uses the content and text explaining the item to predict its relevance to the users [51]. The users are also profiled based on their provided opinion on items (e.g. explicit rating). This technique is mainly used in Information Retrieval research where the users generally provide a query for their search; the items are provided to match both the query and user’s prior behaviour[52]. Items are also characterised based on a text representation technique. Machine Learning techniques are used

to classify items based on the user’s prior implicit or explicit interest on other items[51].

Generally the content based filtering approach contains three main components : *Content Analyser*, *User Profile Learner* and the *Filtering* component [53]. The *Content Analyser* provides a representation for the item based on the items description. In fact, this component transforms the unstructured text of item description into a structured representation. The *User Profile Learner*, uses the user’s past likes or dislikes (or rating levels) to generate user interest profile. This component applies supervised learning algorithms to generate a predictive model to further be used by the filtering component. Profile Learner component works with a list of items along with their assigned user preferences (e.g.ratings). Finally, a new item for an active user is evaluated by content analyser and compared based on items similarity with the highest ranked items for the user (given in profile learner) using the *Filtering* component. The filtering component then ranks the items and provides the recommendation list to the active user [53].

In our prior work [54], we investigated multiple text representation techniques to analyse the content of news articles. We then used various machine learning algorithms with multiple time constraints to build users profile. We evaluated the quality of our recommended item list to the active users based on the prediction accuracy measures. This study was performed in collaboration with a news company on a real world dataset from user interaction log with the news website.

Apart from the two main approaches presented above (i.e.CF and CB) there are also Demographic based [55] , knowledge based [56, 57] and community based [58, 59] approaches presented in the literature.

Hybrid: All RS approaches have advantages and disadvantages for a specific recommendation application [2]. The hybrid filtering approach is suggested to overcome the disadvantages and challenges in one and benefit from the advantages of other approaches. For instance, the collaborative filtering approaches suffer from evaluating and suggesting new items; whereas the new items for content based recommender systems are evaluated based on their descriptions and such

problem does not exist for these types of systems [2]. The hybrid systems are the combination of multiple approaches to overcome the common RS challenges such as cold start, data sparsity, computational power and over specialisation problems [2].

Burk, in his work [60], categorised the hybrid methods into 7 types: (1) weighted (2) switching (3) mixed approach (4) feature combination (5) cascade (6) feature and (7) meta data hybridization approaches. He explains that the use of hybridization approaches improve the accuracy of predictions and improve the functionality of recommender systems in general [60].

Despite all the advancements in computational power and the incorporation of cognitive computing techniques in recommender systems, the area still attracts much attention for improvements of these techniques [3]. This is because there are still ongoing challenges with the design and implementation of recommender systems, such as cold start problem, data sparsity, implicit user preference modeling, etc. (see Section 3.2). Each study contributes to the improvement of the design and implementations of such systems by addressing one or some of these challenges.

One of the main factors in evaluating the performance of recommender systems is the choice of evaluation metrics. In the following section, we review some of the RS related evaluation metrics available in the literature that are out of the scope of the current work.

3.2 Major Challenges in Recommender Systems

The general task of recommendation as mentioned is to provide a list of items to an active user based on his/her historical preferences [2]. A typical structured user preferences data that is generally used by a recommender system can be represented as shown in Table 3.1.

TABLE 3.1: Typical RS data representation example

	Item1	Item2	Item3	Item4	Item5
User1	R_{11}	?	R_{13}	?	R_{15}
User2	?	R_{22}	?	?	?
User3	?	R_{32}	R_{33}	?	R_{35}
User4	R_{41}	?	R_{43}	?	?

As illustrated in Table 3.1, the available training data is not desirable for a learning tasks. The format and availability of RS data source provides common challenges for the design of any recommender system application [3]. In this section, we outline some of these challenges that one should consider, while designing a recommender system.

Data sparsity is one of the biggest challenges for the RS designs [61]. Regardless of the type of RS application the data sparsity problem exists due to the limited number of user feedback on the available items in the system (see Table 3.1). For example, in movie recommendation domain, each user only provides his/her opinion on a handful of movies and the information regarding the rest of the available movies in the system are unknown. The ultimate goal of a RS application is to predict the active user’s opinion on those unknown items and provide him/her with a list of items that are of his/her interest. The limited number of known user interest results in an inaccurate user profiling and ultimately a poor recommendation list for the active user [62]. In the example provided in Table 3.1, the information for *User2* is so scarce that it does not allow for the RS algorithm to model this user’s profile accurately. Therefore, the system is not able to provide a set of useful item recommendations for this user. This affects the overall performance of the recommender system as was also observed in our experiments (see Section 5).

Data sparsity problem can be mitigated by the use of additional information. Implicit information is sometimes used along with explicit ratings to reduce the data sparsity problem [16]. Sometimes the use of hybrid techniques can mitigate the data sparsity problems [60]. Dimensionality reduction techniques are also

proposed to mitigate the data sparsity problem [63], such as the use of Singular Value Decomposition (SVD) [64] or the Principal Component Analysis (PCA) [65].

Cold start problem is also considered as another major challenge in the RS design. This problem occurs when new users enter the system or new items are added to the catalogue [7, 66–68]. In the example provided in Table 3.1, *Item4* does not have any user preferences. If the RS system is designed based on CF approaches, the system is not able to model the item based on not available user-item preference values. The item based cold start problem is minor in recommender systems using CB approaches, as they model the items based on their descriptions. However, for cold start users both approaches suffer from lack of information [7].

There are several techniques to mitigate cold start problem; some includes: gaining insight by (a) asking the users to rate some items at the beginning of their interaction with the application (b) including some extra information about the users from other sources and predict user’s taste from those information (c) asking the users to explicitly specify their taste [10]. However, adding extra information to the system might not be feasible or might induce privacy concerns.

Privacy and security issues are one of the most important challenges in the RS design [3]. While collecting the database for a RS application, users might be reluctant to provide their information including their preferences due to the privacy concerns. Also, competing companies might be reluctant of sharing users information for a cross domain user profiling analysis [69]. There are two main approaches to mitigate the privacy concerns: (a) Encryption and (b) Randomisation [70, 71]. These techniques are normally used in order to protect user specific information in different applications.

Recommender systems that are built on the minimum user preference information are less prone to the privacy invasions. Also, these systems are more generalizable to various recommendation domains. This is because most of the time, in real world applications, the access and extraction of user specific data might not be possible, as they may raise privacy concerns. Therefore, in this research we aim to

design a recommender system algorithm that uses the minimum user preference information, being a single source implicit information.

Scalability is another challenge in the design of RS. As the number of user and item increases the RS should still generate real time recommendations. In fact, a high quality RS is expected to produce relevant recommendations in a timely manner [72]. This requirement highly depends on the implementation of the RS models. Different techniques are proposed in literature, such as dimensionality reduction for SVD [64] the use of Bayesian Networks [73], clustering based RS [63] and etc. Parallelisation of algorithms is also another scalability mitigation technique that is the result of the recent advancements in computations. There are a body of work available on this line of research. some examples for RS parallelisation can be found in [74–76].

In this research, we aim to consider the sparsity problem in most RS applications by using implicit information [16]. In most cases we deal with a sparse matrix and therefore we need to enrich the information content of the model input to improve the prediction accuracy of the model. Implicit user preferences in that sense are used to overcome the negative effect of data sparsity in prediction accuracy. The implicit user preference information and its dependencies are explained in the following section.

3.3 Implicit User Preferences

User preference information can be considered as the core of any recommender system design [77]. This information is the “Transaction” explained in Section 3.1. User preferences can be captured in two main formats; it’s either explicitly provided by the user in the form of likes and dislikes or ratings, or it can be implicitly derived from the user interaction with the system’s application [12].

Recommender Systems are made to decrease the amount of users effort for gathering information. Therefore, in the process of data collection for RS design, we

should not expect that the users provide their preferences explicitly. For example, in an e-commerce application, a user might not be willing to rate a product or explicitly provide his/her opinion on an item. However, based on his/her purchase the RS should be able to predict his/her preferences without requesting for any extra information. In these situations the type of user preference information is implicitly derived from the user interaction with the application [12]. There is a clear trend towards collection and use of implicit information instead of a traditional explicit evaluation of items by ratings[16]. Moreover, Adomavicius and Tuzhilin [7], stated that the future RS should be less intrusive and should rely more on the implicit user feedback to provide recommendations. Hence, in this work, we focus on the RS applications, where the user opinion can be “implicitly” derived from his/her interaction with the system.

Implicit information is normally extracted from user interaction log with the target application (e.g. movie recommendation website or e-commerce application). Some implicit source of information are user clicks, purchases, selections, saves, downloads, time spent on a page and etc [12]. One drawback of this type of information is that it only provides one class information [12]. This means that we are only able to detect the users positive feedback (e.g. clicks, purchase) and the negative feedback can not be detected [12]. Normally lack of positive feedback is considered as negative feedback, which might not always be correct [1]. For example, lack of click on an item might be due to the fact that the user have never observed the item, while it is considered as negative feedback from user in many RS research [8, 27–29].

Although RS design based on explicit information are more straight forward, modeling user preferences from the implicit information can be considered the primary source of information for most recommender systems [27]. However, there is relatively little published research on the implementation and design of the algorithms using implicit user feedback for recommendations [78]. Jawaheer et al. argue that this might be due to the lack of publicly available datasets with implicit user feedback information [78]. However, there are studies transforming the datasets with explicit information into an implicit information representation datasets (referred

to as pseudo implicit user preferences [78]) and they propose solutions to generate recommendations based on implicit user feedback [18, 22, 31, 32, 79, 80].

Some of the studies using implicit source of information transform this type of information into a user preference rating format (an explicit preference information type) and perform their RS analysis on the transformed data. For example, Choi et al. [29], consider a function to transform user's implicit information into explicit rating to be used in recommender systems. They first compute the absolute preference of a user that is relative to the user's purchase frequency in isolation and then compute the relative user preference based on other users preferences. The explicit rating derived from the implicit information is then a function of all user preferences, based on their frequency of purchase. Peska et al. [27] used multiple sources of implicit information to generate a preference relation model. They focused on modeling user's preferences based on different sources of information and use this model to enhance their prediction algorithm. In our prior works [81], we focused on modeling user behaviour based on the implicit information for IBM Watson Analytics application. The data used in this study was extracted from the log of real user interactions with the application. We modelled these interaction with a probabilistic approach based on the current recommendation positions.

One of the first papers investigating the pure use of implicit information, without transforming it into a rating scale, is the work by Claypool et al. [28]. In their work, they compared the impact of different sources of implicit information on the accuracy of recommender systems. In [8] the authors consider the availability of an implicit feedback as a confidence level rather than a pure preference information. Among all studies using pure implicit source of information without data transformation, the Bayesian Personalized Ranking (BPR) approach presented by Rendle et al. [1], gained so much attention, as it shows a more accurate performance compared with other baseline models dealing with this type of information directly [18, 22, 31, 32, 79, 80].

Rendle et al. [1], model the implicit user preferences information to directly optimize a personalized ranking model. They provide a Bayesian formulation for

the recommendation problem based on a Learning to Rank approach. Looking at recommendations as learning to rank problems, allows to differentiate user preferences as their relative importance to each other compared to ratings[78]. In the next section we briefly explain the Learning to Rank approaches used in the design of recommender systems.

3.4 Learning to Rank for Recommender Systems

As explained, the ultimate goal of recommender systems is to provide a ranked relevant list of items for each active user. Therefore, we can consider the goal of recommender systems to be similar to the goal of Learning To Rank (LtR) algorithms in information retrieval [30]. The existing LtR methodologies can be categorised into three main approaches: the **point-wise**, **pair-wise**, and **list-wise** approaches [9].

The **point-wise** ranking method, in its original form, has a loss function that examines the accuracy of prediction based on a predefined label for each document. Point-wise ranking algorithms can be regression, classification or ordinal regression models [9]. The RS made with this approach will then rank items according to their predicted scores. Point-wise LtR for RS methodologies are the collaborative filtering methods that learn to rank based on the preference score of items [82].

The **pair-wise** ranking models, consider the relative preference of a pair of items by a user [9]. In the case of explicit information, for example, item1 is preferred over item2 by user1, if user1 has assigned a higher rating to item1 than item2. Based on this definition, the input space for this approach would be a pair of items for each user. The output space is the label of either +1 or -1, showing whether the first item is preferred to the second or vice versa, respectively. The loss function for this approach detects the inconsistency between the ground truth label and the predicted label. This analogy can be also used with implicit source of information. For example, the purchased products in an e-commerce application can be considered as more preferred to not purchased products. A typical example of

pairwise comparison approach for RS is Bayesian Personalized Ranking (BPR)[1]. This method uses the Bayesian framework to model a pair-wise comparison of items for a RS problem (see Section 3.5 and 4.2).

In the **list-wise** ranking approach, the input space contains an entire group of items and the output space is the ranked list of all items [9]. An important branch of collaborative filtering methods with this approach are designed to directly optimize a ranking metrics, such as Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Normalize Discounted Cumulative Gain (NDCG), etc. Some typical examples are CofiRank [15], CLiMF[14] and TFMAP[83]. The objective function of the LtR algorithms following list-wise approaches can get fairly more complex compared with the point-wise and pair-wise approaches. Also, the sample space for learning could get exponentially large as multiple permutations of items should be considered to generate all possible lists[84]. Hence, the computational complexity of list-wise approach is considerably higher than the two former approaches.

In the next section, we will review the most commonly used pairwise comparison Learning to Rank approach for recommender systems, being Bayesian Personalized Ranking (BPR). BPR is one of the most effective solutions dealing with implicit user preference information [1, 18, 22, 31, 32, 79, 80] and hence is considered as the baseline in our research to design a recommender system, using single source implicit information.

3.5 Bayesian Personalized Ranking- Review

Bayesian Personalized Ranking (BPR) is a pairwise Learning to Rank approach for recommender systems using a single source implicit user preference information. It is a generic approach that can be used with any prediction model. Rendle et al.[1] provide the implementation of the approach for both Matrix Factorization and K- Nearest Neighbors (KNN). The parameters of each model with this approach are optimised based on the correct prediction of pairwise item preferences. The metric used for this optimization is the Area Under the ROC Curve(AUC) [1]. The

details for the implementation of this approach is extensively provided in Section 4.2.

This approach has some limitations and is extensively investigated in the literature. The extensions of this approach are provided to improve its performance for recommender systems [18, 22, 31, 32, 79, 80]. Most of these extensions propose the use of additional information to improve the accuracy of the system [22, 32, 79].

For example, Pan et al. [32] used browsing information and examined the problem of heterogeneous implicit feedback, where the fundamental challenge is the uncertainty of the examination records. They considered that the items that are observed and not purchased are most certainly not preferred by the user compared with the items that have never been viewed by the user. They proposed an Adaptive BPR (ABPR) that reduces the uncertainty on examination records (the observed items) and accurate pairwise preference learning on multiple source of implicit feedback (e.g. browsing information). This makes the RS problem harder as it introduces the uncertainty factor to the learning problem [80].

Qiu et al. [22] used view and likes as an auxiliary information and proposed the Trinity BPR (TBPR) as an extension to the original BPR approach. The TBPR considers two type of pairwise comparisons on three types of items based on different types of user actions. The three types of items are: (1) items that are purchase, (2) items that are viewed (3) items that are not purchased nor viewed. The comparisons are only made between (1) and (2) and between (2) and (3). The problem with this algorithm is the need for extra information that might not always be available.

Wang et al. [79] use additional source of information, such as likes and clicks along with the purchase information to enhance the implicit user feedback and generate better recommendations with their Confidence Learning BPR (CL-BPR) approach. Hence, a confidence factor is added to the original model that makes the problem more complicated. They later extended their original model in [85] with a novel confidence estimation approach that adds another layer to the prediction

model. They also provided a better sampling strategy for pairwise comparison using this extra source of information [79, 86]. However, additional information used in their research might not always be available in all recommendation application domains.

Zhao et al [17], use social network information and proposed Social BPR (SBPR). Their assumption is that users will assign higher preferences to the items that previously was preferred by their friends. This extension needs additional Social networks information as well as user behaviour information for the active user's friends. This additional information is expensive to gain for most of the RS applications. Also, individuals may have different opinions from the friends in their social network [87]. Guo et al. [20], also, use multiple attribute aware approach to enhance the performance of BPR. This model is proposed to deal with cold start problem. The model uses user's attribute information in the learning process along with the collaborative filtering method. The problem with this model is the need for extra user based information that is considered sensitive information. The sampling in this model are also uniformly selected that generates the same limitations as the original BPR.

Some BPR extensions propose different strategies to limit the number of items used for the pairwise comparison purposes and consequently update the BPR's objective function [23, 88].

Pan and Chen [23, 88], assume that group preferences on a single item is a stronger indicator of an individual's preference of one item over another item. The ultimate goal of their algorithm GBPR or GBPR+ is to provide an individual with recommendation rather than providing a group recommendation; however, they consider the group preferences along with individual preferences to relax the assumption that users preferences and items preferences are independent. One main problem with this approach is also the uniform sampling of negative items, which is the limitation of BPR. An additional problem introduced in their algorithm is the random selection of groups, which again arises the concern of selecting a non

informative group for the selected sample, which adds to the BPR approximate inference problem.

All above-mentioned studies use additional information to either refine the objective function or select samples in more informative manner. The problem with those approaches is the need for additional information that might not always be available in a recommendation task. Therefore, these solutions can not be used in the design of a recommender system, using a single source implicit information.

There are some studies addressing the sampling problem for the approximate inference of the posterior distribution in BPR [18, 21]. These studies provide evidence that the system’s performance can increase by the selection of more informative samples without the use of any additional information.

For example, Zhang et al. [21], propose a Dynamic Negative Item Sampling method to mitigate the BPR’s approximation problem (we refer to it as DNS-BPR). They indicate that their approach performs superior to the original BPR, specially on long tailed datasets. Their proposed sampling strategy is completely dependent on the initialisation of parameters and hyper parameters in the scoring model. Hence, the performance of the system is affected from a poor initialisation of the parameters and hyper parameters [89].

The original authors of BPR, Rendle et al., also indicated that the original BPR has some limitations due to the selection of not informative negative samples in their later study [18]. They proposed two non uniform sampling approaches, being static and adaptive sampling strategies to improve the approximation of the posterior probability. They provided evidence that the adaptive sampling improves the limitations of the BPR significantly in terms of accuracy [18]. However, in order to reduce the computational cost they presented a modified version of adaptive sampling and proposed that the sampling should be updated after several number of iterations that is dependent on the number of items. This solution can only be useful for datasets with small item space. Also, this strategy highly depends on the quality of initialisation of the hyper parameters similar to the previous work by Zhang et al. [21]. If the feature learning method cannot obtain high-quality

representations for items, then this kind of adaptive sampler may suffer from a bad performance [89].

In this research, we propose an approach to find the most informative samples to approximate the posterior probability in BPR, using the concept of decision boundaries. Specifically, we propose a similarity based Monte Carlo Markov Chain sampling method to better approximate the posterior probability in BPR. Our proposed approach searches for a decision boundary [33], separating the positive and negative items for a given user. This solution is effective, since as the number of items increase detecting this boundary becomes a harder problem to solve, and hence an efficient approximate method can increase the performance of the model.

Table 3.2 and Table 3.3 provides a summary of the reviewed extensions to the BPR approach, as explained in this section.

TABLE 3.2: Summary of BPR Extensions

Reference	Year	Algorithm Name	Extension Type	Dataset(s)	Evaluation Metric(s)
[18]	2014	Adaptive-BPR	Sampling	BBC ECML'09	MAP@1000 HLU
[21]	2013	DNS-BPR	Sampling	Netflix Yahoo!music LastFM	prec@5 Prec@10 nDCG@5 nDCG@10 MAP
[32]	2015	ABPR	Additional information	Netflix MovieLens 1M	prec@5 nDCG@5 MRR ARP AUC
[17]	2014	SBPR	Additional information	Ciao Lthing Delicious Epinion	Rec@5 Rec@10 nDCG AUC
[22]	2016	TBPR	Additional information	Sobazzar	Prec@5 Rec@5 nDCG AUC MAP MRR

TABLE 3.3: Summary of BPR Extensions

Reference	Year	Algorithm Name	Extension Type	Dataset(s)	Evaluation Metric(s)
[85]	2017	CL-BPR	Additional information	REC-TMALL IJCAI-15	prec@5 prec@10 Rec@5 Rec@10 nDCG MAP AUC MRR
[20]	2015	Maa-BPR	Additional information	MovieLens 100k DBLP	prec@3 prec@10 prec@20 prec@50 prec@100 MAP
[23]	2013	GBPR	Objective Function	MovieLens 100k MovieLens 1M UserTag NF5K5K	prec@5 Rec@5 nDCG@5 1-call@5 AUC
[88]	2016	GBPR+	Objective Function	MovieLens 100k MovieLens 1M UserTag NF5K5K	prec@5 Rec@5 nDCG@5 1-call@5 AUC

Chapter 4

Methodology

In this Chapter, we discuss the details of our methodology along with the datasets used to evaluate the proposed approach. The aim of this study is to design a recommender system algorithm that provides an accurate ranked list of items for each user in the system. This problem is similar to the learning to rank problem in Information Retrieval(IR) literature [30]. The only source of information used for the recommendation algorithm in this research is a single source implicit information, such as user click, item selection or purchase behaviour. One of the most effective learning to rank techniques dealing with implicit information is the pairwise ranking approach, where the items are evaluated in a pairwise manner and the selected items are ranked higher than the not-selected items for each user [1] (see Section 3.4).

To formulate the pairwise comparison learning to rank approach, we let U be the set of all users and I be the set of all items in the dataset. In an implicit feedback setting we observe $S \subseteq U \times I$ a set of actions taken by users $u \in U$ on items $i \in I$. Consider an “active user” as the user who is presented with the list of recommended items generated by a recommender system algorithm. The task of Recommender System (RS) in this context is to provide an active user with a personalized total ranking $>_u \subset I^2$ of all items, where $>_u$ represents the preference of one item over the other by user u .

As explained in Section 3.3, in an implicit source of information setting, only positive feedback is available for the selected items and the user preference for other items in the system is missing. Among these missing information, some are actually missing, while others are not preferred by the user. The actual missing information is related to the items that were never seen by the user. The not preferred items, on the other hand, are the items that were presented to the user but the user did not click, select or purchase them. However, most of the time it is not possible to find the the information on the items that were offered to the user, but were not selected by the user.

A simple solution to tackle this missing information is to ignore it. However, such a sparse dataset (i.e. having only positive preferences) would not provide sufficient amount of data to train a learning algorithm, and would also require a more complex model to generalise. An alternative solution would be to consider all missing values as not preferred. This approach considers both choices that are not seen, and choices that are seen but not selected as missing items. In other words, all missing items are considered as items that are not preferred. This is an imbalanced data problem and hence the prediction model learns the majority class (the missing items) as being always not preferred. In this case, the model will overfit to always consider a small portion of items (the positive class). Therefore, the actual missing items (i.e. not seen items) would never be recommended to the user. The only scenario where these algorithms may provide good recommendations is when regularization factors are used to mitigate overfitting tendency of the learning algorithm. The regularisation factors can be thought of as a way of shrinking the coefficients of the parameters that balance fit and complexity of the model, allowing both accurate prediction and generality [90].

One of the most effective solutions to tackle the one class implicit information problem is the use pairwise comparison ranking approach. In a pairwise comparison approach, a pair of items are evaluated against each other for each user. This

allows to distinguish between the missing items and the items that are not preferred, if the information is available. Figure 4.1, provides an illustration of the differences between the approaches that are explained above.

FIGURE 4.1: Schematic representation of data from implicit information and the transformation solutions. (a) consider all missing as not preferred (b) pairwise comparison approach

In Figure 4.1 part (a), all positive feedback are considered as having a value of 1 and all missing values (i.e the question marks) are considered as negative feedback, and are assigned a value of 0. For example, consider in an e-commerce application, user u_1 observes both item i_1 and item i_2 ; though he purchases item i_2 but not item i_1 . In the first implicit information representation (4.1 part (a)), the information that u_1 did not select i_1 , although he observed it, is missing. On the contrary, in a pairwise comparison framework the distinction between the two scenarios is considered. In Figure 4.1 part(b), each matrix on the right hand side belongs to one single user. The same example of u_1 here can be observed in the top right table. This table shows that u_1 preferred i_2 over i_1 that is now represented by j_1 as a row of this matrix. The pairwise ranking approach highlights the actual missing information occurs in two cases: (1) When two items are purchased, for instance, i_2 and i_3 for u_1 , we are not able to understand which of the two is

preferred more by the user. (2) Similarly we can not understand which of the two items is less preferred in the case of two items that are not purchased by the user such as i_4 and i_5 for u_1 . Normally, publicly available datasets to evaluate a recommender system algorithm only contains the positive items (i.e. the items that are purchased) and do not distinguish between the items that are shown to the user and not purchased and the items that were never shown to the users. The pairwise comparison approach in this case considers all not purchased/selected items as being less preferred by a user compared with the items that are purchased/selected by that user.

One of the most common pair-wise comparison approaches for item recommendations is Bayesian Personalized Ranking (BPR) [18, 22, 31, 32, 79, 80]. BPR provides a generic optimization criterion that can be used with any rating algorithm, such as Matrix Factorization or K Nearest Neighbours to provide item recommendations. BPR follows the Bayesian machine learning framework to account for uncertainties in both the available data and user behaviour measurements. The gathered user preference data is effected by sources of uncertainties, such as biases in user preference interpretation. An example can be referred to as considering all not selected items as not preferred items, regardless of whether the user was presented with the items or not. Also, users provide their opinion only on the items that are presented to them. Hence, there might be uncertainties in their behaviour in terms of their provided preferences that should be taken into account.

In the Bayesian Machine Learning framework, approximate inference methods play a significant role in providing accurate predictions using Bayesian models [24]. In this work, we improve the accuracy of the BPR approach by incorporating a novel method to approximate the posterior probability in BPR. Our proposed model does not use any additional information. Similar to the BPR approach, it only uses a single source implicit information, the selection of items (i.e. clicks), which is the most readily available type of data typically used RS domain. Our proposed method Similarity based Monte Carlo Bayesian Personalized Ranking (SMC-BPR), in essence searches for a decision boundary [33] separating the positive and negative items for a given user. As the number of items increase detecting

this boundary becomes a harder problem to solve exactly. This leads researchers to come up with efficient approximate methods. SMC-BPR presents an alternative approximate solution to find the most informative negative items in order to better estimate the decision boundary between the positive and negative items in BPR framework.

In this Chapter, we start by explaining the basics of Bayesian Machine Learning including Bayesian Inference and approximation methods (Section 4.1). This builds a baseline for describing the BPR method in Section 4.2. We then formulate the main research problem addressed in this research in Section 4.4. This is followed by the details of our proposed algorithm in Section 4.5.

4.1 Bayesian Inference

One of the main concepts in the field of pattern recognition and Machine Learning is uncertainty [24]. This uncertainty may be due to the lack of available information or due to the noise in the measurements. In this section we will review how Bayes' theorem plays a role in Machine Learning to tackle the problem of uncertainty in learning and inference.

In a Bayesian Machine Learning framework we formulate our knowledge of a phenomenon in a probabilistic manner [91]. We define a model to formulate our knowledge from the phenomenon. The model has some unknown **parameters** that should be defined based on an observed data. We also specify a **prior** probability for these unknown parameters. These priors should express our belief about those parameters before observing any data. We then gather data and compute a **posterior** probability distribution for the parameters using the observed data. We use the posterior distribution to either reach a scientific conclusion, make predictions by averaging the posterior distributions or make decisions to minimise the posterior loss [91].

To find the posterior distribution for model parameters given the observed data, a product of the likelihood function and the priors is considered following Bayes rule from the rules of probability.

4.1.1 Bayesian Probability

Consider a curve fitting problem, where $\mathbf{w}$ is a set of parameters of the model that fits the distribution. The assumptions for $\mathbf{w}$, before observing the data is referred to as *prior* distribution of $\mathbf{w}$, being $p(\mathbf{w})$. The effect of the observed data D on $\mathbf{w}$ can be represented with a conditional probability of $p(D|\mathbf{w})$. The Bayes theorem can then be used to compute the uncertainties on $\mathbf{w}$ after the observation of the data D ; this is referred to as *posterior* probability, being $p(\mathbf{w}|D)$.

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)} \quad (4.1)$$

The $p(D|\mathbf{w})$ in Equation 4.1, shows the probability of the observed dataset in different settings for $\mathbf{w}$ and is referred to as *likelihood function*. The $p(D)$ in this equation is the normalisation constant that ensures that the posterior probability integrates to one. Therefore, the posterior probability can be rewritten as:

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \quad (4.2)$$

Integrating both side of the Bayes theorem in Equation 4.1 with regards to $\mathbf{w}$ we can represent the denominator as shown in Equation 4.3.

$$p(D) = \int p(D|\mathbf{w})p(\mathbf{w})d\mathbf{w} \quad (4.3)$$

In the frequentist approach, $\mathbf{w}$ is considered as a fixed parameter, whose value is estimated with an estimator that minimises the error based on the given data D . On the contrary, in the Bayesian framework, the observed data D is used to

estimate the uncertainties in the parameters $\mathbf{w}$ through a probability distribution over $\mathbf{w}$ [24]. The parameters of a non-Bayesian approach are defined in a way that the observed data has the highest probability for the model with those parameters. This is in contrary to the Bayesian approaches, where the parameters are selected to maximise the posterior probability [24].

4.1.2 Bayesian Machine Learning

Machine learning is designing a computer program that optimizes certain performance criteria using a sample data or past experiences [33]. Essentially, the computer program is a model with certain parameters and learning is the optimization of the parameters of that model based on certain criterion using the historical data. The model can be predictive and make predictions about the future or it may be descriptive and explain a phenomenon or it can be both predictive and descriptive at the same time [33].

In Bayesian Machine Learning, all components of the model, such as the model parameters are treated as random variables, where their uncertainties are represented by a probability distribution. The core principles of Bayesian machine learning is as follow.

Consider a model M with parameters Θ . Before making any observations, the *prior* probability of Θ over the model is $p(\Theta|M)$. After the observation of data D , the parameters Θ will be updated based on $p(\Theta|D, M)$. This *posterior* probability can be computed using Bayes theorem.

$$p(\Theta|D, M) = \frac{p(D|\Theta, M)p(\Theta|M)}{p(D|M)} \quad (4.4)$$

In Equation 4.4, the quantity $p(D|\Theta, M)$ is the *likelihood function*, that provides the probability of how well parameters Θ explain the data D . The $p(D|M)$ is referred to as *model evidence* or *marginal likelihood*. The model evidence shows how appropriate the model is for a given data.

To make predictions on an unobserved data D^* , based on the probability theory, we should integrate over all the sources of uncertainties. The predictive distribution is derived from Equation 4.5.

$$p(D^*|D, M) = \int p(D^*|\Theta, M)p(\Theta|D, M)d\Theta \quad (4.5)$$

The likelihood function (i.e. $p(D^*|\Theta, M)$) in this integration accounts for the observation noise, while the $p(\Theta|D, M)$ accounts for parameter uncertainties.

In this work, we built our proposed Bayesian ML algorithm as an extension to the BPR model (see Section 4.2). Our proposed model, although a bit more complex, provide a better approximation and therefore it attains higher accuracy compared to the original BPR model.

4.1.3 Approximate Inference

In Bayesian models, often the posterior and model evidence are computationally intractable [24, 92, 93]. Therefore, there is a need for techniques to approximate these values. The process of finding an approximate for a posterior distribution is referred to as *approximate Bayesian Inference* [93].

The two main classes of approximate inference are **Variational Inference** and approximations based on **numerical sampling** [93]. Variational inference is parametric approximation, in which the posterior is replaced by a computationally convenient distribution. Then, the objective is set to minimise the divergence between the actual and approximated distribution. Some of the existing approaches include Variational Bayes [48], Expectation Propagation[94], Laplace Approximation [95] and mean field estimation [96]. These methods differ in their underlying objective function that they try to minimise.

On the contrary, the approximate inference based on numerical sampling, uses a subset of data to approximate the posterior distribution. Most of sampling techniques to approximate complex distributions are based on Markov Chain Monte

Carlo method [24]. This method is the basis of our proposed algorithm and is explained in the following section. (see Section 4.1.4).

4.1.4 Markov Chain Monte Carlo Sampling

For most Bayesian machine learning problems, the posterior distribution is required primarily for the purpose of evaluating the expectations to make predictions [24]. Also, as explained in the previous sections, the quantification of the posterior is generally intractable. Therefore, there is a need for an approximation method to solve the problem in hand. In this section, we explain the approximate inference method based on numerical sampling using Markov Chain Monte Carlo (MCMC) method.

In general, the posterior approximation problem can be solved by finding an expectation of some function $f(z)$ with respect to a probability distribution $p(z)$ [24]. If $p(z)$ is a continuous distribution the expectation is evaluated with Equation 4.6.

$$\mathbb{E}[f] = \int f(z)p(z)dz \quad (4.6)$$

If $p(z)$ is a discrete variable, then the integration is replaced by summation. The general idea behind sampling methods is to select a set of samples $z^{(l)}$, ($l \in \{1, \dots, L\}$, where L is the number of samples) that are drawn independently from the distribution $p(z)$. Therefore the Expectation in Equation 4.6 can be approximated by a finite sum in Equation 4.7.

$$\hat{f} = \frac{1}{L} \sum_{l=1}^L f(z^{(l)}) \quad (4.7)$$

This is referred to as the classical Monte Carlo approximation [24]. If sample $z^{(l)}$ is drawn from $p(z)$, the $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$. Therefore, the estimator $\hat{f}$ has the correct mean, and the variance of function $f(z)$ under the distribution $p(z)$ is computed with Equation 4.8.

$$\text{var}[\hat{f}] = \frac{1}{L} \mathbb{E}[(f - \mathbb{E}[f])^2] \quad (4.8)$$

This shows that the accuracy of the estimator does not depend on the dimensionality of z and high accuracy maybe achieved with a small number of samples $z^{(l)}$ [24]. However, the problem is that some samples $z^{(l)}$ might not be effective and hence the sample size used for the analysis might be so much smaller than the actual sample size [24].

There are several sampling methods available in the literature [24], such as rejection sampling [97], importance sampling [98] and Gibbs sampling [99]. These sampling techniques have some limitations, specially when the actual distribution is high dimensional and complex. On the other hand, Markov Chain Monte Carlo (MCMC) provides a general and powerful framework for sampling [24]. It is one of the most well known general solutions for several high-dimensional problems [100, 101]. MCMC techniques are applied to solve integration and optimization problems in the fields of machine learning, physics, statistics, econometrics and decision analysis [102].

The advantage of the MCMC sampling over the simple sampling techniques (e.g. importance or rejection sampling) is that the algorithm keeps the record of the current sample [24]. If the next sample does not satisfy some requirements of a good sample then the new sample is discarded and the old sample is replaced as the current sample. Following paragraphs explain MCMC in detail.

Consider we would like to sample from a complicated distribution $p(\mathbf{z})$. Assume that the value of $\mathbf{z}$ can be easily computed upto a normalizing constant. This means that $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$, where $\tilde{p}(\mathbf{z})$ is the value that can be computed and the Z_p is the normalizing constant that can not be computed and is unknown. In all approximation sampling methods a simpler distribution is considered that is referred to as *proposal distribution* $q(\mathbf{z})$. The candidate sample is easily drawn from the proposal distribution. If the candidate sample satisfies certain criterion, then it is accepted, otherwise it is rejected.

In the case of MCMC, the proposal distribution is dependant on the previous stage and would be in the form $q(\mathbf{z}|\mathbf{z}^{(\tau)})$. The $\mathbf{z}^{(\tau)}$ is the sample in the current state that is derived from the proposal distribution. As shown, the proposal distribution is dependent on the current state and hence the sequence of samples $\mathbf{z}^{(1)}$, $\mathbf{z}^{(2)}$, ... forms the Markov Chain [24]. The candidate sample is referred to as $\mathbf{z}^*$ is derived from the proposal distribution. $\mathbf{z}^*$ is evaluated based on certain criterion. If it satisfies the requirements of the criterion, it is accepted and the $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$; otherwise the candidate sample $\mathbf{z}^*$ is discarded and $\mathbf{z}^{(\tau+1)} = \mathbf{z}^{(\tau)}$.

Metropolis algorithm is one of the MCMC methods of sampling. In this method, the proposal distribution is considered to be symmetric [103]. This means that for any given samples $\mathbf{z}_A$ and $\mathbf{z}_B$, we have $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$. The candidate sample is accepted based on the following criterion:

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min(1, \frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})}) \quad (4.9)$$

The $A(\mathbf{z}^*, \mathbf{z}^{(\tau)})$ is the *acceptance probability* of the candidate sample $\mathbf{z}^*$. The process is that we select a random number u from a uniform distribution over a unit interval (0,1). This will be the threshold of acceptance. The candidate sample is accepted if $A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) > u$. If the candidate sample is accepted, then the $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$, otherwise $\mathbf{z}^{(\tau+1)}$ is set to $\mathbf{z}^{(\tau)}$ and another candidate sample is drawn from $q(\mathbf{z}|\mathbf{z}^{(\tau+1)})$. As the $\tau \rightarrow \infty$ the distribution of $\mathbf{z}^{(\tau)}$ tends to $p(\mathbf{z})$.

A more general form of this algorithm is Metropolis Hasting [104], in which the transition distribution is not symmetric. The Acceptance probability for this algorithm is as shown in Equation 4.10

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min(1, \frac{\tilde{p}(\mathbf{z}^*)q(\mathbf{z}^{(\tau)}|\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})q(\mathbf{z}^*|\mathbf{z}^{(\tau)})}) \quad (4.10)$$

In this work we considered a uniform transition distribution and therefore the algorithm we used for approximation is Metropolis algorithm. The details of our proposed method is outlined in Section 4.5.

4.2 Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) provides a generic optimization criterion that can be applied on any recommender system (scoring based) algorithm [1]. The authors provided an example implementation with matrix factorization as well as K Nearest Neighbours (KNN) algorithms [1]. BPR-OPT is the Bayesian analysis representation of the pair-wise ranking problem. The authors further present a learning algorithm called LEARNBPR to learn the BPR-OPT optimization criterion [1]. The details of this algorithm is presented in this section.

In the pairwise comparison approach, for each user u we would have a positive item i that is the selected item by user u and a negative item j . Item j is an item in the dataset that is not selected by user u . The final dataset includes all items that are selected by each user. The items selected by other users (u 's) and not by the active user (u) are considered as negative items for the active user. Consider the dataset $D_S : U \times I \times I$, where :

$$D_S := \{(u, i, j) \mid i \in I_u^+ \wedge j \in I \setminus I_u^+\} \quad (4.11)$$

In the recommender system framework, we would like to rank items based on user preferences. In the pairwise ranking approach, essentially, we model the user preferences in way that we assign higher rank to positive items compared with the negative items. For example, consider in a movie recommendation domain, a user selects "titanic" movie, while she does not select "superman" from a list of movies. The designed model should assign a higher rank to "titanic" compared with the "superman" movie for this user. In this example "titanic" is considered as the positive item i and the "superman" is considered as negative item j .

The parameters of the model that formulates such phenomenon should be optimised based on the correct prediction of assigning positive items over the negative ones. This is contrary to the point wise predictions, where the model parameters

are optimised based on the prediction of the actual score of each item by each user.

In a Bayesian Machine Learning framework, as discussed in the previous Section 4.1, the model parameters are computed based on a posterior probability distribution given the observed data [24]. This posterior is proportional to the product of the likelihood of the observed data given the parameters and the prior probability distribution of the parameters.

$$p(\Theta|D) \propto p(D|\Theta)p(\Theta) \quad (4.12)$$

Here in Equation 4.12 D is the observed data and Θ shows all the parameters of the model. To formulate the pair-wise comparison framework for learning to rank Rendle et al. [1] provides the following Equation 4.13:

$$p(\Theta |>_u) \propto p(>_u | \Theta)p(\Theta) \quad (4.13)$$

In Equation 4.13, Θ is the parameter vector of an arbitrary model such as Matrix Factorization. $>_u$ is the desired assignment of a positive item i over a negative item j for each user u that shows the latent preference of the user u . This means that the model should be formulated in a way that positive items are assigned higher ranks compared with the negative items. The posterior probability, $p(\Theta |>_u)$, is computed as a proportion of the product of the likelihood function and the prior beliefs of the parameters.

The likelihood function for all users can be written as two likelihood products over all combination of user, positive item and negative item. These combinations include all the observed and un-observed combinations of $\langle \text{user}, \text{positive item}, \text{negative item} \rangle$ in the dataset.

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u,i,j) \in U \times I \times I} p(i >_u j | \Theta)^{\delta((u,i,j) \in D_S)} \cdot (1 - p(i >_u j | \Theta))^{\delta((u,i,j) \notin D_S)} \quad (4.14)$$

The first term on the right hand side of this equation, shows the probability of the correct assignment of i s over j s in the observed data. The second term, computes the probability of the incorrect assignment of i over j for an un-ordered (i,j) pair due to an un-observed data point (i.e $(u, j, i) \notin D_S$). Therefore, the second term is computed as one minus the probability of the correct assignment of i over j regardless of the order. In this equation δ is the indicator function being:

$$\delta(b) := \begin{cases} 1 & \text{if } b \text{ is true} \\ 0 & \text{else} \end{cases} \quad (4.15)$$

A sound pairwise ranking scheme follows certain properties. totality and antisymmetry are two of these properties:

$$\begin{aligned} \forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i & \quad (\text{totality}), \\ \forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j & \quad (\text{antisymmetry}). \end{aligned} \quad (4.16)$$

The totality indicates that if an item exist in the item list I , it can be considered either preferred or not preferred for the user. The antisymmetry shows that if i is preferred to j by a user and j is preferred to i by the same user, then both items are the same. Based on these two properties the likelihood function can be simplified as Equation 4.17.

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u,i,j) \in D_S} (p(i >_u j | \Theta)) \quad (4.17)$$

Another property of sound pairwise ordering is referred to as transitivity that is shown in Equation 4.18.

$$\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \quad (\text{transitivity}). \quad (4.18)$$

The simplified likelihood function in Equation 4.17 does not guarantee a personalized ranked order [1]. In order to achieve that, all properties of total order (i.e. totality, antisymmetry and transitivity) should be fulfilled. Therefore, Rendle et al. [1] proposed the single probability that a user prefers one item over the other as shown in Equation 4.19:

$$p(i >_u j \mid \Theta) := \sigma(\hat{x}_{u,i,j}(\Theta)). \quad (4.19)$$

Here, σ is the logistic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4.20)$$

The $\hat{x}_{uij}$ in Equation 4.19 is the function that explains the relationship between user u , selected item (i.e. positive item) i and the not selected item (i.e. negative item) j . This function is based on a scoring function, such as matrix factorization. The scoring function predicts the relative score of each item given by each user. In the pairwise ranking framework, the differences between the item's scores is used to predict the item's rank. The properties of Matrix Factorization algorithm is explained in detail in Section 4.3.

To complete the Bayesian formulation of the ranking problem, Rendle et al. [1] consider the parameter's prior to follow a Gaussian distribution with 0 mean and standard deviation of Σ_{Θ} .

$$p(\Theta) = N(0, \Sigma_{\Theta}) \quad (4.21)$$

The BPR optimization criteria (BPR-OPT) is used to optimize the parameters of the model being Θ . This optimization is based on the formulation of maximum posterior estimator [1] and is suggested as follows:

$$\begin{aligned}
 BPR - OPT &:= \ln p(\Theta | >_u) \\
 &= \ln p(>_u | \Theta) p(\Theta) \\
 &= \ln \prod_{(u,i,j) \in D_s} \sigma(\hat{x}_{uij}) p(\Theta) \quad (4.22) \\
 &= \sum_{(u,i,j) \in D_s} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
 &= \sum_{(u,i,j) \in D_s} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2
 \end{aligned}$$

In this Equation 4.22, to reduce the number of unknown hyper parameters Rendle et al. [1] suggested $\Sigma_\Theta = \lambda_\Theta I$ and therefore the prior was formulated as follows:

$$p(\Theta) = N(0, \lambda_\Theta I) \quad (4.23)$$

λ_Θ is the regularisation rates for parameters in the scoring function. The regularization rates are normally used in prediction models to avoid overfitting of the model on the given data. Regularisation rates are used as the coefficient of the latent parameters in matrix factorization algorithm as explained in Section 4.3.

The parameters of BPR are learned by considering the Area Under the ROC Curve (AUC) of the predicted rank for the items. AUC measures the correct detection of a user's preference. This means how well the model can predict that a user u preferred a positive item i to a negative item j . To demonstrate how AUC is related to the $\ln \sigma$ function in the BPR-OPT criterion, let's compute the AUC for each user (Equation 4.24):

$$AUC_u = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0) \quad (4.24)$$

Here, again the $\hat{x}_{uij}$ comes from the differences between the predicted scores for the positive item i and negative item j for user u . If this difference is positive then the desired order of i over j is achieved. Then the average of AUC for all users would be:

$$AUC := \frac{1}{|U|} \sum_{u \in U} AUC_u \quad (4.25)$$

The AUC for each user (Equation 4.24) can be re-written as Equation 4.26 for the (u, i, j) in the observed dataset D_S :

$$AUC_u = \sum_{(u,i,j) \in D_S} z_u \delta(\hat{x}_{uij} > 0) \quad (4.26)$$

In this equation z_u is a normalizing constant $\frac{1}{|U||I_u^+||I \setminus I_u^+|}$ and $\delta(x > 0)$ is considered as Heaviside function being:

$$\delta(x > 0) = H(x) := \begin{cases} 1 & x > 0 \\ 0 & else \end{cases} \quad (4.27)$$

This equation (Equation 4.26) is similar to the equation for the BPR optimization (Equation 4.22). The difference is the use of a differentiable function $\ln \sigma$ function in BPR-OPT versus the non-differentiable function δ in the AUC evaluation. The σ or the more smooth $\ln \sigma$ function is normally used as an approximation for the non-differentiable Heaviside (δ) function [105]. This means that the BPR optimization criterion is an approximation of the correctly ranked items for each user based on the AUC measure.

In order to learn the BPR optimization criterion provided in Equation 4.22, the authors in [1], propose a stochastic gradient descent based algorithm with bootstrap sampling. They refer to this learning algorithm as LERNBPR. The gradient of BPR-OPT with respect to the parameters Θ is :

$$\begin{aligned}
\frac{\partial BPR - OPT}{\partial \Theta} &= \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_{\Theta} \frac{\partial}{\partial \Theta} \|\Theta\|^2 \\
&\propto \sum_{(u,i,j) \in D_S} \frac{-e^{\hat{x}_{uij}}}{1 + e^{\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_{\Theta} \Theta
\end{aligned} \tag{4.28}$$

This gradient is then used to update the parameters of the model (i.e. Θ). Gradient Descent algorithms are one of the most popular algorithms for parameter learning [106]. The full gradient Descent can not be used in the BPR-OPT problem as this algorithm requires the full gradient computation over all training triplet in the observation data. The parameters are then updated with the full gradient. Although the gradient would be on the correct direction, this approach yields to very slow convergence. This is because the analysis of the full gradient of all combinations triplets of $(u, i, j) \in D_S$ (i.e. $O(|S||I|)$) is computationally expensive. This approach also generates a biased update as the gradient of the highly skewed items (i.e. popular items that are considered as positive items for many users in D_S) would dominate the update function for all triplets [1].

Therefore, the authors in [1], proposed the use of Stochastic Gradient Descent (SGD) to update the parameters in BPR model. This is because the Stochastic Gradient Descent updates the parameters for each triplet as soon as the gradient is computed for that specific instance.

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_{\Theta} \Theta \right) \tag{4.29}$$

The SGD mitigates the skewness problem, however the computational complexity still exist. The authors in [1] also proposed the use of bootstrap sampling of the triplets with replacement to decrease the computational cost. They argue that often a fraction of a full cycle is sufficient for convergence [1]. This is the idea behind the approximation using sampling techniques in Bayesian machine learning [24]. The bootstrap sampling reduces the large amount of training samples and therefore the computational complexity is reduced. The pseudo code for LearnBPR is provided in Algorithm 1.

Algorithm 1 Pseudocode for the original BPR method that learns to distinguish the selected items (i.e. positive items) $(u, i) \in S$ from other items $j \in I \setminus I^+(u)$ in the dataset[1].

```

1: procedure LEARNBPR( $\alpha, S$ )
2:   Randomly initialize  $\Theta$ 
3:   repeat
4:     Draw  $(u, i) \in S$  uniformly
5:     Draw  $j \in I \setminus I^+(u)$  uniformly
6:     for  $\theta$  in  $\Theta$  do
7:       Update  $\theta$ 
8:     end for
9:   until convergence
10:  return  $\Theta$ 
11: end procedure

```

As explained above, while learning BPR the parameters of the BPR model, Θ , should be updated. The BPR is a general learning algorithm that can be used with any prediction model. The authors in [1], provide an example of kNN as well as Matrix Factorization (MF) model as the scoring function. The use of MF model is widespread in the literature [1, 18, 22, 31, 32, 79, 80] and therefore we provide our proposed solution using MF model in this work. The following Section provides a detailed explanation 4.3. Our proposed solution can also be generalised using other scoring functions such as kNN.

4.3 Scoring Function with Matrix Factorization

Matrix Factorization (MF) has become a popular algorithm that is used in collaborative filtering models [2, 4, 8, 81, 107]. The popularity of Matrix Factorization dates back to its superior performance in Netflix Prize competition in 2006 [4]. One of the most important factors that differentiate MF model from the other models is that it scales well when dealing with large datasets through parallel computing [2].

The problem of predicting $\hat{x}_{uij}$ can be decomposed into the prediction of $\hat{x}_{ui} - \hat{x}_{uj}$. $\hat{x}_{ui}$ is the predicted score for item i by user u and $\hat{x}_{uj}$ is the predicted score of item j by user u . The problem of predicting $\hat{x}_{ui}$ can be considered as finding a

matrix $X : U \times I$. Matrix Factorization models approximates X from the two low ranked matrices P and Q . each row in P represents a latent factor vector for user u (being p_u) and each row in Q represents a latent factor vector for item i (being q_i). The prediction of an unknown $\hat{x}_{ui}$ can be estimated by the dot product of these two matrices:

$$\hat{x}_{ui} = \sum_{f=1}^k p_{uf} \cdot q_{if} \quad (4.30)$$

Where, k is the dimension of the latent factors that should be initialised for each dataset. An earlier model using Singular Value Decomposition (SVD) was proposed by Sarwar et al. [107]. Figure 4.2 provides a schematic overview of the SVD model. Since this model was prone to overfitting, Hu et al. and Pan et al. extended it by adding a regularization term to overcome the overfitting problem [8, 108].

FIGURE 4.2: Score Prediction based on Matrix Factorization

As explained earlier, in MF algorithm users and items are modelled by factor vectors p_u and q_i , respectively [4]. Hence, a predicted score $\hat{x}_{ui}$ by user u for item i is the dot product of the transpose of item's vector by the user's vector:

$$\hat{x}_{ui} = q_i^T \cdot p_u. \quad (4.31)$$

As we discussed in the earlier sections using only the observed scores in a dataset (D_S) to make predictions is prone to overfitting. Therefore, a regularization factor is normally added to the prediction model. λ in Equation 4.32 is the regularisation factor. Regularisation factors are used for prediction models in the cases when the

dataset is imbalanced [109]. In a matrix factorization framework, when the number of observations differs significantly among rows and columns (i.e. imbalance data), a single number of feature dimension might be too large for some observations and too small for the others. To have a more generalizable model regularization factors are added to the model. The value of these factors can be obtained based on multiple implementation of the model with various regularization rate values. The best value is selected based on the best performing model on the validation set [109].

$$\hat{x}_{ui} = q_i^T \cdot p_u + \lambda(\|q_i\|^2 + \|p_u\|^2). \quad (4.32)$$

To learn the factor vectors for both users and items, the system minimises the regularised squared error on the set of observed scores in the training set using an optimization method. In our work, we use Stochastic Gradient Descent (SGD). SGD for Matrix Factorization models was proven to be successful in the context of Learning to Rank problems [4]. The algorithm iterates through the observations in order to minimise the prediction accuracy of the user preferences.

Even with the regularisation factors the model may still overfit to specific user or item [50]. The bias feature idea in the context of Recommender Systems with matrix factorization was first introduced by Paterek [50]. Accounting for these biases would improve accuracy drastically for rating prediction. User biases are considered to account for the differences in user behaviour. It is quantified by the deviation of the mean of a specific user's preference degree (i.e. rating) from the global mean of all user preferences. These biases account for the users who tend to have extremely high or extremely low preference degrees. Apart from user biases item biases are also included in the matrix factorization model. Item biases are considered as the deviation of the mean of an item's preference degree from the mean of all other items preference degrees. Including this bias will also alleviate the impact of overly popular items on the general rating prediction. The refined matrix factorization algorithm with basic biases is given by:

$$\hat{x}_{ui} = q_i^T \cdot p_u + b_i + b_u + \mu, \quad (4.33)$$

where μ is the average rating, and b_u and b_i are the user and item biases, respectively. While learning, the regularisation factor should also be applied to the biases. Hence, the system learns by minimising the squared error function:

$$\min \sum (x_{ui} - q_i^T \cdot p_u + b_i + b_u + \mu)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2). \quad (4.34)$$

In an optimization problem for simple prediction of $\hat{x}_{ui}$, Stochastic Gradient Descent (SGD) algorithm loops through all user preferences (i.e. ratings) in the training set and calculates estimated preferences for each user-item pair. Then, the error associated with the prediction is calculated (e_{ui}). Consequently, it modifies parameters of p_u and q_i by a magnitude proportional to a predefined learning rate α in the opposite direction of the gradient. Learning rate for each dataset and model should also be predefined and optimized prior to the final presentation of the results. b_u and b_i are also updated using similar learning rate. The SGD optimization update for each parameter in x_{ui} is summarized as follows:

x_{ui} parameter update :

$$\begin{aligned} p_u &\leftarrow p_u + \alpha(e_{ui} \cdot q_i - \alpha \cdot p_u), \\ q_i &\leftarrow q_i + \alpha(e_{ui} \cdot p_u - \alpha \cdot q_i), \\ b_i &\leftarrow b_i + \alpha(e_{ui} - \alpha \cdot b_i), \\ b_u &\leftarrow b_u + \alpha(e_{ui} - \alpha \cdot b_u). \end{aligned} \quad (4.35)$$

This update will be computed until convergence. There are several hyperparameters that should be set at the initialization of this algorithm. These hyper parameters as explained are regularization rate and learning rate as well as the number of iterations and number of features[110] that should be optimized in the validation phase [33].

As was presented in Section 4.2, the LearnBPR optimizes the parameters of the MF model based on the pairwise ranking approach. Therefore, the above parameter learning would be on the error associated with the $\hat{x}_{uij}$ for each triplet $(u, i, j) \in D_S$. Also, as indicated the number of (u, i, j) triplet combinations is massive and the authors in [1] proposed a bootstrap random sampling of the triplets. Using this sampling provides an approximation of the actual function. In the following section we outline the problem that occurs by random selection of samples for the LearnBPR approach. This section is followed by the descriptions of our proposed method to overcome the problem.

4.4 Problem Formulation

In the pairwise learning framework, the model parameters Θ are learned from the comparison of the positive items (e.g. selected items $I^+(u)$) and the rest of the items ($I \setminus I^+(u)$) in the observed dataset D_S . The basic idea is to find a function that shows item i is preferred to item j if and only if item i was selected by user u , while item j was not selected by user u . This analogy is shown in Equation 4.36.

$$i \succ_u j \Leftrightarrow i \in I^+(u) \wedge j \in I \setminus I^+(u). \quad (4.36)$$

Each observation for the pairwise preference evaluation is derived from a data space of $D_S \subset U \times I \times I$ and can be formulated as:

$$(u, i, j) \in D_S :\Leftrightarrow i \in I^+(u) \wedge j \in I \setminus I^+(u). \quad (4.37)$$

As explained, the pairwise preference based on BPR can be modeled with a sigmoid function over the scoring model (Equation 4.19) as is re-stated in Equation 4.38:

$$p(i \succ_u j) := \sigma(\hat{x}_{ui} - \hat{x}_{uj}) \quad (4.38)$$

The optimization criteria is to maximize the likelihood of correctly ordering the preferences as shown in Equation 4.39.

$$\underset{\Theta}{argmax} \prod_{(u,i,j) \in D_S} p(i \succ_u j), \quad (4.39)$$

This is equivalent to minimizing the negative log of the likelihood (i.e. NLL):

$$NLL := - \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{ui} - \hat{x}_{uj}). \quad (4.40)$$

As explained, in the BPR approach the model parameters Θ are updated based on the gradient descent algorithm. The gradient of a model parameter $\theta \in \Theta$ in the BPR framework can be computed based on Equation 4.41:

$$\frac{\partial NLL}{\partial \theta} := \sum_{(u,i,j) \in D_S} (1 - \sigma(\hat{x}_{ui} - \hat{x}_{uj})) \frac{\partial(\hat{x}_{ui} - \hat{x}_{uj})}{\partial \theta} \quad (4.41)$$

Since the number of pairs in the sample space is very large the use of an exact model including all item pairs is computationally expensive. Therefore, in the original BPR approach a bootstrap sampling with replacement from D_S is suggested to reduce the computational complexity. The parameters $\theta \in \Theta$ are updated using SGD as:

$$\theta \leftarrow \theta - \underbrace{\alpha((1 - \sigma(\hat{x}_{ui} - \hat{x}_{uj}))}_{=:\Delta_{u,i,j}} \frac{\partial}{\partial \theta} (\hat{x}_{ui} - \hat{x}_{uj})) \quad (4.42)$$

In Equation 4.42, α is the learning rate. In SGD the learning rate is normally selected as a small number at the initiation step to ensure the update in each step moves the parameter in the right direction. Each gradient step have a multiplicative scalar to update parameters being:

$$\Delta_{u,i,j} := (1 - \sigma(\hat{x}_{ui} - \hat{x}_{uj})) = (1 - p(i \succ_u j)) \quad (4.43)$$

$\Delta_{u,i,j}$ shows how much influence a selected pair of positive and negative item for each user (u, i, j) has on the update of the $\theta \in \Theta$ parameters. The greater $\Delta_{u,i,j}$ the larger the learning rate for the θ parameters. Rendle et al. [18] refer to the $\Delta_{u,i,j}$ as the “Gradient Magnitude” of a sample (u, i, j) .

Gradient magnitude (i.e. $\Delta_{u,i,j}$) is close to 0 if item i is correctly estimated to have larger score than item j and is close to 1 if j is falsely assigned a higher score than i . When this value is close to 0 nothing would be learned from the sample (u, i, j) as its gradient vanishes and the θ is not changed while in update step.

For the datasets with large number of items in the item set $|I|$, the random selection of negative items (i.e. $j \in I \setminus I^+(u)$) increase the chance of having non informative samples [18]. The non informative samples are the items that do not carry much information in terms of gradient magnitude to update the θ parameters. Non informative samples lower the accuracy of the algorithm, as no learning takes place in a training epoch and the gradient descent algorithm might get trapped into a local optima, which yields a lower accuracy. To find the samples that all contribute to the learning process, one solution may be a search for a decision boundary [33] separating the positive and negative items for a given user. As the number of items increases, detecting this boundary becomes a harder problem to solve exactly. The problem can be tackled by an efficient approximate method. This idea is the basis of our proposed approach that is explicitly explained in the following section.

4.5 Proposed Similarity based Monte Carlo Bayesian Personalized Ranking (SMC-BPR) approach

In this section, we explain how our proposed approach tackles the problem of searching for decision boundary between the positive and negative items in the Bayesian Personalized Ranking (BPR) approach. Our proposed approach, the Similarity based Monte Carlo-BPR (SMC-BPR) provides a framework to find

an optimal decision boundary between the positive and negative items and approximate the posterior probability in BPR. To test our proposed approach we used factorization based model on four datasets from different recommendation domains.

The main motivation of our proposed approach is to overcome a limitation of the original BPR algorithm. More specifically, a general pairwise comparison framework suggests the comparison of all items against each other, which is computationally expensive, specially when the item space is too large. The BPR suggests bootstrap sampling as an approximate solution to reduce the computational complexity [1]. As the number of items increase the search space becomes larger and finding the right sample that contributes to the learning process becomes hard. This causes the Gradient Descent algorithm to settle at a local optima and do not move forward. Therefore, the parameters of the model would not be updated to represent the users and items and hence the probability of assigning positive item i s as having higher rank than negative item j s becomes smaller. This yields a low performance for the general algorithm in terms of accuracy of the predicted ranked list of items. Our proposed method, provides a framework to detect the most informative samples for the Gradient Descent algorithm to find the optima.

Essentially, in the original BPR, negative items are sampled to be compared against a positive items for each user. This negative sample j for each user-positive item pair (u, i) , is derived from $p(j|u)$ distribution (here we drop i from the $p(j|(u, i))$ for simplicity). In the original BPR, bootstrap uniform sampling is suggested to approximate the complete pairwise comparison ordering. In the case of uniform sampling of j for each (u, i) pair, $p(j|u) \propto 1$, where each negative item j has the same probability of being selected for a given user-item pair (u, i) . However, as it was also previously stated in [18] and [74], not all negative items are informative for BPR learning model. This means that the uniform distribution might not provide a good approximation for BPR optimization function. A more informative sampling strategy might help to improve the approximation that yields to a better overall performance of the RS algorithm.

As explained in the previous section, negative samples that generate higher gradient magnitude are the most informative ones. Revisiting the formulation of the gradient magnitude, $\Delta_{u,i,j} := 1 - \sigma(\hat{x}_{ui} - \hat{x}_{uj})$ shows that an informative negative sample j should generate a small sigmoid function $\sigma(\hat{x}_{ui} - \hat{x}_{uj})$ so that the gradient magnitude is at its maximum value. A sigmoid function as shown in Figure 4.3 goes to zero as x decreases to a negative value.

FIGURE 4.3: Illustration of Sigmoid function. This function goes to its minimum, being zero, as x decreases to negative values.

Based on the gradient magnitude function, the sigmoid function is as shown in Equation 4.44

$$\sigma(\hat{x}_{ui} - \hat{x}_{uj}) = \frac{1}{1 + e^{(\hat{x}_{ui} - \hat{x}_{uj})}}, \quad (4.44)$$

For this function to be at its minimum value we would have:

$$(\hat{x}_{ui} - \hat{x}_{uj}) \leq 0, \quad (4.45)$$

which is similar to have:

$$\hat{x}_{ui} \leq \hat{x}_{uj}, \quad (4.46)$$

The ultimate goal of BPR is to predict and assign higher scores to i s compared with j s. This means that the optimization function works towards maximising the probability of assigning i s over j s. Based on this modeling framework, the prediction of $(\hat{x}_{ui} - \hat{x}_{uj})$ should never be negative. This is contradictory to what we state in Equation 4.46. The only scenario where this inequality might be true is when:

$$\hat{x}_{ui} = \hat{x}_{uj}, \quad (4.47)$$

Consider the scoring function is Matrix Factorization (see Section 4.3). Based on Equation 4.31:

$$q_i^T \cdot p_u = q_j^T \cdot p_u, \quad (4.48)$$

This can be simplified into:

$$q_i = q_j \quad (4.49)$$

This means that by searching for similar positive and negative items, we can effectively find an optimal decision boundary between the positive and negative items. These item pairs then generate the predicted scores that have the minimum sigmoid function value that yields the maximum gradient magnitude. Following this process of finding the j s for a selected i increases the chance of learning from each samples and finding the optima with gradient descent algorithm. Hence, we propose the use of a similarity function based on the latent factor vectors that represent the items.

A similarity in a data mining context is usually described as a distance with dimensions representing features of the objects. If this distance is small, there will be a high degree of similarity; if a distance is large, there will be a low degree of similarity [111]. There are several measures of similarity in the literature. for

example, Euclidean distance measures the length of the path connecting two data points. This is the most common measure of distance for the situations when the data is dense or continuous. Another example is Manhattan distance that is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. Cosine similarity metric, on the other hand, finds the normalised dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0 is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors. Usually, the measure of similarity computed in the reduced dimensional space is the cosine between vectors [112], as this measure tends to work well, and there are some weak theoretical grounds for preferring it [113].

Therefore, the selected similarity measure for our proposed approach is the cosine similarity between a positive item i and a negative item j . The Cosine Similarity of item i and j is measures based on their latent factor vector that is learned using the scoring function (e.g.MF model):

$$Cosine(i, j) = 1 - \frac{i \cdot j}{\|i\|_2 \|j\|_2} \quad (4.50)$$

We propose a novel sampling approach for the negative items. We generate a non uniform distribution based on the similarity of items. The distribution comes from the ranked similar items. Specifically, in this method, we first compute the similarity of all items against each other. We then rank the negative items based on their relative similarities with the selected item i . The non uniform distribution is:

$$p(j|u) \propto \exp(-r(j|u)/\lambda), \quad \lambda \in \mathbb{R}^+. \quad (4.51)$$

The sampler for negative item j given the (u, i) pair, would first randomly sample for r from an exponential distribution. r is the ranked position of an item based on its cosine similarity value with the selected positive item i . Item j is then selected as the item that is located at rank r for user u . This is equivalent to $j = r^{-1}(r|u)$. At this stage we check whether the selected item j is actually a negative item for user u . In case it is not a negative item, another r is selected and the next $j = r^{-1}(r|u)$ is picked until the sampler finds an actual negative item for the selected (u, i) pair.

The ranked similarity value should be updated at each learning iteration as the latent factor vector for the items is learned and updated. We consider ranked similarity instead of the pure similarity value since the rank is an absolute value and it is not respective to the actual item similarity score.

The proposed sampling solution is summarised in the following steps:

1. Randomly select a user and positive item pair (u, i)
2. Compute the similarity of all items with i
3. Sort the items based on their similarity value with i
4. Randomly select r from an exponential distribution
5. Find $j = r^{-1}(r|i)$
6. Check if j is a negative item for user u
7. If condition in step 6 is not satisfied, repeat from step 4 until finding a j that is an actual negative item for user u

The pseudo code for this algorithm is provided in Algorithm2.

The proposed sampling strategy is both adaptive and user specific. It is adaptive as it is updated during the learning process. This is because the similarity values are dependent on the Θ parameters that is learned with the scoring function. It is also user specific, as the negative items are selected for a unique user at each stage. The results from this algorithm provides a higher accuracy than the original BPR algorithm. However, the computation of similarity measures for all available item pairs at each epoch is time consuming. Therefore, we propose the computation of

Algorithm 2 Pseudo-code for the proposed SB-BPR.

```

1: procedure LEARNSIMILARITYBASEDOVERSAMPLING( $\alpha, D_S$ )
2:   Randomly initialize  $\Theta$ ,  $q=0$ 
3:   repeat
4:      $q \leftarrow q + 1$ 
5:     if  $q = 1$  OR 5 OR 10 OR 15 OR 20 then
6:       for  $j \in \{1, \dots, |I|\}$  do
7:         Compute  $Cosine(i, j)$ 
8:         Compute  $r(j|i)$ 
9:       end for
10:    end if
11:    Draw  $(u, i) \in D_S$  uniformly
12:    Draw  $r$  from  $p(r) \propto \exp(-r/\lambda)$ 
13:     $j \leftarrow r^{-1}(r|(u, i))$ 
14:    for  $\theta$  in  $\Theta$  do
15:      Update  $\theta$ 
16:    end for
17:  until convergence
18:  return  $\Theta$ 
19: end procedure

```

similarity and the update of the non uniform distribution for the selection of j s (i.e. $p(j|u)$) to occur at every 5 iteration until iteration 20. Line 5 in algorithm 2 is reflective of this strategy. The numbers 5 and 20 are arbitrary numbers that are selected based on the experiment results for one of our datasets. More precisely, the decision on when the updates should occur is completely dependent on the dataset and the design of the experiments.

In order to mitigate this problem and to overcome the computational cost of computing all pair's similarity values, we propose an approximate solution following the Monte Carlo Markov Chain methods. We propose the use of a Metropolis algorithm to select a small number of negative items to compute their similarity with the selected positive item. This will provide an informative approximation of the distribution, where the negative sample should be derived from. The Metropolis algorithm is selected over the simpler rejection or importance sampling, as it provides an iterative process to better approximate the proposal distribution and select the most informative samples from that distribution [24]. The proposed approach is referred to as Similarity based Monte Carlo BPR (SMC-BPR).

The SMC-BPR approach finds the most similar negative items to the selected positive items, using Markov Chain Monte Carlo sampling approaches. The MCMC sampling as explained in 4.1.4, randomly selects a sample, compares it with a former sample against a set criteria. If the current sample is better than the former in terms of the criteria, then the current sample is kept as the new sample. If it is not a better sample, then the former sample is considered as the new sample. The process is repeated for a set number of times. In our proposed approach, the comparison criteria is considered to be the cosine similarity of positive item i with the selected sample.

The SMC-BPR runs as follows. It first randomly selects the user-positive item pairs (u, i) . Then the negative item j is selected using the Metropolis algorithm, that is a type of Monte Carlo Markov Chain (MCMC) algorithms. The transition probability distribution for our proposed model is uniform distribution. This means that each item is selected randomly for the Metropolis evaluation. Therefore, we have:

$$q(j^{(\tau)}|j^*) = q(j^*|j^{(\tau)}) \quad (4.52)$$

from Equation 4.10. The target function $\tilde{p}(j)$, in our proposed approach is the cosine similarity of the negative item j and the positive item i . Since the transition probability is symmetric the acceptance probability is derived from the following function 4.53:

$$A(j^{(\tau)}, j^*) = \min(1, \frac{\tilde{p}(j^{(\tau)})}{\tilde{p}(j^*)}) \quad (4.53)$$

This process is summarised in the following steps:

1. Randomly select a user and positive item pair (u, i)
2. Filter the item lists for u to have all negative items

Metropolis starts here:

3. Starting from $\tau = 1$, pick an item $j^{(\tau)}$ from the list of negative items for u

4. Check $j^{(\tau)}$'s similarity with i
5. Pick an item j^* from the list of negative items for u
6. Check if j^* 's similarity with i is higher than the $j^{(\tau)}$
7. If condition in step 6 is satisfied, replace $j^{(\tau+1)}$ with j^* and if not assign $j^{(\tau+1)} = j^{(\tau)}$ and repeat from step 4 until the maximum number of τ

The pseudo code for the SMC-BPR approach is provided in Algorithm 3

Algorithm 3 Pseudo-code for the proposed SMC-BPR.

```

1: procedure LEARNSIMILARITYBASEDMONTECARLOOVERSAMPLING( $\alpha, D_S$ )
2:   Randomly initialize  $\Theta, \tau=[1:50]$ 
3:   repeat
4:     Draw  $(u, i) \in D_S$  uniformly
5:     Draw  $j^{(\tau)}$  from  $(I \setminus I_u^+ \in D_S)$ 
6:     Compute  $Cosine(i, j^{(\tau)})$ 
7:     for  $\tau \in \{1, \dots, 50\}$  do
8:       Draw  $j^*$  from  $(I \setminus I_u^+ \in D_S)$ 
9:       Compute  $Cosine(i, j^*)$ 
10:      if  $Cosine(i, j^*) > Cosine(i, j^{(\tau)})$  then
11:        Set  $j^{(\tau+1)} = j^*$ 
12:      Else
13:        Set  $j^{(\tau+1)} = j^{(\tau)}$ 
14:      end if
15:    end for
16:     $j \leftarrow j^{(\tau)}$ 
17:    for  $\theta$  in  $\Theta$  do
18:      Update  $\theta$ 
19:    end for
20:  until convergence
21:  return  $\Theta$ 
22: end procedure

```

In our experiments we observed that with the Metropolis solution the system can be updated at each epoch without increasing the computational cost of the similarity evaluation of all available item pairs. In our experiments the Metropolis algorithm converges fast and hence the maximum value for τ can be kept at 50 as we noticed during our experiments.

We compared the performance of our algorithm in terms of the accuracy of the recommended list of items with the original BPR as well as an extension of BPR

that addresses the sample negative sampling problem in the original BPR [18]. In the next Section, we provide an overview of the alternative sampling strategies presented in [18].

4.6 Alternative Strategies

Rendel et al. [18], proposed non-uniform sampling strategy for BPR and provided evidence that this model converges faster as the selected sample by this strategy is more informative. They refer to their sampling method as **Adaptive and User specific Sampling** strategy. Let's consider a static user specific item popularity distribution:

$$p(j|u) \propto |\{(u', j') \in S : u = u', j = j'\}| = \delta((u, j) \in S) \quad (4.54)$$

This simple sampling approach that is based on user specific item popularity is not adaptive to the model parameters. Also, since there are only a limited number of items that are selected more than once by one single user, the approach provides a small subset of items for each specific user. Therefore, the authors propose a scoring function to find a negative item j for a sample $(u, i) \in S$, in a way that the relative score for j is large and close to the score for i . This is similar to selecting a j to increase the gradient magnitude $\Delta_{u,i,j}$. finding a j with a high relative score (i.e. $\hat{x}(u, j)$) is similar to finding a j with a low rank (i.e. $\hat{r}(u, j)$). The advantage of using rank over score is that the score is relative to other items, while the rank is an absolute value. Therefore, the authors in [18] used the ranking function to formulate the distribution of negative items (Equation 4.55).

$$p(j|u) \propto \exp(-\hat{r}(j|u)/\lambda), \quad \lambda \in \mathbb{R}^+. \quad (4.55)$$

The sampler for negative item j given the (u, i) pair would be to first randomly sample for r and then find the j that is located in the rank r for user u . This is

equivalent to $j = \hat{r}^{-1}(r|u)$. A trivial implementation of the second step would be to compute the score $\hat{x}(j|u)$ for all j and sort them to get their respective rank $\hat{r}(j|u)$ and then find the j in the rank r . However, since the item space is too large this would take time to compute exactly. Therefore, the authors proposed an efficient sampling algorithm based on Matrix Factorization [18]. They present a general MF model to generate the scores. In order to provide a fast adaptive and user specific sampler they provide a normalized version of the MF model that generates a rank mixture to approximate the original ranking function in Equation 4.55.

The scoring model using MF is in Equation 4.56:

$$\hat{x}(j|u) := \sum_{f=1}^k v_{u,f} v_{j,f}, \quad V \in \mathbb{R}^{(C \cup I) \times k} \quad (4.56)$$

Then the mixing probability is derived from a normalized version of the MF scoring function the normalized version of this function is as follows:

$$\hat{x}^*(j|u) := \sum_{f=1}^k p(f|u) \operatorname{sgn}(v_{u,f}) v_{j,f}^* \quad (4.57)$$

In this Equation 4.57 $p(f|u)$ is the probability function proportional to:

$$p(f|u) \propto |v_{u,f}| \sigma_f \quad (4.58)$$

and $v_{j,f}^*$ is the standardized item factors:

$$v_{j,f}^* = \frac{v_{j,f} - \mu_f}{\sigma_f} \quad (4.59)$$

In Equation 4.59, μ_f is the empirical mean over all features and σ_f is derived from the variance over all item factors. In $\hat{x}^*$, the $p(f|u)$ is the mixing probability over standardized item factors. The Equation 4.57 shows that the larger $p(f|u)$

the more important is dimension f for the specific user u . Hence, the sampling distribution can be shown as the mixture:

$$p(j|u) := \sum_{f=1}^k p(f|u)p(j|u, f) \quad (4.60)$$

Since the item factors are standardized and transformed into $v_{j,f}^*$ the sampling distribution for j outlined in Equation 4.55 can be re-written as Equation 4.61:

$$p(j|u, f) \propto \exp(-\hat{r}^*(j|u, f)/\lambda), \quad (4.61)$$

The transformed ranking function $\hat{r}^*(j|u, f)$ can be derived from the scoring function $\hat{x}^*(j|u, f)$ that can be computed as follows for each negative item j and user u :

$$\hat{x}^*(j|u, f) := \text{sgn}(v_{u,f}) v_{j,f}^*. \quad (4.62)$$

We can replace the original item factors by the standardized factors and arrive at Equation 4.68 for the scoring function:

$$\hat{x}(j|u, f) := \text{sgn}(v_{u,f}) v_{j,f}. \quad (4.63)$$

The authors outline the process sampling as follows [18]: (1) sample a rank r from a geometric distribution. They later substitute the geometric distribution by exponential distribution of $r \sim \lfloor \text{Exp}(1/\lambda) \rfloor$. (2) sample factor f from the probability distribution $p(f|u)$ using Equation 4.58. (3) Sort items according to their factor f . (4) Return item j on rank r from the top if the $\text{sgn}(v_{u,f}) = 1$ or rank r from the bottom if the $\text{sgn}(v_{u,f}) = -1$.

To further reduce the computational complexity of the sampling the authors propose the pre-computation of the inverse ranking of items based on dimension f for each of the k factors. They also indicate that since the ranking does not change

always during the learning we can only compute the ranking every $|I| \log |I|$ iterations. This value is normally large and increases as the item space gets larger. Therefore, the updates in the non uniform distribution normally do not occurs at any later iterations. In this case, this solution might not differ from the bootstrap sampling that randomly selects the negative items. In that, this approach does not provide informative samples to update the learning model and the gradient descent algorithm might settle at a local optima. This then yields to a poor prediction of the assignment of positive items over negative ones and hence a poor accuracy of the predicted ranked list of items.

The pseudo code for BPR using adaptive sampling strategy is also provided by the authors in [18] (Algorithm 4).

Algorithm 4 Pseudocode for BPR with adaptive user specific oversampling of negative items for matrix factorization to improve learning convergence.[18]

```

1: procedure LEARNADAPRIVEOVERSAMPLING( $\alpha, S$ )
2:   Randomly initialize  $\Theta$ ,  $q=0$ 
3:   repeat
4:      $q \leftarrow q + 1$ 
5:     if  $q \% |I| \log |I| = 0$  then
6:       for  $f \in \{1, \dots, k\}$  do
7:         Compute  $\hat{r}^{-1}(\cdot|f)$ 
8:         Compute  $\sigma_f^2$  and  $\mu_f$ 
9:       end for
10:    end if
11:    Draw  $(u, i) \in S$  uniformly
12:    Draw  $r$  from  $p(r) \propto \exp(-r/\lambda)$ 
13:    Draw  $f$  from  $p(f|u) \propto |v_{u,f}| \sigma_f$ 
14:     $j \leftarrow \begin{cases} r^{-1}(r|f), & \text{if } \text{sgn}(v_{u,f}) = 1 \\ r^{-1}(|I| - r + 1|f), & \text{else} \end{cases}$ 
15:    for  $\theta$  in  $\Theta$  do
16:      Update  $\theta$ 
17:    end for
18:  until convergence
19:  return  $\Theta$ 
20: end procedure

```

In order to evaluate the performance of our proposed approach we used an evaluation metric from a common recommender system evaluation metrics described in the following section. We performed a set of analysis on four datasets from

different recommendation tasks. The description of these datasets is provided in the proceeding section.

4.7 Recommender System Evaluation

Evaluating Recommender system performance is one of the ongoing challenges in the field [3]. Some focus on the accuracy measures, while others focus on user satisfactions that goes beyond accuracy of predictions and Learning to Rank [114]. Generally, there are three main types of evaluation approaches: *Offline evaluation*, *User study* and *Online evaluation* [114]. The *Offline evaluation* is the simplest and cheapest approach and therefore many algorithms and systems can be tested on the application in hand within a short amount of time. The result from this evaluation is normally a score or a quantitative value that can easily be compared and communicated within parties from different backgrounds. The metrics in this category normally focus on rating prediction or ranking accuracy [114]. In *user study*, a group of users evaluate the RS based on different qualitative and quantitative measures. The success of the system is measured by certain factors such as accuracy, serendipity, novelty and etc. The assumption is that the selected group are representative of the actual users of the application. *Online evaluation* is normally the most time consuming and sensitive approach, where the evaluation takes place on the implemented RS in the actual application. In this case, the RS is evaluated based on business success measures. For example, in an e-commerce application the success of a RS providing product recommendations, can be evaluated based on the business profits (i.e. the number of sold items).

Some researchers have argued that the bottom-line measure of recommender system success should be user satisfaction [115]. However, the offline evaluations is normally performed to filter out a number of alternative approaches. A typical use of offline experiments is when the parameters of the algorithms are tuned in the offline experiment, and then the algorithm with the best tuned parameters continues to the next phase [114].

In offline evaluation, the algorithms are tested on pre-collected datasets from the actual interaction of the users with the application. The interaction is either recorded as ratings (i.e. Explicit information) or selection, purchase or clicks (i.e. implicit information). The assumption for this evaluation approach is that user behaviour and their interaction with the system would be similar to the time of gathering data [114]. There are certain protocols that should be followed while selecting the training and testing samples from a pre-collected dataset. This is to simulate user behaviour to be as close to the reality as possible [2].

There are several evaluation metrics to evaluate the performance of a recommender system in an offline manner [2, 115]. The commonly used evaluation metrics measure the performance of machine learning algorithms used in RS and can be categorised in three groups: (1) Predictive Accuracy metrics, (2) Classification Accuracy metrics, and (3) Ranking Accuracy metrics [115].

(1) **Predictive Accuracy metrics.** The most common evaluation metric for accuracy is Root Mean Square Error (RMSE). We used this measure to set the hyperparameters of our algorithm. RMSE for a validation set of T observations can be calculated as follows:

$$RMSE = \sqrt{\frac{1}{T} \sum_{(u,i) \in T} (\hat{r}_{ui} - r_{ui})^2}, \quad (4.64)$$

where $\hat{r}_{ui}$ is the predicted rating on item i given by user u and r_{ui} is the actual rating for item i by user u .

(2) **Classification Accuracy metrics.** The classification accuracy metrics measure the quality of the set of recommendations [16]. A common metric of classification accuracy is Precision and Recall [16]. The precision is commonly referred to the ratio of the relevant recommended items and the total number of recommended items; and recall is the ratio of the relevant recommended items and the total number of relevant items. Formally, precision and recall are computed as

follows:

$$\begin{aligned} Precision &= \frac{TP}{TP + FP}, \\ Recall &= \frac{TP}{TP + FN}. \end{aligned} \tag{4.65}$$

where TP is the number of relevant items for a user, FP is the number of non relevant items for that user, and FN is the number of relevant items recommended that are not recommended to a user. The final result can be presented as the average of precision and recall for all users in the test sample.

(3) **Ranking Accuracy metrics.** To evaluate the quality of the list of recommendations, ranking metrics are used. One of the most commonly used ranking accuracy metric is Mean Average Precision (MAP). MAP as the name implies is the mean of the Average Precision (AP) for all users in the dataset. The Average Precision also explains how well a list of provided items for a single user. The measure gives higher importance to the relevant items on top of the list and penalises for the relevant items that appears towards the end of the list. In Equation 4.66 and 4.67 the n is total number of items, N is the number of relevant items and Q is total number of users. In Equation 4.67 the $P(k)$ is the precision at cut off k and the $rel(k)$ is an indicator function equalling 1 if the item at rank k is a relevant document and zero otherwise.

$$MAP = \frac{\sum_{q=1}^Q AP(q)}{Q}, \tag{4.66}$$

$$AP = \frac{\sum_{k=1}^n (P(k) \times rel(k))}{N}. \tag{4.67}$$

The Area Under the ROC Curve is also categorised in this type of accuracy metrics. This measure is based on the percentage of correctly ranked items for each user. As Huang et al. [116] explains, AUC is equivalent to the probability that a randomly selected irrelevant item will have smaller estimated probability of belonging to the relevant items class than a randomly chosen relevant item. Hand and Till [117]

provide the following simple approach to compute the AUC:

$$AUC = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1}. \quad (4.68)$$

where n_0 and n_1 are the numbers of relevant and not relevant items, respectively. $S_0 = \sum r_i$, where r_i is the rank of relevant item i in the ranked list.

In this work, we evaluate our proposed algorithm in an offline manner. We use four pre-collected datasets from different applications. We divided the dataset into training and testing based on certain protocols that is explained in the experiment chapter (see Chapter 5). Our model is optimised based on the AUC measure and the final results is presented in terms of Mean Average Precision Measure at every training epoch for each dataset. The description of the datasets used in our experiments is provided in the following section.

4.8 Datasets

In this research, we used four datasets, from different recommendation application tasks, to test and compare our proposed model with the two baseline models. The datasets used in this study are publicly available except one that is a unique dataset extracted by our team during our collaboration with IBM. This dataset is an obfuscated user interaction data with IBM Watson Analytics (WA) application [118]. The three other datasets are from movie recommendation [119], mobile application recommendation [120] and joke recommendation application [65].

The models using binary relevance metrics for optimization, such as the algorithms used in this study, can not be used on explicit data, where the rating levels are provided. On the other hand, the datasets with binary relevance information to test these models are scarce. One solution for these models to work on rating scale data is to transform the graded relevance (i.e. rating scales) into binary preferences [30]. This transformation can be done by setting some thresholds on the scales. For example, considering anything beyond 4 as being preferred

(positive) and anything below 4 as not preferred (negative). This transformation causes the loss of information and therefore, the techniques using binary relevance optimization functions are not preferred for this type of information. However, these models normally perform well on the applications with binary relevance type of information [30]. Since datasets with implicit information are limited, we transformed some well studied public rating based datasets to a binary relevance type datasets as we wanted to validate our proposed model on multiple real life applications.

We performed another type of transformation, where we consider the recommendation task to be a binary relevance recommendation problem. For instance, for one of the datasets, Movielens 100K¹, that has rating information, we changed the recommendation task from finding user's next favourable movie, to finding the movie that the user will rate next. Specifically, we consider the availability of a use-item pair in the dataset as a positive preference (the user selected this movie to rate) and the unavailability of the user-item pair as negative preference (the user did not select that movie to rate). For the jester dataset² we also have user ratings on jokes. We consider the availability of rating to be a positive user preference towards the joke. The task is now to find the next joke the user might select to rate. In Frappe dataset³ the frequency of user interaction with the mobile application is provided. We did not consider the frequency of usage and just considered one interaction with the application as a positive user feedback. The task here is the next application the user might interact with.

The description of the datasets used in this study is summarised in Table 4.1 and a detailed explanation is provided in this section.

All of these dataset suffer from the sparsity problem that is a common problem in Recommender System (RS) framework [2]. This is because the users are only able to provide their feedback to a handful of items. Also, some not so popular items are left with very limited number of feedback from users. The overall sparsity

¹<https://grouplens.org/datasets/movielens/>

²<http://eigentaste.berkeley.edu/dataset/>

³<http://baltrunas.info/research-menu/frappe>

TABLE 4.1: Description of the datasets used in this study. IBM WA is the unique dataset from obfuscated user interaction with IBM Watson Analytics data visualisation recommendation application. ML-100K is the movie rating dataset made available by Grouplens. Frappe dataset is user mobile application interaction dataset and jester is a joke rating dataset

Dataset Title	Number of Observations	Number of Users	Number of Items
IBM WA	4,187	616	2,692
ML-100K	100,000	943	1,682
Frappe	96,202	957	4,082
Jester	1,700,000	59,132	150

of each dataset is summarised in Table 4.2. The sparsity rate is computed using Equation 4.69.

$$\text{Sparsity Rate} = \frac{\text{Number of observations}}{\text{Maximum Cardinality}} \quad (4.69)$$

TABLE 4.2: Sparsity Rate for each dataset

Dataset Title	Sparsity Rate
IBM WA	0.25×10^{-2}
ML-100K	0.63×10^{-1}
Frappe	0.24×10^{-1}
Jester	0.19

The accuracy of the recommendation system decreases as the data sparsity increases [121]. Table 4.2 shows that the first three dataset are very sparse and hence we do not expect a very high accuracy value for these datasets. For the last data though, we expect a higher accuracy of prediction as the sparsity for this dataset is approximately 0.19 that is a low sparsity level, which is a rare case in Recommender System domain. The details of the datasets used in this research along with their sparsity graphs based on items' popularity is provided below.

IBM Watson Analytics (WA): IBM Watson Analytics is a data visualisation recommendation application ¹. The users upload their dataset to the application and the system provides them with visualisation recommendations based on the data headers. The user may receive the recommendations on a dataset by either clicking on the dataset or by submitting a specific question regarding the data

¹<https://www.ibm.com/ca-en/marketplace/watson-analytics>

in natural language. Figure 4.4 provides an illustrative example of the current recommendations provided by WA. Terminologies used throughout this research for WA dataset are highlighted in this figure. In this research, we focus on the recommendations provided at the “starting point”. This is referred to the recommendations provided to the user before they ask any question and on the first page they visit as soon as they click on their dataset. There are always ten recommendations from position 0 to 9 at the starting point. Each recommendation includes “Data field(s)” that are the headers of the provided datasets. Recommendations also contain one of the 11 possible “visualisation types” that are appropriate to visualise the provided/selected data field(s).

FIGURE 4.4: Illustrative example of IBM Watson Analytics starting point recommendations

In a recommender system framework, we deal with the triplets of $\langle \text{user, item, preference} \rangle$ and the recommendations are provided through prediction of the item ranking based on past preferences of users.

The current recommendation for this application is provided based on a set of rules. We refer to the current recommendation engine a “rule based recommendation system”, rather than a “learning-based recommender system”. A learning based recommender system is an algorithm that *“learns from experience E , with respect to some class of task T with a performance measure P , if its performance at task T measured by P improves by learning the experience E ”* Mitchell et al. [122,

p.2]. For WA, the task can be the recommendation of data fields along with the appropriate visualisation type; the Experience can be derived from the user interaction with the system, which provides user preferences through selection and not selection actions; the performance of the system can be measured with two approaches: either by the commonly used performance metrics in machine learning or through user studies and evaluation of user satisfaction [81]. We evaluate our proposed model in an offline manner, using the common performance measures for recommender systems, such as Mean Average Precision (MAP) [114].

We define the items in WA to be characterised as the combination of meaningful visualisation types along with the data fields. This means that an item is a tuple of $\langle \text{data field(s)}, \text{visualisation type} \rangle$. In this research, we only focused on the items that were previously provided to the users and were either selected or not selected by users. For further information regarding the initial recommender system design for this application can be found at our earlier publication [81], that is also available as Appendix ??.

IBM Watson Analytics maintains a logging database based on ElasticSearch. Kibana which is an open source data visualization API for ElasticSearch is used to query the ElasticSearch engine, viewing logs and exploring events in real time. However, specific user actions and all Watson-generated recommendations are not available in this interface. For this reason, a data-extraction application was developed in Java to query complete logs from the Logstash servers. These logs contain all user actions and recommendations generated by Watson Analytics. Information about all the other events that were triggered when the user was interacting with the system is also stored in the logs. To extract specific item selection actions and recommended visualisations, we had to parse the extracted JSON logs and filter unnecessary information. Two Python scripts that parse over a month long data is developed for this purpose. The physical size of the unfiltered data is 40 Gigabytes. This dataset was extracted from the users interacting with Watson Analytics application-version1 during the month of October 2015 [118].

FIGURE 4.5: Overview of datasets extracted from IBM Watson Analytics Logstash

As illustrated in Figure 4.5, one database was created based on all the recommendations that were provided to the users, which we refer to as “Questions dataset”. A second database was created based on the selection actions, which we call “Selected dataset” in Figure 4.5. This includes the specifications of the selected recommendations by each user. We also had access to some high-level user information, such as their account type and email address providers, that was stored in the “User dataset”. All data bases were merged to a large database (i.e.CSV file) for further analysis.

The final dataset contains 716,945 observations and 17 variables. Each observation corresponds to a recommendation generated by Watson Analytics application. A subset of the variables defines the characteristics of the visualisation such as the type of the visualisation (e.g. bar chart), the caption of the visualisation (e.g. what is the breakdown of sales by year), the list of attributes (i.e. data fields) corresponding to the visualisation (e.g. sales or year) and the position of the visualisation in the list of recommendations. The selection action is stored as a Boolean variable that specifies if the visualisation was clicked by the user or not. The other subset of the variables stores information about the user. This

information is obfuscated in a way that it is not possible to decode and trace back to the users.

The final dataset used in our analysis for BPR contains the user IDs and item IDs for the items that were selected by the users. For an active user (i.e. the user we would like to generate a recommendation for), the items that are selected by other users are considered as negative items. The sparsity of this dataset in terms of the items' popularity is illustrated in Figure 4.8.

FIGURE 4.6: a

FIGURE 4.7: b

FIGURE 4.8: Item popularity (sparsity) for WA dataset. Figure (a) shows the probability of selection for the items in the WA dataset. Figure (b) shows the log of the selection frequency for each item in the dataset

Figure 4.8 shows the sparsity of the WA datasets with two illustrations. Figure 4.6 shows the probability of the selection of an item based on the item's historical selection frequency. As illustrated in this figure, many of the items were rarely selected and hence the information regarding those items is very scarce. This

yields to a poor modeling of those items and hence a poor prediction accuracy. Figure 4.7 shows the same frequency of selection in a logarithmic scale. The items with zero value on the selection frequency are selected only once in this dataset. This graph shows that most of the items in this dataset were selected only once by users.

ML-100K: Movie Lens 100K dataset contains explicit rating information provided by each user on each item. This dataset was collected through the MovieLens website (movielens.umn.edu) during the seven month period from September 19th, 1997 through April 22nd, 1998. This data was pre-processed, removing the users without complete demographic information [119]. The demographic information about the users includes age, gender, occupation and zipcode. Each item (movie) also accompanies with its characteristics. These characteristics include the movie title, release date and genre along with the IMDB URL. This Movielens dataset contains 100K observation from 943 users rating 1682 movies [119].

In this study, we do not use any of the available information for this dataset. In fact, the main problem that we address in this research is to rank items based on the minimum amount of information such as user’s binary preferences. In order to follow the BPR setting for pairwise comparison with implicit information, we considered the appearance of each user item pair as a positive source of information. Essentially, the ranking task in this application would be to find the next movie the user is going to rate. Therefore, the rated movie by a user is considered as positive feedback and the rest of the movies in the dataset not rated by the active user is considered as negative feedback. The sparsity of this dataset in terms of the items is illustrated in Figure 4.11

FIGURE 4.9: a

FIGURE 4.10: b

FIGURE 4.11: Item popularity (sparsity) for ML-100K dataset. Figure (a) shows the probability of rating each item in the ML-100K dataset. Figure (b) shows the log of the selection frequency for each item in the dataset

Figure 4.11 shows the item sparsity for ML-100K dataset with two illustrations. Figure 4.9 shows the probability that each item is rated by the users in the dataset based on the historical trend. As illustrated, this dataset is less sparse compared to the WA dataset. Hence, we expect a better overall recommendation accuracy

for this dataset compared with WA data. Figure 4.10 shows the frequency each item is rated by the users in the ML-100K dataset in a logarithmic scale. This graph also confirms that some of the items are being selected by the users less often than the others and hence those items (e.g. item 1588) might not be modelled as precise as the items with higher selection frequencies, such as item 50 in this dataset.

Frappe: This dataset is extracted from a mobile application recommendation platform for Google Play [120]. The recommender system algorithm behind frappe is a context aware recommender system algorithm, which provides recommendations that are suitable for a specific situation. This dataset is publicly available as the number of time a user interacted with an application on his/her mobile phone in various contextual information. The contextual information available in this dataset includes weather, country, day of the week, work or home, cost and time of the day.

For this dataset also, we only took a binary type of information. We considered each user-application interaction as a one time positive feedback. The applications that were never used by an active user and are listed in this dataset is considered as negative items for this user. The recommendation task in this case is the ranked list of applications the user will pick next. The dataset contains 96202 records by 957 users for 4082 apps. The sparsity of this dataset in terms of the items' popularity is illustrated in Figure 4.14

FIGURE 4.12: a

FIGURE 4.13: b

FIGURE 4.14: Item popularity (sparsity) for Frappe dataset. Figure (a) shows the probability of selection for the items in the Frappe dataset. Figure (b) shows the log of the selection frequency for each item in the dataset

Figure 4.14 shows the frequency that a mobile application is used by all users with two illustrations. This illustration considers the binary relevance data and does not use the frequency of usage information by each user from the original data. Figure 4.12 shows the probability of the selection of an app by all users based

on historical data. This dataset is the second most sparse datasets used in our experiments. As illustrated, the majority of items (i.e. mobile applications) are only used by one user or a handful amount of users in this dataset. The logarithmic illustration (Figure 4.13) shows this observation in a better visualisation scale. As shown in this figure, item number 16 is the most popular item in the dataset, while items such as item number 3539 is among the less popular ones, being used by a single user in the dataset.

Jester: This dataset consists of ratings provided by users on a set of jokes. The data was collected from Jester joke recommendation website [65]. This dataset is slightly different from the other datasets used in this study. First, the number of users in this dataset is very large compared with the number of users in the other datasets. This is while the number of items in this data is limited to 150. Also, the number of observations in this dataset is very large. Second, there are some items in the system that are rated by all users. Therefore, while transforming the dataset into a binary scale, some of the items can never be considered as negative items for any of the users. Hence, the proposed algorithm for this dataset is slightly different. For this data the proposed algorithm excludes those items from the Metropolis sampling. We outline the differences in more details in Section 5.4. The total number of users in this dataset is 59,132, who provided ratings to 150 jokes. The total number of user-item pair observations (positive feedback) is 1,700,000. The sparsity of this dataset in terms of the items' popularity is illustrated in Figure 4.17

FIGURE 4.15: a

FIGURE 4.16: b

FIGURE 4.17: Item popularity (sparsity) for Jester dataset. Figure (a) shows the probability of selection for the items in the Jester dataset. Figure (b) shows the log of the selection frequency for each item in the dataset

Figure 4.17 shows the frequency that a joke is rated by the users in this dataset. As shown in Figure 4.15, although there are still obvious popular items in this dataset, but most of the items received a considerable amount of feedback from the users. Therefore we expect the recommender system algorithm to be able to

model all items with a high level of accuracy. The selection frequency of the items with logarithmic scale (Figure 4.16) shows this with a better illustration scale.

Chapter 5

Experiments and Results

In this chapter we explain our experiment design that is based on the guidelines provided by Alpaydin [33]. The results of the analysis is summarised in Section 5.6.

5.1 Aim of the study

As described in the problem statement (Section 2), we aim to design a Recommender System algorithm using a single source implicit user feedback. Bayesian Personalized Ranking (BPR) approach is one of the most popular approaches to deal with implicit user feedback for recommender systems [1, 3, 18, 22, 31]. We conducted a replication study on BPR, and we observed a limitation in this approach. Our proposed algorithm in this work, is an extension to the BPR approach that addresses the observed limitation. This limitation comes from the approximate solution suggested in the original BPR for the posterior probability. In a pairwise comparison framework all items should be compared against each other, which causes a large number of computations, specially when the item space is large. The original BPR suggests bootstrap sampling as an approximate solution to reduce the number of computations. As the number of items increases the search space becomes larger and finding the right sample that contributes most to the

learning process becomes hard. Our proposed method, provides an approach to detect the most informative samples for the learning algorithm. The details of our proposed Similarity based Monte Carlo-BPR (SMC-BPR) approach is provided in Section 4.5.

5.2 Selection of the response variable

Recommender systems performance is measured using various types of evaluation metrics [114]. A thorough review of these metrics is provided in Section 4.7. In this work, we evaluate our proposed Recommender System algorithm in an offline manner and based on the accuracy of the recommended list of items.

In most recommender system applications, the system provides a list of items to a user typically in a vertical or horizontal manner, where a certain natural browsing order is presented. In these applications the accurate prediction of rating might not be the most important measure to evaluate the RS success; rather the correct ordering of items according to the users (also known as correct ranking of the list of items) is considered as a success factor. There are two approaches for measuring the accuracy of such a ranking: (1) determine the correct order of a set of items for each user and measure how close a system comes to this correct order. In this case we need the actual correct order of items for each user. (2) measure the utility of the systems ranking to a user [114]. In this work, the second approach is considered as the actual correct order of items ranking is not available for the users in our experiments.

Some of the most common utility base rank list evaluation measures are : (1) half-life utility [123], which assumes an exponential decrease in the interest of users as they move away from the recommendations at the top. (2) normalised discounted cumulative gain (nDCG), for which all of the top k documents are scored at a decaying discount factor [124]. nDCG can consider a non binary relevance items in the evaluation of the ranked list of items. (3) Mean reciprocal rank (MRR) is the inverse position of the first relevant document in the list of items and is

therefore commonly used in applications, which only the first result matters [125]. (4) Mean Average Precision is a measure to evaluate the binary relevance items [126]. MAP discount lower ranks more than nDCG, as nDCG has a heavier tail at high ranks. In this work we use Mean Average Precision (MAP) [126], as we deal with binary relevance items; also, we would like to assign higher importance to the items on top of the list of recommended items to highlight the accuracy of the recommended list to the users.

MAP evaluates if the users preferred items are located at the top ranked positions. It also penalises for the correct prediction of items that are included in the lower positions in the item list. Figure 5.1 provides an example of MAP@10 computation that is the MAP for a list of 10 recommendations for two users. The overall MAP@100 reported in this study is on all users in our datasets.

FIGURE 5.1: An example of Mean Average Precision@10 computation for two users

The Average Precision measures how accurate is the list of provided items for a single user. This measure gives higher importance to the relevant items on top of the list and penalises for the relevant items that appears towards the end of the list. In this study, we compute the MAP@100 using Equation 5.1. In Equation 5.2, N is the number of relevant items. The $P(k)$ is the precision at cut off k and the $rel(k)$ is an indicator function equalling 1 if the item at rank k is a relevant document and zero otherwise. Q in Equation 5.1 is the total number of users and q represents each user.

$$MAP@100 = \frac{\sum_{q=1}^Q AP@100(q)}{Q}, \quad (5.1)$$

$$AP@100 = \frac{\sum_{k=1}^{100} (P(k) \times rel(k))}{N}. \quad (5.2)$$

We measure this value at each epoch using the test set. The graphs provided in Section 5.6, shows the MAP@100 at each training epoch.

5.3 Choice of factors and levels

We compare the performance of our proposed approach with two other approaches. One is the original BPR approach that uses uniform bootstrap sampling to approximate the posterior probability (Section 4.2). The second approach is an extension to the BPR that addresses the approximate inference problem using a non-uniform adaptive sampling strategy (Section 4.6). We measure the performance of each algorithm based on Mean Average Precision at each training epoch on the same test instances. Our target is to find the algorithm that generates the highest accuracy based on the MAP measure.

5.4 Choice of experiment design

To evaluate the performance of our proposed method, we used four datasets from different recommendation tasks. Each dataset was pre-processed to follow the binary relevance measure (see Section 4.8). We then split the data into training and testing chunks for the final analysis. Each dataset has unique characteristics and therefore pre-processing for each was slightly different.

Every Machine Learning algorithm needs a good amount of data to learn from. During our pre-processing step, we selected the users with more than 20 positive actions for each dataset to have a good amount of data to understand the user

behaviour. This is a common practice in the studies using Bayesian Personalized Ranking [1, 18, 23, 88]. The IBM WA data is an exception, as it is a small dataset with limited number of user activities. For IBM WA dataset, we considered the minimum user activity to be 5. For the Jester dataset, due to the large number of observations we decided to select users with a minimum of 30 activities. The final number of observations, users and items after this filtering is reported in Table 5.1. This table reports the description of the data used in the final analysis.

TABLE 5.1: Description of the pre-processed datasets used in this study.

Dataset Title	Number of Observations	Number of Users	Number of Items
IBM WA	2,696	253	1,757
ML-100K	89,640	850	1,671
Frappe	93,342	487	3,808
Jester	1,249,950	18,160	140

In this research, we use 80 and 20 split for training and testing respectively, which is a common practice in machine learning experiments [33]. We also ensured that for each dataset we have a minimum number of user activities in the training set. This means that we randomly picked a minimum number of user activities for training set and set this part of the data aside. We then randomly picked the test instances from the remaining data. The remaining instances (not included in test set) are then combined with the set aside training set to form the full training set. The final training and test sets include 80 and 20 percent of the data respectively. The minimum user activities for training set for IBM WA datasets is 4, for ML-100K and Frappe is 10 and for Jester dataset this value is 20. The results reported in Section 5.6 are based on the 80-20 split following the minimum observation criteria for training set. The summary of minimum user activity for inclusion and minimum user activity in training is provided in Table 5.2.

TABLE 5.2: Summary of minimum user activity criterion for each dataset.

Dataset Title	Minimum User Activity for Inclusion	Minimum User Activity for Training set
IBM WA	5	4
ML-100K	20	10
Frappe	20	10
Jester	30	20

The hyper-parameters of the Matrix Factorization model in BPR (see 4.3) should be set at the initial step [110]. Two of these parameters are Learning Rate (LR) and Regularisation Rate (RR). These parameters should be set based on the characteristics of each dataset for an optimal accuracy. We performed an extensive search for the best parameters through multiple run of the experiments, which is a common practice in this field [109]. The summary of the learning rate and regularisation rate used for our reported results are listed in Table 5.3 for reproducibility purposes.

TABLE 5.3: Summary of Regularisation Rate and Learning Rate for each dataset.

Dataset Title	Regularisation Rate	Learning Rate
IBM WA	0.01	0.001
ML-100K	0.01	0.005
Frappe	0.0001	0.001
Jester	0.05	0.05

Another hyper-parameter that should be initialised in matrix factorization models is the number of latent features. This number also depends on the number of observations, number of users and number of items for each dataset [110]. It should be noted that we performed our experiments with 16, 32, 64 latent features and the results for all are approximately similar. We only report the results with one set of number of features for each dataset to avoid confusions. Table 5.4 summarises the number of features used for each dataset in the reported experiment results.

TABLE 5.4: Summary of the number of features for each dataset.

Dataset Title	Number of Features
IBM WA	16
ML-100K	64
Frappe	32
Jester	64

The next parameter that should be set at the initialisation step in BPR approach is the number of samples per training epoch. This parameter is also set according to the number of observations in each dataset. In both [18] and [1], the authors suggest the use of $10 * |D_S|$, that is 10 times the number of observations. We tried different variations from $2 * |D_S|$ to $20 * |D_S|$ and we report the results only for

the number of samples that are listed in Table 5.5. It should be noted that the accuracy of the results with different number of samples were slightly different. However, the overall trend in terms of the algorithm comparison was similar to the reported results.

TABLE 5.5: Summary of the number of samples per training epoch for each dataset.

Dataset Title	Number samples per Training epoch
IBM WA	10
ML-100K	10
Frappe	10
Jester	2

In our proposed algorithm Similarity based Monte Carlo -BPR (SMC-BPR), a Monte Carlo Markov Chain method is applied (see Section 4.5). The number of iterations for this method should also be set at the initial step. This number for our experiment depends on the number of items in the dataset. We performed our experiments with multiple iteration numbers from 20 to 200 for all datasets. We did not observe any significant changes in the final results depending on this parameter. Hence, we only report the results for one of these numbers outlined in Table 5.6 for each dataset.

TABLE 5.6: Summary of the number MCMC iterations for each dataset.

Dataset Title	Number MCMC iterations
IBM WA	50
ML-100K	50
Frappe	150
Jester	20

In order to reduce the computational cost, we do not perform the similarity based metropolis sampling for our proposed method at every training epoch. We propose the updates to be performed at every 5 epochs. This was based on our observation of the final results comparing the updates at every 2, 5, 10, 20 epochs. The optimal results were observed at the updates at every 5 epochs.

5.5 Performing the experiment

We performed 500+ experiments on each dataset based on different configurations of the model for each algorithm (i.e. BPR-MF, Adaptive-BPR-MF and SMC-BPR-MF). However, we report the comparison of the result of the algorithms on a fixed hyper-parameter settings for each dataset. We also observed for the monotonic decrease in the cost function for each experiment to observe the performance of the algorithms.

All codes for the SMC-BPR algorithm were developed by the author of this manuscript. The data preparation as well as statistical analysis of the data and the results are performed using R software. The algorithm was developed on top of theano-bpr library ¹ using python 2.7. The updated version of the code is available at our github repository ².

5.6 Analysis of the data and discussion of the results

In this section, we provide the summary of our results. Each plot shows the comparison of the accuracy of the three algorithms based on their performance on each dataset. The results are reported based on MAP@100 at each training epoch for the test instances. The respective hyper-parameter for the reported results and each dataset is summarised at Tables 5.3, 5.4, 5.5 and 5.6.

Figure 5.2, shows the performance of the three algorithms (i.e. BPR-MF, Adaptive-BPR-MF and SMC-BPR-MF) for WA dataset.

¹<https://github.com/bbc/theano-bpr>

²<https://github.com/RyersonU-DataScienceLab/parisa>

FIGURE 5.2: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for WA dataset in terms of MAP on each training epoch for test instances

As illustrated in Figure 5.2, The SMC-BPR-MF algorithm outperforms both the BPR-MF and the BPR-MF with adaptive sampling (i.e. AS-BPR). As was expected the performance for none of the algorithms shows a high accuracy in terms of MAP. This is because of the data sparsity that does not allow the Matrix Factorization algorithm to model the items accurately and hence the prediction accuracy is not high. However, the result is better compared with a randomly generated list without any learning model. Depending on the number of relevant items in each dataset the MAP for a random approach would be approximately zero. We also performed a t -test to evaluate the differences between the performance of the three algorithms. The result indicated that the proposed algorithm SMC-BPR-MF performs significantly better than the other two algorithms with p -value of $< 2.2 \times 10^{-16}$ for both baseline algorithms (see Table 5.7).

Similarly, in Figure 5.3, the performance of the three algorithm in terms of MAP at each training epoch for test instances is illustrated for the ML-100K dataset.

FIGURE 5.3: The comparison of the three algorithms, Blue: BPR, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for ML-100K dataset in terms of MAP on each training epoch for test instances

As shown in Figure 5.3, our proposed algorithm (i.e. SMC-BPR-MF) outperforms the two other baseline algorithms for this dataset too. The difference between the performance of the SMC-BPR-MF algorithm and the baselines for this dataset is also significant based on t -test, with p -value $< 2.2 \times 10^{-16}$, as reported in Table 5.7. For this dataset, we observe a better overall MAP value compared with the MAP for WA dataset. This is because ML dataset is less sparse compared with the WA dataset and the scoring model was able to make more accurate predictions for this dataset.

Similar graph is provided for the Frappe dataset in Figure 5.4. This figure, shows the performance of the three algorithms on the Frappe dataset.

FIGURE 5.4: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Frappe dataset in terms of MAP on each training epoch for test instances

As shown in Figure 5.4, The proposed algorithm SMC-BPR-MF outperforms the other two base lines for this dataset too. The difference is also significant with $p\text{-value} < 2.2 \times 10^{-16}$ based on $t\text{-test}$ for both baselines, as reported in Table 5.7. The overall MAP accuracy value for this dataset is also very similar to the WA dataset due the data sparsity as was expected.

The result of the analysis on the last dataset, the Jester joke recommendation, is provided in Figure 5.5.

FIGURE 5.5: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Jester dataset in terms of MAP on each training epoch for test instances

The graph in Figure 5.5, shows a similar result reported for the former three datasets. The proposed algorithm, SMC-BPR-MF, outperforms the two baseline algorithms in terms of MAP accuracy measure. This difference is also significant with p -value $< 2.2 \times 10^{-16}$ based on t -test, as reported in Table 5.7. The MAP result for this dataset shows around 70% accuracy in terms of MAP. As was reported in Section 4.8, this dataset has low sparsity and most of the items receives more than one user feedback. Therefore, the items can be modelled with a higher accuracy by the Matrix Factorization model. This results in higher performance accuracy for the ranked list of items.

TABLE 5.7: The p -value and t -value for the t -test analysis to show the significant difference in the performance of **SMC-BPR-MF** compared with the baselines BPR-MF and AS-BPR-MF in terms of MAP. This table shows the result for the reported results in this study.

Dataset Title	BPR		AS-BPR	
	p -value	t - value	p -value	t - value
IBM WA	$< 2.2 \times 10^{-16}$	17.97	$< 2.2 \times 10^{-16}$	65.54
ML-100K	$< 2.2 \times 10^{-16}$	67.93	$< 2.2 \times 10^{-16}$	34.19
Frappe	$< 2.2 \times 10^{-16}$	61.90	$< 2.2 \times 10^{-16}$	60.81
Jester	$< 2.2 \times 10^{-16}$	19.45	$< 2.2 \times 10^{-16}$	187.91

All reported results are for a specific combination of hyper-parameters that are listed in Section 5.4. To validate the generalisation of our reported results, we performed an across study analysis through several experiments with multiple combinations of learning rate and regularization rate. Figures 5.6, 5.7, 5.8 and 5.9 provides the result of these analyses. Each graph shows the mean of the MAP values at each epoch for each algorithm along with the error bars for the across study evaluation.

FIGURE 5.6: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for WA dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars.

Figure 5.6 shows the mean of MAP for different experiments with WA dataset, along with the error bars showing the standard deviation of the MAPs from the mean. For this dataset, the SMC-BPR algorithm works almost similar to the original BPR. We believe that the result is due to the sparsity problem in WA dataset. As was reported, this data is the most sparse dataset in our experiments. This yields a poor modeling of items using the matrix factorization algorithm and hence the SMC-BPR is not able to correctly detect the most similar items to use for optimization. Therefore, the result of the proposed algorithm is similar to the original BPR that approximates the posterior using bootstrap sampling. However, as was reported in Figure 5.2, specific configuration of this algorithm can perform significantly superior to the original BPR. The result from this graph also provides evidence that the SMC-BPR approach works superior to the alternative approach, using the across study evaluation.

FIGURE 5.7: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for ML dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars.

FIGURE 5.8: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Frappe dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars.

Figure 5.7 and 5.8 shows a similar graph to Figure 5.6 for ML dataset and Frappe dataset respectively. The two graphs provides evidence that the SMC-BPR approach performs superior to the original BPR and the alternative approach, using an across study evaluations. For these two datasets, as the sparsity is less compared with the WA dataset, our proposed algorithm is able to detect informative samples for the learning algorithm. This results in the prediction of more accurate list of items for the users that is evaluated by MAP. Both ML and Frappe datasets has approximately similar sparsity and hence both datasets show approximately similar result in terms of MAP for across study evaluations.

FIGURE 5.9: The comparison of the three algorithms, Blue: BPR-MF, Red: Adaptive sampling-BPR (AS-BPR-MF) and Green: Similarity based Monte Carlo-BPR (SMC-BPR-MF) for Jester dataset in terms of MAP on each training epoch for test instances. Each line shows the mean of the MAPs at each epoch along with the error bars.

Jester dataset is the least sparse dataset used in our experiments, with millions of observations and small number of items. As illustrated in Figure 5.9, the across study evaluation of MAP shows a small error bar for the experiments with variations of learning rate and regularization rates. This result is observed for all 3 algorithms and is more illustrated for the proposed SMC-BPR algorithm. The result for the proposed algorithm shows almost zero deviation from the mean. This means that the algorithm is almost stable across different variations of learning rate and regularization rates for this dataset. The result confirms that the proposed algorithm performs superior to both the original BPR and the alternative approach, when the data is less sparse.

The across study evaluations provide evidence that the proposed algorithm, SMC-BPR performs superior to the baseline BPR and the alternative approach, AS-BPR, when sparsity is low. As data sparsity increases the algorithm becomes less stable and more dependent on the hyper parameters, such as learning rate and regularization rates.

5.7 Threats to Validity

There are different ways to categorise the threats to validity of a study in the literature. Campbell and Stanley considered two types of threats to validity as external and internal validity [127]. Later, Cook and Campbell extended the threats to the validity of each experiment to four main categories, namely: internal, external, conclusion and construct validity [128]. In this dissertation, we adopt the latter and outline the possible threats to validity of our experiments based on four categories of validity. We also explain how we mitigated those threats in our experiments.

Internal validity concerns with the causal relationship between the response variables and independent variables based on the measurement, research setting, and the whole research design [129].

One of the threats to the internal validity of our study is sampling bias. To mitigate this threat in our experiments, we repeated the experiments multiple times for different samples to ensure that the behaviour of the model is not biased towards a specific sampled data.

Another threat to the internal validity of our study can be the scripts. All scripts for study are developed by the author and is available at the github repository for replication purposes ¹. The scripts for exploratory analysis, the visualisations and final analysis of the results are developed with R software, while the algorithm is developed on top of the theano-bpr library in python2.7. This threat was mitigated by ongoing debugging of the scripts by the author and a reviewer.

External validity concerns with how much the result of the experiment can be generalized to other application settings and how it can be transferred to other researchers to build their studies on [129].

The threats to the external validity of our study comes from the data used for our analysis. This study, was performed on three publicly available datasets and

¹<https://github.com/RyersonU-DataScienceLab/parisa>

one unique dataset extracted during our collaboration with IBM company. It is possible to argue that the datasets used in this research may not be a good representatives of the data from all recommendation domains. However, we performed our experiments on 4 different recommendation domains with different characteristics and the results were consistent in all four datasets. This threat is also diminished by the fact that the three public datasets are used by various researchers in the field to evaluate their work. The unique dataset was also collected, cleaned, pre-processed and evaluated with extra caution. This process was also reviewed multiple times by reviewers of the published research [81, 118] and the author.

For other researchers to be able to build upon our study, we documented our experiment design in this manuscript. Also, all scripts used in the analysis are available at our github repository for further references.

Construct validity is the degree to which the researchers can measure what they claimed in their study based on their research questions [129].

Our research question was related to the improvement of the performance of BPR based on the accuracy of a ranked list of items recommended to the users. In order to mitigate the biases due to the performance measurements, we performed the analysis in different experiment settings, where we observed a consistent result. We also performed a cross study evaluation to validate our experiment results. The evaluation measure that we used in this study reflects the research question that was addressed through this work similar to other studies in the literature [115].

Conclusion Validity is concerned with the extent to which the results are dependent on a specific researcher. This is concerned with the level of reproducibility of each study [129].

The statistical conclusion validity is one of the main conclusion validity concerns. It is defined as the degree to which the conclusion from the data is correct. Most of the statistical tests have some assumptions. Violating these assumptions may lead

to incorrect conclusions about the relationship between variables. To mitigate the threats to the statistical validity of our study, we carefully checked the assumptions of the statistical tests not to violate any assumptions.

To mitigate the risk of researcher’s bias and the risk of conclusion validity threat, we strictly followed the guidelines by Alpydin [33] to design our experiments. All steps are explained in details in the experiment and results Section. Also all scripts are documented and available at our github¹ repository for reproducibility purposes.

¹<https://github.com/RyersonU-DataScienceLab/parisa>

Chapter 6

Conclusion and Future Directions

Recommender Systems (RS) emerged to offer solutions to mitigate the information overload problem [7]. Recommender Systems are software tools and techniques that provide users with a list of items that might be of their interest, based on their historical preferences. These suggestions can be related to various decision making tasks, such as what items to buy, which mobile application to select, which movie to watch, or what news item to review. These tools are widely used in World Wide Web and e-commerce applications to help users find the most suitable items (e.g. movies, books, products or travels).

There are various models and algorithms to perform the recommendation tasks in different domains [2]. Each model is designed to work with the type of the available information from user's feedback [2]. One of the main and most available source of user's feedback is user's click or item selection [12]. This source of user preference information is normally referred to as implicit information as the preference should be implicitly derived from the user action [12]. In the domain of Recommender System design, this information is one of the most valuable source of information, as this does not need any user effort to generate and can be extracted from user interaction log [12]. Therefore, in this research, we decided to focus on the situations, where only a single source implicit information is available.

The factorization based Bayesian Personalized Ranking (BPR-MF) algorithm for implicit information is widely used in the Recommender System literature [1, 18, 22, 31, 32, 79, 80]. This algorithm follows a Bayesian machine learning framework to design a recommender system using a pairwise comparison learning to rank approach. However, the BPR-MF algorithm does not perform well as the item space gets large. This is because as the number of items increases the search space becomes larger. The bootstrap random sampling approximation that is suggested in the original BPR approach does not perform well, as it might not select the samples that contribute to the learning process.

In this research, we propose an extension to the Bayesian Personalized Ranking(BPR) framework that deals with the posterior approximation problem in the original approach. We proposed a similarity based Monte Carlo Markov Chain (MCMC) method to approximate the posterior distribution in BPR. Specifically, the Metropolis algorithm was adopted based on the cosine similarity of the latent factor vectors representing the items in the Matrix Factorization model.

The results provide evidence that our proposed algorithm (SMC-BPR-MF) performs significantly better than the original BPR-MF algorithm [1] in terms of the accuracy of the recommended ranked list of items. This is more generalizable for less sparse datasets. For the datasets with large sparsity, the correct selection of the hyper parameters generates more accurate results.

The proposed algorithm also significantly outperforms an alternative approach presented in [18], which attempts to tackle the similar approximation problem in BPR. The alternative approach also suggest the informative approximation of the posterior. However, the sample selection with latent factors updates rarely, and in our datasets with large item space the updates almost never occurs. This yields to a poor approximation of the posterior, as the samples to approximate the posterior do not contribute to the learning process. Hence, the gradient descent algorithm rarely learns the model parameter and it settles in the closest local optima. The result is consistent for all datasets used in our experiments based on the Mean Average Precision performance measure [114]. Also, the across study evaluation

provides evidence that the alternative model is the least stable algorithm and the most dependent on hyper parameter settings among the three.

6.1 Contributions

The contributions of this research can be summarised as both theoretical contributions and practical implications in the following sections.

6.1.1 Theoretical and Methodological Contributions

In this research, we design a recommender system algorithm, using a single source implicit information. Specifically, we propose a solution to solve the approximate inference problem in Bayesian Personalized Ranking approach. The theoretical contribution of our proposed solution is two folds. First we proposed a solution to find the most informative samples for the gradient descent algorithm to learn from. Second, we provided an approximate solution to reduce the number of computations, while keeping the performance higher than the baseline models.

- *Similarity based solution for BPR approximations:* The similarity based solution provides a framework to search for a decision boundary [33], separating the positive and negative items for a given user. As the number of items increase detecting this boundary becomes a harder problem to solve exactly, which forces researchers to come up with efficient approximate methods. SB-BPR presents an alternative approximate solution to better estimate the decision boundary between the positive and negative items in BPR framework. The correct detection of this boundary yields to the selection of the most informative samples that contribute to the learning algorithm. The gradient descent algorithm is then able to find the parameters of the model in a more precise manner, which results in the prediction of a more accurate list of items for the users.

- *Monte Carlo Markov Chain method to approximate the similarity based solution:* Monte Carlo Markov Chain sampling method is proposed on top of the similarity solution to find the most similar samples with less number of computations. A factorization based scoring model is used along with the proposed approach to test its performance of the SMC-BPR approach. The Similarity based Monte Carlo-BPR (SMC-BPR) approach with Matrix Factorization scoring model performs significantly more accurate than the baseline BPR approach and the alternative approach addressing the same approximation problem in BPR using the similar scoring function (i.e. Matrix Factorization).

6.1.2 Practical Implications

The motivation behind this research was the formulation and the design of recommender system for IBM Watson Analytics application. Hence, the practical implication of this work can be listed as follows:

- *Formulation of IBM Watson Analytics recommendations as a learning based Recommender system problem:* The current engine behind the IBM Watson Analytics application provides visualisation suggestions based on a set of predefined rules. The system provides similar suggestions to the users regardless of their preferences. We formulate the user interaction with WA system as a learning based problem, specifying the items and modeling user preferences. We then built recommender system algorithms that is able to learn from user's historical preferences and provide personalised recommendations. One of this algorithms is explained in our prior works [81] also attached to this manuscript as appendix A. The current manuscript is dedicated to the explanation of the algorithm that deals with a single source implicit information.
- *IBM Watson Analytics User Interaction Data Extraction:* To validate the performance of our methodology and the design of the recommender system

algorithm, we extracted a unique dataset from user interaction with IBM Watson Analytics application. This data is reflective of actual user's preferences implicitly derived from user interaction with the application. The data was pre-processed and investigated in our prior works [81, 118]. During our second round data extraction from the second version of this application, we noticed the scarcity of user information. The motivation behind the current research, designing a recommender system with single source implicit information is due to the data scarcity problem in IBM Watson Analytics application.

- *More accurate recommender system algorithm with single source implicit user preference information:* Although the motivation behind this study was the formulation and the design of a recommender system algorithm for IBM Watson Analytics application, our solution is generalizable to other recommender system applications. Essentially, in this work, we proposed an extension to the BPR that provides more accurate list of items to the users. Our proposed approach uses a single source implicit information, similar to the original BPR method. The SMC-BPR approach can be used with any scoring function such as Matrix Factorization.

Throughout our experiments, we observed that the computational complexity of our model is comparable to the original BPR and the alternative approach. We also observed the the overall accuracy and stability of the system is directly dependent on the sparsity of the input data. The accuracy of the algorithms and their stability decreases as the sparsity of the input data increases. This is consistent for all variations of BPR tested in this work.

6.2 Future Direction

As future work, we plan to provide a parallelised implementation of our proposed algorithm. The current model uses a single core for the analysis and its performance is comparable to the original BPR. A parallelised version would perform the analysis in a shorter time span and can deal with larger amount of input data.

The proposed approach in this research can also be extended using any additional information to the single source implicit feedback. Any extensions to the original BPR algorithm that uses additional information may also be considered as an extension to our proposed approach. Going forward, we would like to test the performance of our proposed model, using additional sources of information.

Bibliography

- [1] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.
- [2] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.
- [3] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.
- [4] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.
- [5] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.
- [6] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.
- [7] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6): 734–749, 2005.

- [8] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.
- [9] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.
- [10] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.
- [11] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
- [12] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83. WoUongong, 1998.
- [13] Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 85–92. ACM, 2013.
- [14] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.
- [15] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*, pages 1593–1600, 2008.
- [16] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46: 109–132, 2013.

- [17] Tong Zhao, Julian McAuley, and Irwin King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 261–270. ACM, 2014.
- [18] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 273–282. ACM, 2014.
- [19] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 187–196. ACM, 2009.
- [20] Weiyu Guo, Shu Wu, Liang Wang, and Tieniu Tan. Multiple attribute aware personalized ranking. In *Asia-Pacific Web Conference*, pages 244–255. Springer, 2015.
- [21] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 785–788. ACM, 2013.
- [22] Huihuai Qiu, Guibing Guo, Jie Zhang, Zhu Sun, Hai Thanh Nguyen, and Yun Liu. Tbpr: Trinity preference based bayesian personalized ranking for multivariate implicit feedback. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 305–306. ACM, 2016.
- [23] Weike Pan and Li Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, volume 13, pages 2691–2697, 2013.
- [24] Christopher M Bishop and Tom M Mitchell. Pattern recognition and machine learning. 2014.

- [25] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.
- [26] Steven M Shugan. The cost of thinking. *Journal of consumer Research*, 7(2):99–111, 1980.
- [27] Ladislav Peska and Peter Vojtas. Using implicit preference relations to improve recommender systems. *Journal on Data Semantics*, 6(1):15–30, 2017.
- [28] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40. ACM, 2001.
- [29] Keunho Choi, Donghee Yoo, Gunwoo Kim, and Yongmoo Suh. A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications*, 11(4):309–317, 2012.
- [30] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 493–494. ACM, 2013.
- [31] Shuang Qiu, Jian Cheng, Ting Yuan, Cong Leng, and Hanqing Lu. Item group based pairwise preference learning for personalized ranking. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1219–1222. ACM, 2014.
- [32] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, 73:173–180, 2015.
- [33] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.
- [34] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19(4):285–330, 2003.

- [35] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34. ACM, 2008.
- [36] Philip M Podsakoff, Scott B MacKenzie, Jeong-Yeon Lee, and Nathan P Podsakoff. Common method biases in behavioral research: A critical review of the literature and recommended remedies. *Journal of applied psychology*, 88(5):879, 2003.
- [37] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.
- [38] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of machine learning research*, 10(Mar):623–656, 2009.
- [39] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1633–1647, 2017.
- [40] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.
- [41] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
- [42] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

- [43] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.
- [44] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.
- [45] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.
- [46] Miha Grčar, Blaž Fortuna, Dunja Mladenič, and Marko Grobelnik. knn versus svm in the collaborative filtering framework. In *Data Science and Classification*, pages 251–260. Springer, 2006.
- [47] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM, 2003.
- [48] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [49] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.
- [50] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.
- [51] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

- [52] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006.
- [53] Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating tags in a semantic content-based recommender. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 163–170. ACM, 2008.
- [54] Parisa Lak, Ceni Babaoglu, Ayse Basar Bener, and Pawel Pralat. News article position recommendation based on the analysis of article’s content-time matters. In *CBRecSys@ RecSys*, pages 11–14, 2016.
- [55] Tariq Mahmood and Francesco Ricci. Towards learning user-adaptive state models in a conversational recommender system. In *LWA*, pages 373–378. Citeseer, 2007.
- [56] Francesco Ricci, Dario Cavada, Nader Mirzadeh, Adriano Venturini, et al. Case-based travel recommendations. *Destination recommendation systems: behavioural foundations and applications*, pages 67–93, 2006.
- [57] Derek Bridge, Mehmet H Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3): 315–320, 2005.
- [58] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *International Conference on Trust Management*, pages 93–104. Springer, 2006.
- [59] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.
- [60] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

- [61] Badrul Munir Sarwar. *Sparsity, scalability, and distribution in recommender systems*. University of Minnesota, 2001.
- [62] Deepa Anand and Kamal K Bharadwaj. 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, 2011.
- [63] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.
- [64] Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *Icml*, volume 98, pages 46–54, 1998.
- [65] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151, 2001.
- [66] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.
- [67] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1): 37–51, 2008.
- [68] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: tag informed collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 69–76. ACM, 2009.
- [69] Anirban Basu, Jaideep Vaidya, and Hiroaki Kikuchi. Efficient privacy-preserving collaborative filtering based on the weighted slope one predictor. *J. Internet Serv. Inf. Secur.*, 1(4):26–46, 2011.

- [70] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.
- [71] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
- [72] Evgeny Frolov and Ivan Oseledets. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3), 2017.
- [73] Cyrus Shahabi and Yi-Shin Chen. Web information personalization: Challenges and approaches. In *International Workshop on Databases in Networked Information Systems*, pages 5–15. Springer, 2003.
- [74] Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel sgd for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 249–256. ACM, 2013.
- [75] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems*, 41(3):793–819, 2014.
- [76] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 765–774. IEEE, 2012.
- [77] Michael Reusens, Wilfried Lemahieu, Bart Baesens, and Luc Sels. A note on explicit versus implicit information for job recommendation. *Decision Support Systems*, 98:26–35, 2017.

- [78] Gawesh Jawaheer, Peter Weller, and Patty Kostkova. 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, 2014.
- [79] Jing Wang, Lanfen Lin, Heng Zhang, and Jiaqi Tu. Confidence-learning based collaborative filtering with heterogeneous implicit feedbacks. In *Asia-Pacific Web Conference*, pages 444–455. Springer, 2016.
- [80] Hongzhi Liu, Zhonghai Wu, and Xing Zhang. Cplr: Collaborative pairwise learning to rank for personalized recommendation. *Knowledge-Based Systems*, 2018.
- [81] Parisa Lak, Can Kavaklioglu, Mefta Sadat, Martin Petitclerc, Andriy V Miransky, Graham Wills, and Ayse Basar Bener. A probabilistic approach for modelling user preferences in recommender systems. In *CASCON*, 2017.
- [82] Yehuda Koren and Joe Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 117–124. ACM, 2011.
- [83] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. 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*, pages 155–164. ACM, 2012.
- [84] Olivier Chapelle, Yi Chang, and T-Y Liu. Future directions in learning to rank. In *Proceedings of the Learning to Rank Challenge*, pages 91–100, 2011.
- [85] Jing Wang, Lan-fen Lin, Heng Zhang, Jia-qi Tu, and Peng-hua Yu. A novel confidence estimation method for heterogeneous implicit feedback. *Frontiers of Information Technology & Electronic Engineering*, 18(11):1817–1827, 2017.

- [86] Sheng Wang, Xiaobo Zhou, Ziqi Wang, and Ming Zhang. Please spread: recommending tweets for retweeting with implicit feedback. In *Proceedings of the 2012 workshop on Data-driven user behavioral modelling and mining from social media*, pages 19–22. ACM, 2012.
- [87] Tiejian Luo, Su Chen, Guandong Xu, and Jia Zhou. *Trust-based collective view prediction*. Springer, 2013.
- [88] Weike Pan and Li Chen. Group bayesian personalized ranking with rich interactions for one-class collaborative filtering. *Neurocomputing*, 207:501–510, 2016.
- [89] Weiyu Guo, Shu Wu, Liang Wang, and Tieniu Tan. Personalized ranking with pairwise factorization machines. *Neurocomputing*, 214:191–200, 2016.
- [90] Jane Elith, Steven J Phillips, Trevor Hastie, Miroslav Dudík, Yung En Chee, and Colin J Yates. A statistical explanation of maxent for ecologists. *Diversity and distributions*, 17(1):43–57, 2011.
- [91] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.
- [92] Neil Houlsby. *Efficient Bayesian active learning and matrix modelling*. PhD thesis, University of Cambridge, 2014.
- [93] Ferenc Huszr. *Scoring rules, divergences and information in Bayesian machine learning*. PhD thesis, University of Cambridge, 2013.
- [94] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.
- [95] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)*, 71(2):319–392, 2009.

- [96] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural computation*, 12(11):2655–2684, 2000.
- [97] John Von Neumann. 13. various techniques used in connection with random digits. *Appl. Math Ser*, 12(36-38):3, 1951.
- [98] Andrew W Marshall. The use of multi-stage sampling schemes in monte carlo computations. Technical report, RAND CORP SANTA MONICA CALIF, 1954.
- [99] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.
- [100] Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38(1):1–17, 1991.
- [101] Mark Jerrum and Alistair Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.
- [102] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.
- [103] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- [104] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [105] Alan Herschtal and Bhavani Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*, page 49. ACM, 2004.

- [106] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
- [107] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.
- [108] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.
- [109] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.
- [110] Parisa Lak, Bora Caglayan, and Ayse Basar Bener. The impact of basic matrix factorization refinements on recommendation accuracy. In *Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing*, pages 105–112. IEEE Computer Society, 2014.
- [111] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.
- [112] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.
- [113] Thomas K Landauer and Susan T Dumais. A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.
- [114] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

- [115] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
- [116] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3):299–310, 2005.
- [117] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.
- [118] Parisa Lak, Mefta Sadat, Carl Julien Barrelet, Martin Petitclerc, Andriy V Miransky, Craig Statchuk, and Ayse Basar Bener. Preliminary investigation on user interaction with ibm watson analytics. In *CASCON*, pages 218–225, 2016.
- [119] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.
- [120] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR*, abs/1505.03014, 2015. URL <http://arxiv.org/abs/1505.03014>.
- [121] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.
- [122] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

- [123] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.
- [124] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126. ACM, 2010.
- [125] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 88–96. ACM, 2008.
- [126] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.
- [127] Donald T Campbell and Julian C Stanley. Experimental and quasi-experimental designs for research. *Handbook of research on teaching. Chicago, IL: Rand McNally*, 1963.
- [128] Thomas D Cook, Donald Thomas Campbell, and Arles Day. *Quasi-experimentation: Design & analysis issues for field settings*, volume 351. Houghton Mifflin Boston, 1979.
- [129] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.