Scalable Bayesian Factorization Models for Recommender Systems

A THESIS

submitted by

AVIJIT SAHA

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

JUNE 2016

THESIS CERTIFICATE

This is to certify that the thesis titled **Scalable Bayesian Factorization Models for Recommender Systems**, submitted by **Avijit Saha**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Science**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. B Ravindran
Research Guide
Associate Professor
Dept. of CSE
IIT-Madras, 600 036

Place: Chennai

Date: June 08, 2016

ACKNOWLEDGEMENTS

Throughout my stay here, I was fortunate enough to be surrounded by an amazing set of people. I would like to take this opportunity to extend my gratitude to these people. Firstly, I would like to thank my advisor Balaraman Ravindran, who provided me the freedom to explore several areas, and supported me throughout the course. He was more than a teacher and a collaborator from my day one here, a complete mentor. Often I get amazed looking at his diverse knowledge in several domains.

I am charmed to have worked with Professors and colleagues at IIT Madras and elsewhere, who have been of utmost importance during my education. I would like to thank Ayan Acharya, with whom I have started working towards the end of my course on multiple Bayesian nonparametric factorization model problems. He is a fantastic collaborator. He helped me to get a clear understanding on the recent advances in Bayesian statistics. I would like to thank Ayan Acharya and Joydeep Ghosh for the collaboration which resulted in an ICDM paper. I would also like to thank Rishabh, who worked with me for a long time on Bayesian factorization models. I got a lot of support from him. It was a pleasure working with Janarthanan and Shubhranshu for the Recsys challenge 2014. I am also thankful to Nandan Sudarsanam for his time on several discussions on bandit problems. I am grateful to Mingyuan Zhou, who gave me the opportunity to collaborate with him on nonparametric dynamic network modeling.

I am largely indebted to my friends without whom I can hardly imagine a single day in the Institute. First of the lot, I would like to thank Saket, with whom I attended several courses. He always stood besides me during my tough days. I am thankful to Animesh and Sai for their support. Also I would like to thank Vishnu, Sudarsan, Aman, Arnab, Ahana, Shamshu, Abhinoy, Vikas, Pratik, Sarath, Deepak, Chandramohan, Priyesh, and Rajkaran for all their help. Thanks to Arpita, Aditya, Viswa, Nandini, Sandhya, and all my 2012 MS friends for making my life at IIT Madras cherishable.

I owe many thanks to my advisor Balaraman Ravindran for helping me to get grants for my research. I am grateful to Ericsson India for funding my research.

Finally, I would like to thank my family for being one of my primary source of inspiration, without whom I would not have been able to complete the course. I specially want to thank my mother Shikha Saha, who has supported me always, and has been my biggest strength.

To the memory of my father, Dipak Saha.

ABSTRACT

KEYWORDS: Recommender Systems; Collaborative Filtering; Latent Variable Models; Factorization Models; Matrix Factorization; Factorization Machine; Probabilistic Modeling; Scalable; Bayesian; Variational Bayes; Markov Chain Monte Carlo; Nonparametric.

In recent years, Recommender Systems (RSs) have become ubiquitous. Factorization models are a common approach to solve RSs problems, due to their simplicity, prediction quality and scalability. The idea behind such models is that preferences of a user are determined by a small number of unobserved latent factors. One of the most well studied factorization model is matrix factorization using the Frobenius norm as the loss function. In more challenging prediction scenarios where additional "side-information" is available and/or features capturing higher order may be needed, new challenges due to feature engineering needs arise. The side-information may include user specific features such as age, gender, demographics, and network information and item specific information such as product descriptions. While interactions are typically desired in such scenarios, the number of such features grows very quickly. This dilemma is cleverly addressed by the Factorization Machine (FM), which combines high prediction quality of factorization models with the flexibility of feature engineering. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization, SVD++, TimeSVD++, Pairwise Interaction Tensor Factorization (PITF), and factorized personalized Markov chains (FPMC). Also, due to the availability of large data, several sophisticated probabilistic factorization models have been developed. However, in spite of having a vast literature on factorization models, several problems exist with different factorization model algorithms.

In this thesis, we take a probabilistic approach to develop several factorization models. We adopt a fully Bayesian treatment of these models and develop scalable approximate

inference algorithms for them.

Bayesian Probabilistic Matrix Factorization (BPMF), which is a Markov chain Monte Carlo (MCMC) based Gibbs sampling inference algorithm for matrix factorization, provides state-of-the-art performance. BPMF uses multivariate Gaussian prior on latent factor vector which leads to cubic time complexity with respect to the dimension of latent space. To avoid this cubic time complexity, we develop the Scalable Bayesian Matrix Factorization (SBMF) which considers independent univariate Gaussian prior over latent factors. SBMF, which is a MCMC based Gibbs sampling inference algorithm for matrix factorization, has linear time complexity with respect to the dimension of latent space and linear space complexity with respect to the number of non-zero observations.

We then develop the Variational Bayesian Factorization Machine (VBFM) which is a batch scalable variational Bayesian inference algorithm for FM. VBFM converges faster than the existing state-of-the-art MCMC based inference algorithm for FM while providing similar performance. Additionally for large scale learning, we develop the Online Variational Bayesian Factorization Machine (OVBFM) which utilizes stochastic gradient descent to optimize the lower bound in variational approximation. OVBFM outperforms existing online algorithms for FM as validated by extensive experiments performed on numerous large-scale real world data.

Finally, the existing inference algorithm for FM assumes that the data is generated from a Gaussian distribution which may not be the best assumption for count data such as integer-valued ratings. Also, to get the best performance, one needs to cross-validate over the number of latent factors used for modeling the pairwise interaction in FM, a process that is computationally intensive. To overcome these problems, we develop the Nonparametric Poisson Factorization Machine (NPFM), which models count data using the Poisson distribution, provides both modeling and computational advantages for sparse data. The ideal number of latent factors is estimated from the data itself. We also consider a special case of NPFM, the Parametric Poisson Factorization Machine (PPFM), that considers a fixed number of latent factors. Both PPFM and NPFM have linear time and space complexity with respect to the number of non-zero observations. Using extensive experiments, we show that our methods outperform two strong baseline methods by large margins.

TABLE OF CONTENTS

ABBREVIATIONS

RSs	Recommender Systems
CF	Collaborative filtering
SVMs	Support Vector Machines
SGD	Stochastic gradient descent
MCMC	Markov chain Monte Carlo
PMF	Probabilistic Matrix Factorization
BPMF	Bayesian Probabilistic Matrix Factorization
FM	Factorization Machine
PITF	Pairwise Interaction Tensor Factorization
FPMC	Factorized personalized Markov chains
SBMF	Scalable Bayesian Matrix Factorization
VBFM	Variational Bayesian Factorization Machine
OVBFM	Online Variational Bayesian Factorization Machine
NPFM	Nonparametric Poisson Factorization Machine
PPFM	Parametric Poisson Factorization Machine
RMSE	Root mean square error
NDCG	Normalized Discounted Cumulative Gain
IDCG	Ideal Discounted Cumulative Gain

LIST OF TABLES

LIST OF FIGURES

NOTATION

$\boldsymbol{X}$	Matrix
$\boldsymbol{x}$	Column vector, unless explicitly specified as row vector
T	Transpose
$\boldsymbol{X}_{\cdot j}$	Summed over index i
$(i,j) : i < j$	All (i,j) pairs with $i < j$
Γ	Gamma function
$\mathcal{N}(\cdot,\cdot)$	Gaussian distribution
Gamma$(\cdot,\cdot)$	Gamma distribution
ΓP$(\cdot,\cdot)$	Gamma process
NB$(\cdot,\cdot)$	Negative binomial distribution
CRT$(\cdot,\cdot)$	Chinese restaurant table
Pois$(\cdot,\cdot)$	Poisson distribution
$p(\cdot\|\cdot)$	Probability distribution
$x \sim$	Distribution of random variable x
$\cdot\|-$	Conditional probability distribution (conditioned on everything except $\cdot$)

CHAPTER 1

INTRODUCTION

"Information overload occurs when the amount of input to a system exceeds its processing capacity. Decision makers have fairly limited cognitive processing capacity. Consequently, when information overload occurs, it is likely that a reduction in decision quality will occur" - Speier et al. (1999).

Information overload has become a major problem in recent years with the advancement of Internet. Social media users and microbloggers receive large volume of information, often at a higher rate than they can effectively and efficiently consume. Often, users face difficulties in finding relevant items due to the sheer volume of information present in the web, for e.g., finding relevant books from Amazon.com book catalog. Recommender Systems (RSs) and web search help users to systematically and efficiently access information and help to avoid the information overload problem.

In recent years, RSs have become ubiquitous. RSs provide suggestions of items to users for various decision making processes such as: what song to listen, what book to buy, what movie to watch, etc. Often RSs are built for personalized recommendation and provide user specific suggestions. As few examples: Amazon.com employs a RS to personalize the online store for each customer; Netflix provides movie recommendation based on users' past rating history, current data-query, and users' profile information; Twitter uses personalized tweet recommendation engine to suggest relevant tweets to users.

Collaborative filtering (CF) [Su and Khoshgoftaar, 2009; Lee *et al.*, 2012; Bell and Koren, 2007] is widely used in RSs and have proven to be very successful. CF takes multiple user's preferences into account to recommend items to a user. The fundamental assumption of CF is that if two user's preferences are same on some number of items, then their behavior will be same on the other unseen items. CF [Bell and Koren, 2007] can be viewed as missing value prediction task where given a user-item matrix of scores with many missing values, the task is to estimate the missing entries in the matrix based on the

given ones. Memory-based CF algorithms [Su and Khoshgoftaar, 2009] were a common choice in RSs earlier due to their simplicity and scalability. Neighborhood-based (kNN) CF algorithms [Su and Khoshgoftaar, 2009; Bell and Koren, 2007] are prevalent memory-based CF techniques which identify pairs of items that have similar rating behavior or users with similar rating patterns, to find unknown user-item relationship. However memory-based methods are unreliable when the data are sparse [Su and Khoshgoftaar, 2009]. In order to achieve better accuracy and alleviate the short comings of memory-based CF, model-based CF approaches [Hofmann, 2004; Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007] have been investigated.

Model-based CF fits a parametric model to the training data, which is used later to predict the unknown user-item ratings. One particular type of model-based CF algorithms is based on latent variable models such as, pLSA [Hofmann, 2004], neural networks [Salakhutdinov *et al.*, 2007], Latent Dirichlet Allocation [Blei *et al.*, 2003], and matrix factorization [Salakhutdinov and Mnih, 2007] which try to uncover hidden features that explain the ratings. Latent variable models involve supplementing a set of observed variables with additional latent, or hidden, variables. In probabilistic latent variable model framework, the distribution over the observed variables is obtained by marginalizing out the hidden variables from the joint distribution over observed and hidden variables. The hidden structure of the data can be computed and explored through the posterior which is defined as the conditional distribution of the latent variables given the observations. This hidden structure, computed through the posterior distribution, is useful in prediction and exploratory analysis.

Factorization models [Koren *et al.*, 2009; Koren, 2009; Salakhutdinov and Mnih, 2008; Xiong *et al.*, 2010] are a class of latent variable models and have received extensive attention in the RSs community, due to their simplicity, prediction quality, and scalability. These models represent both users and items using a small number of unobserved latent factors. Hence, each user/item is associated with a latent factor vector. Elements of a item's latent factor measure the extent to which the item possesses those factors and elements of a user's latent factor measure the extent of interest the user has in items that are high on the corresponding factors. Throughout the thesis, we will adopt a Bayesian approach to

analyze different factorization models.

One of the most well studied factorization model is matrix factorization [Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007, 2008; Gopalan *et al.*, 2015] using the Frobenius norm as the loss function. Formally, matrix factorization recovers a low-rank latent structure of a matrix by approximating it as a product of two low rank matrices. A popular approach to solve matrix factorization is to minimize the regularized squared error loss. The optimization problem can be solved using stochastic gradient descent (SGD) [Koren *et al.*, 2009]. SGD is an online optimization algorithm which obviates the need to store the entire dataset in memory and hence is often preferred for large scale learning due to memory and speed considerations [Silva and Carin, 2012]. Though SGD is scalable and enjoys local convergence guarantee [Sato, 2001], it often overfits the data and requires manual tuning of the learning rate and the regularization parameters [Salakhutdinov and Mnih, 2007]. On the other hand, Bayesian methods [Salakhutdinov and Mnih, 2008; Beal, 2003; Tzikas *et al.*, 2008; Hoffman *et al.*, 2013] for matrix factorization automatically tune learning rate and regularization parameters and are robust to overfitting. Bayesian Probabilistic Matrix Factorization (BPMF) [Salakhutdinov and Mnih, 2008] directly approximate the posterior distribution using Markov chain Monte Carlo (MCMC) based Gibbs sampling inference mechanism and outperform the variational based approximation. BPMF uses multivariate Gaussian distribution as prior on latent factor vector which leads to cubic time complexity with respect to the dimension of latent space. Hence, many times it is difficult to apply BPMF on very large datasets.

In more challenging prediction scenarios where additional "side-information" is available and/or features capturing higher order may be needed, new challenges due to feature engineering needs arise. The side-information may include user specific features such as age, gender, demographics, and network information and item specific information such as product descriptions. While interactions are typically desired in such scenarios, the number of such features grows very quickly. This dilemma is cleverly addressed by the Factorization Machine (FM) [Rendle, 2010], which combines high prediction quality of factorization models with the flexibility of feature engineering. FM represents data as real-valued features as in standard machine learning approaches, such as Support Vec-

tor Machines (SVMs), and uses interactions between each pair of variables as well but constrained to a low-dimensional latent space. By restricting the latent space, the number of parameters needed is kept manageable. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [Koren *et al.*, 2009], SVD++ [Koren, 2008], TimeSVD++ [Koren, 2009], Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme, 2010], and factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2011*a*]. Other advantages of FM include – 1) FM allows parameter estimation with extremely sparse data where SVMs fail; 2) FM has linear complexity, can be optimized in the primal and, unlike SVMs, does not rely on support vectors; and 3) FM is a general predictor that can work with any real valued feature vector, while several state-of-the-art factorization models work only on very restricted input data.

FM is usually learned using stochastic gradient descent (SGD) [Rendle, 2010]. FM that uses SGD for learning is conveniently addressed as SGD-FM in this thesis. As mentioned above, though SGD is scalable and enjoys local convergence guarantees, it often overfits the data and needs manual tuning of learning rate and the regularization parameters. Alternative methods to solve FM include Bayesian Factorization Machine [C. Freudenthaler, 2011] which provides state-of-the-art performance using MCMC based Gibbs sampling as the inference mechanism and avoids expensive manual tuning of the learning rate and regularization parameters (this framework is addressed as MCMC-FM). However, MCMC-FM is a batch learning method and is less straight forward to scale to datasets as large as the KDD music dataset [Dror *et al.*, 2012]. Also, it is difficult to preset the values of burn-in and collection iterations and gauge the convergence of the MCMC inference framework, a known problem with sampling based techniques.

Also, MCMC-FM assumes that the observations are generated from a Gaussian distribution which, obviously, is not a good fit for count data. Additionally, for both SGD-FM and MCMC-FM, one needs to solve an expensive model selection problem to identify the optimal number of latent factors. Alternative models for count data have recently emerged that use discrete distributions, provide better interpretability and scale only with the number of non-zero elements [Gopalan *et al.*, 2014*b*; Zhou and Carin, 2015; Zhou *et al.*, 2012; Acharya *et al.*, 2015].

In this thesis, we take a probabilistic approach to develop different factorization models and build scalable approximate posterior inference algorithms for them. These factorization models discover hidden structure from the data using the posterior distribution of hidden variables given the observations which is used for prediction purpose.

1.1 Contribution of the Thesis

The contributions of the thesis are as follows:

- Scalable Bayesian Matrix Factorization (SBMF): SBMF considers independent univariate Gaussian prior over latent factors as opposed to multivariate Guassian prior in BPMF. We also incorporate bias terms in the model which are missing in baseline BPMF model. Similar to BPMF, SBMF is a MCMC based Gibbs sampling inference algorithm for matrix factorization. SBMF has linear time complexity with respect to the dimension of latent space and linear space complexity with respect to the number of non-zero observations. We show extensive experiments on three large-scale real world datasets to validate that the SBMF takes less time than the baseline method BPMF and incurs small performance loss.

- Variational Bayesian Factorization Machine (VBFM): VBFM is a batch variational Bayesian inference algorithm for FM. VBFM converges faster than MCMC-FM and performs as good as MCMC-FM asymptotically. The convergence is also easy to track when the objective associated with the variational approximation in VBFM stops changing significantly.

- Online Variational Bayesian Factorization Machine (OVBFM): OVBFM uses SGD for maximizing the lower bound obtained from the variational approximation and performs much better than the existing online algorithm of FM that uses SGD. As considering single data instance increases the variance of the algorithm, we consider a mini-batch version of OVBFM.

  Extensive experiments on four real world movie review datasets validate the superiority of both VBFM and OVBFM.

- Nonparametric Poisson Factorization Machine (NPFM): NPFM models count data using the Poisson distribution which provides both modeling and computational advantages for sparse data. The specific advantages of NPFM include:
  - NPFM provides a more interpretable model with a better fit for count dataset.
  - NPFM is a nonparametric approach and avoids the costly model selection procedure by automatically finding the number of latent factors suitable for modeling the pairwise interaction matrix in FM.
  - NPFM takes advantages of the Poisson distribution [Gopalan *et al.*, 2015] and considers only sampling over the non-zero entries. On the other hand, existing FM methods, which assume a Gaussian distribution, must iterate over both

positive and negative samples in the implicit setting. Such iteration is expensive for large datasets and often needs to be solved using a costly positive and negative data sampling approach. NPFM can take advantage of natural sparsity of the data which existing inference technique of FM fails to exploit.

We also consider a special case of NPFM, the Parametric Poisson Factorization Machine (PPFM), that considers a fixed number of latent factors. Both PPFM and NPFM have linear time and space complexity with respect to the number of non-zero observations. Extensive experiments on four different movie review datasets show that our methods outperform two strong baseline methods by large margins.

1.2 Outline of the Thesis

Rest of this thesis is organized as follows:

- Chapter 2 reviews the necessary background work.
- Chapter 3 describes the model and experimental evaluation of SBMF.
- Chapter 4 describes the VBFM and OVBFM and shows their empirical validation.
- Chapter 5 presents the NPFM which can theoretically deal with infinite number of latent factors and evaluation of NPFM on both synthetic and real world dataset. It also analyses a special case of NPFM, the PPFM, that considers a fixed number of latent factors.
- Chapter 6 concludes and explains possible directions for future works.

CHAPTER 2

BACKGROUND

In this chapter, we explain various background works which will be helpful to understand the thesis. We start by discussing Recommender Systems (RSs), followed by latent variable models and factorization models. Then we describe a generic probabilistic framework, and using this framework we explain some of the existing Bayesian approximate inference techniques which will be used throughout the thesis.

2.1 Recommender Systems

Recommender Systems (RSs) are software tools and techniques which provide suggestions of appropriate items to users on various decision making processes such as: what song to listen, what book to buy, what movie to watch, etc. But, the appropriate set of items are relative to the individuals. Hence, RSs are often built for personalized recommendation and provide user specific suggestions.

Broadly the RS algorithms can be classified into three categories: 1) collaborative filtering (CF); 2) content-based; and 3) knowledge-based. Combination of these algorithms leads to hybrid algorithms. We provide a brief description of these different types of RS algorithms in this section.

2.1.1 Collaborative Filtering

Collaborative filtering (CF) [Su and Khoshgoftaar, 2009; Lee *et al.*, 2012; Bell and Koren, 2007] is a popular and successful approach to RSs which considers multiple users' preferences into account to recommend items to a user. The fundamental assumption behind CF is that if two users' preferences are same on some number of items, then their behavior will

be same on other unseen items. In a typical CF scenario, there is a set of users $\{1, 2, ..., I\}$ and a set of items $\{1, 2, ..., J\}$, and each user i has provided preferences/ratings to some number items. This data can be represented as a matrix $\boldsymbol{R} \in \mathbb{R}^{I \times J}$, where r_{ij} is the preference/rating by the i^{th} user to the j^{th} item. The task of the CF algorithm is to recommend unseen items to users. So CF problems can be viewed as missing value estimation task: estimate the missing values of the matrix $\boldsymbol{R}$. CF algorithms are generally classified into two categories: memory-based; and model-based.

Memory-based

Memory-based CF algorithms are lazy learners. Neighborhood-based (kNN) CF algorithms [Su and Khoshgoftaar, 2009; Bell and Koren, 2007] are most common form of memory-based CF techniques. Earlier, kNN algorithms were mostly user-based approach [Bell and Koren, 2007]. User-based approach estimates the unknown rating of an item for a user based on the ratings of similar users to that item. Formally, to estimate the rating r_{ij}, we consider a set of users $N(i, j)$, whose rating behavior are similar to user i, and have rated item j. Here similarity is defined in terms of how two user's preference behavior match to each other. Then, the estimation of rating r_{ij} is calculated as follows:

$$\hat{r}_{ij} = \bar{r}_i + \frac{\sum\limits_{i' \in N(i,j)} (r_{i'j} - \bar{r}_{i'}) w_{ii'}}{\sum\limits_{i' \in N(i,j)} |w_{ii'}|}, \tag{2.1}$$

where $\bar{r}_i$ and $\bar{r}_{i'}$ are the average ratings score of user i and user $i^{'}$ respectively, and $w_{ii'}$ is the similarity score between user i and user $i^{'}$. The similarity measure plays an important role, as they are both used to select the members of $N(i, j)$ and as well as in Eq. (2.1). Common choices of the similarity measures are Pearson correlation coefficient and cosine similarity. For the user-based algorithm, the Pearson correlation between user i and user $i^{'}$ is calculated as follows:

$$w_{ii'} = \frac{\sum\limits_{j \in J'} (r_{ij} - \bar{r}_i)(r_{i'j} - \bar{r}_{i'})}{\sqrt{\sum\limits_{j \in J'} (r_{ij} - \bar{r}_i)^2} \sqrt{\sum\limits_{j \in J'} (r_{i'j} - \bar{r}_{i'})^2}}, \tag{2.2}$$

where $J^{'}$ is the set of items rated by both user i and user $i^{'}$.

An alternative to user-based approach is item-based approach [Bell and Koren, 2007]. In this method, to predict an unknown rating r_{ij}, we identify a set of items $N(j,i)$ that has rating behavior similar to item j, and user i has rated all the items in the set $N(j,i)$. Then the prediction is done as follows:

$$\hat{r}_{ij} = \frac{\sum\limits_{j' \in N(j,i)} r_{ij'} w_{jj'}}{\sum\limits_{j' \in N(j,i)} |w_{jj'}|}, \tag{2.3}$$

where $w_{jj'}$ is the similarity score between item j and item $j^{'}$. The similarity score $w_{jj'}$ for item-based approach can be calculated using the Pearson correlation coefficient as follows:

$$w_{jj'} = \frac{\sum\limits_{i \in I'} (r_{ij} - \bar{r}_j)(r_{ij'} - \bar{r}_{j'})}{\sqrt{\sum\limits_{i \in I'} (r_{ij} - \bar{r}_j)^2} \sqrt{\sum\limits_{i \in I'} (r_{ij'} - \bar{r}_{j'})^2}}, \tag{2.4}$$

where $I^{'}$ is the set of users who have rated both item j and $j^{'}$, and $\bar{r}_j$ and $\bar{r}_{j'}$ are the average rating of item j and $j^{'}$ respectively.

Model-based

Though memory-based CF algorithms scale to large data, they are unreliable when the data are sparse [Su and Khoshgoftaar, 2009]. In order to achieve better accuracy and alleviate the short comings of memory-based CF, model-based CF algorithms [Koren *et al.*, 2009] have been investigated. Model-based CF fits a parametric model to the training data, which later is used to predict the unknown user-item ratings. Model-based methods include cluster-based CF [Connor and Herlocker, 2001; Xue *et al.*, 2005] and Bayesian classifiers [Miyahara and Pazzani, 2000]. One widely used and successful type of model-based CF algorithms are based on latent variable models such as, pLSA [Hofmann, 2004], neural networks [Salakhutdinov *et al.*, 2007], Latent Dirichlet Allocation [Blei *et al.*, 2003], and matrix factorization [Salakhutdinov and Mnih, 2007], which try to uncover hidden features that explain the ratings. Latent variable models involve supplementing a set of observed variables with additional latent, or hidden, variables. In probabilistic latent variable model

framework, the distribution over the observed variables is obtained by marginalizing out the hidden variables from the joint distribution over observed and hidden variables. The hidden structure of the data can be computed and explored through the posterior which is defined as the conditional distribution of the latent variables given the observations. Factorization models are a widely used model-based CF algorithms which fall under latent variable modeling. We will provide detailed discussion on latent variable models and factorization models in Section 2.2.

2.1.2 Content-based

Besides CF, content-based methods [Su and Khoshgoftaar, 2009] are another important class of RS algorithms. Content-based methods make recommendations by analyzing domain knowledge of users and/or items. Typically, content-based methods first extract features of users and/or items and then apply a classification based algorithm to provide recommendations. Unlike CF, content-based methods are limited to the feature information.

2.1.3 Knowledge-based

Knowledge-based methods [Burke, 2000] use knowledge about users and context, and apply knowledge based techniques to provide recommendation. In knowledge-based methods, users are an integral part of the knowledge discovery process. It uses a knowledge base and develop this base using the user feedback. Knowledge-based methods are particularly helpful when the user wants to give explicit feedback to the system and available user-item interaction data is very less.

2.1.4 Hybrid

There are pros and cons with each type of recommendation algorithms. CF algorithms need a large amount of rating data to provide any useful suggestions. Until there is a large number of users whose habits are known, the system cannot be useful for most users. Also

until a sufficient number of rated items has been collected, the system cannot be useful for a particular user. This problem is known as cold start problem. Similar to CF, content-based methods also suffer from the cold start problem. However, CF and content-based methods are scalable and useful for large datasets. On the other hand, knowledge-based methods avoid cold start problem through user feedback, but suffer from the the problem of costly knowledge base creation process, and hence, less scalable. Hybrid recommender [Burke, 2002] systems combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. Most commonly, CF is combined with some other techniques. Weighted is a hybridization technique which predicts the score of an item using weighted combination of the prediction from different recommender algorithms. Another approach is switching between different algorithms using some criterion.

2.2 Latent Variable Models and Factorization Models

A powerful approach to probabilistic modeling involves describing a set of observed variables using a set of latent/hidden variables which is known as latent variable modeling. Latent variable models are widely used in several domains such as machine learning, statistics, data mining. Latent variable models uncover hidden structure which explains the data. In latent variable models, each observation is associated with a/(a set of) latent variable/variables, therefore, the number of latent variables grows with the size of the data, whereas the number of parameters in a model is usually fixed irrespective of the data size (if latent variables are not part of the parameter set). Latent variable models consider a joint distribution over the hidden and observed variables. Often we seek to find the posterior distribution over the hidden variables given the observed variables. Latent Dirichlet Allocation [Blei *et al.*, 2003] is a well-known example of a latent variable model.

One of the widely used latent variable models in the RSs community are factorization models, due to their simplicity, prediction quality and scalability. The idea behind such models is that preferences of a user are determined by a small number of unobserved latent factors. Matrix factorization [Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007,

2008; Gopalan *et al.*, 2015] is the simplest and most well studied factorization model and has been applied in solving numerous problems related to analysis of dyadic data, such as in RSs [Koren *et al.*, 2009], topic modeling [Arora *et al.*, 2012], and network analysis [Zhou, 2015]. Affinity data between two entities are known as dyadic data. An example of a dyadic data is movie recommendation, where pair of entities involved are user and movie and the affinity response is the rating provided by a user to a movie. Tensor factorization [Xiong *et al.*, 2010; Ho *et al.*, 2014; Chi and Kolda, 2012] is an extension of matrix factorization where the data is represented as three dimensional array, signifying interactions of three different variables.

Many specialized factorization models have further been proposed to deal with non-categorical variables. For example, SVD++ [Koren, 2008] uses neighborhood of a user for analysis of movie rating data, TimeSVD++ [Koren, 2009] and Bayesian Probabilistic Tensor Factorization (BPTF) [Xiong *et al.*, 2010] discretize time which is a continuous variable, and factorizing personalized Markov chains (FPMC) [Rendle *et al.*, 2011*a*] considers the purchase history of users to recommend items. Numerous learning techniques have also been proposed for factorization models which include stochastic gradient descent (SGD) [Koren *et al.*, 2009], alternating least squares [Zhou *et al.*, 2008], variational Bayes [Lim and Teh, 2007; Kim and Choi, 2014; Silva and Carin, 2012], and Markov chain Monte Carlo (MCMC) based inference [Salakhutdinov and Mnih, 2008].

In this section, we explain some of the important factorization models which will be used throughout the thesis.

2.2.1 Matrix Factorization

Formally, matrix factorization recovers a low-rank latent structure of a matrix by approximating it as a product of two low-rank matrices. For delineation, consider a user-movie matrix $\boldsymbol{R} \in \mathbb{R}^{I \times J}$ where r_{ij} cell represents the rating provided to movie j by user i. Matrix factorization decomposes the Matrix $\boldsymbol{R}$ into two low-rank matrices $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_I]^T \in \mathbb{R}^{I \times K}$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_J]^T \in \mathbb{R}^{J \times K}$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are the K dimensional latent

factor vectors associated with user i and item j, such that:

$$\boldsymbol{R} \sim \boldsymbol{U}\boldsymbol{V}^{\intercal}. \tag{2.5}$$

This matrix factorization model is closely related to singular value decomposition, a well-established technique for identifying latent semantic factors in information retrieval. But applying singular value decomposition with incomplete matrix is undefined and which is often the case in CF. Also addressing only the relatively few known ratings creates overfitting issue. Earlier methods used imputation to fill the missing entries of the matrix, making the matrix dense. However, these methods are not scalable. Hence recent methods [Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007] model only the observed ratings while avoiding overfitting through regularization. Typically, the latent factors are learned by minimizing a regularized squared error loss function, which is defined as:

$$\sum_{(i,j)\in\Omega} \left(r_{ij} - \boldsymbol{u}_i^{\intercal}\boldsymbol{v}_j\right)^2 + \lambda\left(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2\right), \tag{2.6}$$

where Ω is the set of all the observed ratings, λ is the regularization parameter and $||\boldsymbol{X}||_F^2$ is the Frobenius norm of $\boldsymbol{X}$.

Learning

The optimization problem in Eq. (2.6) can be solved using stochastic gradient descent (SGD) [Koren *et al.*, 2009]. In SGD, for each given rating r_{ij}, the update equation of user and item latent factor vectors can be written as follows:

$$\boldsymbol{u_i} \leftarrow \boldsymbol{u_i} + \eta\left(2\left(r_{ij} - \boldsymbol{u_i}^T\boldsymbol{v_j}\right)\boldsymbol{v_j} - 2\lambda\boldsymbol{u_i}\right), \tag{2.7}$$

$$\boldsymbol{v_j} \leftarrow \boldsymbol{v_j} + \eta\left(2\left(r_{ij} - \boldsymbol{u_i}^T\boldsymbol{v_j}\right)\boldsymbol{u_i} - 2\lambda\boldsymbol{v_j}\right), \tag{2.8}$$

where η is the learning rate.

2.2.2 Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2007] provides a probabilistic interpretation of matrix factorization. In PMF, factor variables are assumed to be marginally independent whereas rating variables given the factor variables are assumed to be conditionally independent. PMF considers the conditional distribution of the rating variables (the likelihood term) as:

$$p(\boldsymbol{R}|\boldsymbol{U}, \boldsymbol{V}, \tau) = \prod_{(i,j)\in\Omega} \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\intercal \boldsymbol{v}_j, \tau^{-1}), \tag{2.9}$$

where τ is the precision parameter. Zero-mean spherical Gaussian priors are placed on user and movie latent factor vectors as follows:

$$\boldsymbol{U} \sim \prod_{i=1}^{I} \mathcal{N}(\boldsymbol{u}_i|0, \lambda_u^{-1}\boldsymbol{I}), \tag{2.10}$$

$$\boldsymbol{V} \sim \prod_{j=1}^{J} \mathcal{N}(\boldsymbol{v}_j|0, \lambda_v^{-1}\boldsymbol{I}), \tag{2.11}$$

where λ_u and λ_v are hyperparameters and $\boldsymbol{I}$ is the identity matrix.

The main drawback of this model is that inferring the posterior distribution over the latent factors, given the ratings, is intractable. PMF handles this intractability by providing a maximum a posteriori estimation of the model parameters by maximizing the log-posterior over the model parameters, which is equivalent to minimizing Eq. (2.6). So learning PMF model with fixed hyperparameters is equivalent to minimizing Eq. (2.6) which can be done using SGD.

2.2.3 Bayesian Probabilistic Matrix Factorization

Though PMF can be learned using SGD, it suffers from the problem of manual tuning of learning rate and the regularization parameters, and it often overfits the data. One way to solve the problem of manual tuning of regularization parameters is to introduce priors on the hyperparameters and maximize the log-posterior of the model over both parameters

and hyperparameters. Though this method allows automatic model parameter selection, it is not theoretically sound [Salakhutdinov and Mnih, 2008]. So, fully Bayesian Probabilistic Matrix Factorization (BPMF) [Salakhutdinov and Mnih, 2008] has been developed which is robust to the overfitting and avoids model selection problem. BPMF considers the likelihood function as in Eq. (2.9) similar to PMF. The prior over user and item latent factor vectors are assumed as:

$$\boldsymbol{U} \sim \prod_{i=1}^{I} \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_u, \boldsymbol{\Lambda}_u^{-1}), \tag{2.12}$$

$$\boldsymbol{V} \sim \prod_{j=1}^{J} \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_v, \boldsymbol{\Lambda}_v^{-1}), \tag{2.13}$$

where $\boldsymbol{\mu}_u, \boldsymbol{\Lambda}_u, \boldsymbol{\mu}_v$, and $\boldsymbol{\Lambda}_v$ are hyperparameters. As Gaussian-Wishart is the conjugate prior of a multivariate Gaussian distribution with unknown mean and precision matrix, Gaussian-Wishart priors are placed on hyperparameters as:

$$\boldsymbol{\mu}_u, \boldsymbol{\Lambda}_u \sim \mathcal{N}(\boldsymbol{\mu}_u | \boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_u)^{-1}) W(\boldsymbol{\Lambda}_u | \boldsymbol{W}_0, \nu_0), \tag{2.14}$$

$$\boldsymbol{\mu}_v, \boldsymbol{\Lambda}_v \sim \mathcal{N}(\boldsymbol{\mu}_v | \boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_v)^{-1}) W(\boldsymbol{\Lambda}_v | \boldsymbol{W}_0, \nu_0), \tag{2.15}$$

where $\boldsymbol{\mu}_0, \beta_0, \boldsymbol{W}_0$, and ν_0 are hyperprior parameters.

Note that a complete conditional is the conditional distribution of a variable given the observations and all other variables in the model, and a conditionally conjugate model is one where each complete conditional has a close distributional form [Hoffman *et al.*, 2013]. As the BPMF model is conditionally conjugate, learning is done using a closed form Gibbs sampling inference mechanism [Salakhutdinov and Mnih, 2008]. Prediction for a rating r_{ij} is done as follows:

$$\hat{r}_{ij} = \frac{1}{C} \sum_{c=1}^{C} (\boldsymbol{u}_i^c)^\intercal \boldsymbol{v}_j^c, \tag{2.16}$$

where $\boldsymbol{u}_i^c$ and $\boldsymbol{v}_j^c$ are the c^{th} drawn samples for the i^{th} user and the j^{th} item latent factor vectors respectively and C is the number of drawn samples from the Gibbs sampling process. We will describe Gibbs sampling in more detail in Section 2.4.

2.2.4 Factorization Machine

Factorization Machine (FM) combines the advantages of feature based methods like Support Vector Machines (SVMs) with factorization models. Popular feature based methods like SVMs can be learnt using standard tools like LIBSVM [Chang and Lin, 2011] and SVM-Light [Joachims, 2002]. But feature based methods encounter problems when facing high-dimensional but sparse dyadic data where factorization models have been more successful. An FM learns a function $f : \mathbb{R}^D \rightarrow T$ which is a mapping from a real valued feature vector $\boldsymbol{x} \in \mathbb{R}^D$ to a target domain T. The training data for FM consists of N tuples of the form $(\boldsymbol{x}_n, y_n)_{n=1}^N$, where $\boldsymbol{x}_n$ is the feature representation (row vector) and y_n is the associated response variable for the n^{th} training instance respectively. We will denote $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ and $\boldsymbol{X} = (\boldsymbol{x}_1^\intercal, \boldsymbol{x}_2^\intercal, ..., \boldsymbol{x}_n^\intercal)^\intercal$.

Example

Consider a song-count dataset where the response variable is the number of times a user has listened to a particular song, which has an associated genre. Let us denote user, song, and genre by $u, s,$ and g respectively. If the training data is composed of the set of points $\{(u_1, s_1, g_1, 10), (u_1, s_3, g_2, 33), (u_2, s_2, g_3, 19), (u_3, s_1, g_1, 21)\}$, then Figure 2.1 shows the corresponding feature representation of FM where the i^{th} column indicates the data corresponding to the i^{th} variable and the n^{th} row represents the n^{th} training instance. Given the feature representation of Figure 2.1, the model equation for FM for the n^{th} training instance is:

$$\hat{y}_n = w_0 + \sum_{i=1}^{D} x_{ni} w_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni} x_{nj} \boldsymbol{v}_i^\intercal \boldsymbol{v}_j, \tag{2.17}$$

where w_0 is the global bias, w_i is the bias associated with the i^{th} variable, and $\boldsymbol{v}_i$ is the latent factor vector of dimension K associated with the i^{th} variable. $\boldsymbol{v}_i^\intercal \boldsymbol{v}_j$ models the interaction between the i^{th} and j^{th} features. The objective is to estimate the parameters $w_0 \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^D$, and $\boldsymbol{V} \in \mathbb{R}^{D \times K}$. Instead of using a parameter $w_{i,j} \in \mathbb{R}$ for each interaction, FM models the pairwise interaction by factorizing it. Since for any positive definite matrix $\boldsymbol{W}$, there exists a matrix $\boldsymbol{V}$ such that $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{V}^T$ provided K is sufficiently large, FM can express any interaction matrix $\boldsymbol{W}$. This is a remarkably smart way to express pairwise

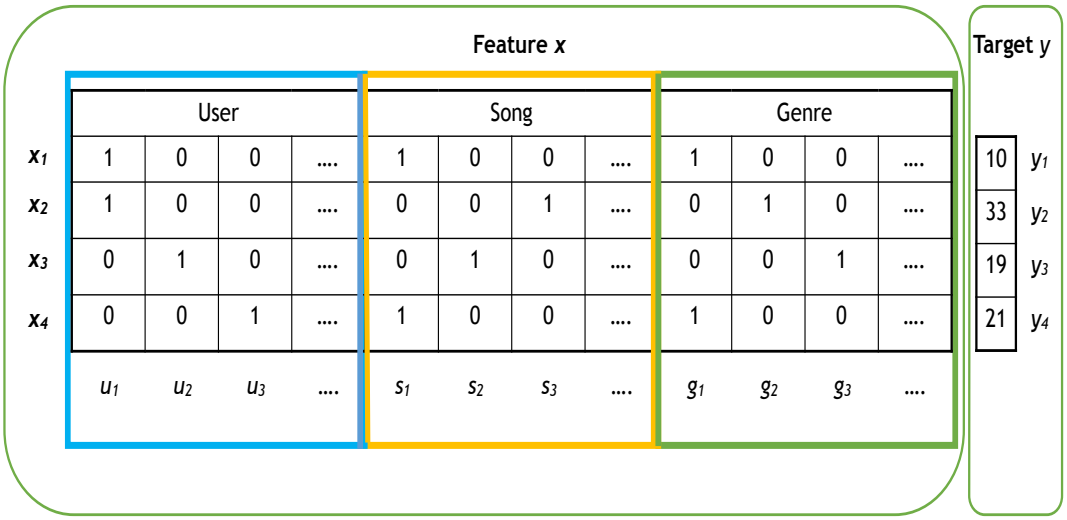


Figure 2.1: Feature representation of FM for a song-count dataset with three types of variables: user, song, and genre. An example of four data instances are shown. Row 1 shows a data instance where user u_1 listens to song s_1 of genre g_1 10 times.

interaction in big sparse datasets. In fact, many of the existing CF algorithms aim to do the same, but with the specific goal of user-item recommendation and thus fail to recognize the underlying mathematical basis that FM successfully discovers.

Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [Koren *et al.*, 2009], SVD++ [Koren, 2008], TimeSVD++ [Koren, 2009], Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme, 2010], and factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2011*a*], and has also been used for context aware recommendation [Rendle *et al.*, 2011*b*]. Other advantages of FM include – 1) FM allows parameter estimation with extremely sparse data where SVMs fail; 2) FM has linear complexity, can be optimized in the primal and, unlike SVMs, does not rely on support vectors; and 3) FM is a general predictor that can work with any real valued feature vector, while several state-of-the-art factorization models work only on very restricted input data.

2.2.5 Learning Factorization Machine

Three learning methods have been proposed for Factorization Machine (FM): 1) stochastic gradient descent (SGD) [Rendle, 2010], 2) alternating least squares (ALS) [Rendle *et al.*, 2011*b*], and 3) Markov chain Monte Carlo (MCMC) [C. Freudenthaler, 2011] inference. Here, we will briefly describe SGD and MCMC learning for FM.

Optimization Task

Optimization function for FM with L2 regularization can be written as follows:

$$\sum_{(\boldsymbol{x}_n, y_n) \in \Omega} l(\hat{y}_n, y) + \sum_{\theta \in \boldsymbol{\theta}} \lambda_\theta \theta^2, \tag{2.18}$$

where Ω is the training set, $\boldsymbol{\theta} = \{w_0, \boldsymbol{w}, \boldsymbol{V}\}$, λ_θ is the regularization parameter, and l is the loss function. For binary observations, the loss function is assumed to be a sigmoid and for other cases, it is assumed to be a square loss.

Probabilistic Interpretation

Both loss and regularization can be motivated from a probabilistic point of view. For least squares loss, the target y follows a Gaussian distribution.

$$y_n \sim \mathcal{N}(\hat{y}_n, \alpha^{-1}), \tag{2.19}$$

where α is the precision parameter. For binary classification, y follows a Bernoulli distribution.

$$y_n \sim \text{Bernoulli}(b(\hat{y}_n)), \tag{2.20}$$

where b is a link function. L2 regularization corresponds to Gaussian prior on the model parameters.

$$\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^{-1}), \tag{2.21}$$

where μ_θ and σ_θ are hyperparameters.

Stochastic Gradient Descent

One of the more popular learning algorithms for FM is based on stochastic gradient descent (SGD). FM that uses SGD for learning is conveniently addressed as SGD-FM. Algorithm 1 describes the SGD-FM algorithm. SGD-FM requires costly search over the parameter space to find the best values for the learning rate and the regularization parameters. To mitigate such expensive tuning of parameters, learning algorithm based on ALS have also been proposed to automatically select the learning rate.

Algorithm 1 Stochastic Gradient Descent for Factorization Machine (SGD-FM)

Require: Training data Ω, regularization parameters λ, learning rate η, initialization σ.
Ensure: $w_0 \leftarrow 0, \boldsymbol{w} \leftarrow (0, ..., 0), v_{ik} \sim \mathcal{N}(0, \sigma^{-1})$.

repeat
  for $(\boldsymbol{x}_n, y_n) \in \Omega$ **do**
    $w_0 \leftarrow w_0 - \eta \left(\frac{\partial}{\partial w_0} l(\hat{y}_n, y_n) + 2\lambda_0 w_0 \right)$
    for $i = 1$ to $D \wedge x_{ni} \neq 0$ **do**
      $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} l(\hat{y}_n, y_n) + 2\lambda_i w_i \right)$
      for $k = 1$ to K **do**
        $v_{ik} \leftarrow v_{ik} - \eta \left(\frac{\partial}{\partial v_{ik}} l(\hat{y}_n, y_n) + 2\lambda_{ik} v_{ik} \right)$
      end for
    end for
  end for
until convergence

Markov Chain Monte Carlo

As an alternative, Markov chain Monte Carlo (MCMC) based Gibbs sampling inference has been proposed for FM. FM that uses MCMC for learning is conveniently addressed as MCMC-FM. MCMC-FM is a generative approach. In MCMC-FM, tuning of parameters is less of a concern, yet it produces state-of-the-art performance for several applications. MCMC-FM considers the conditional distribution of the rating variables (the likelihood term) as:

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \alpha) = \prod_{(\boldsymbol{x}_n, y_n) \in \Omega} \mathcal{N}(y_n|\hat{y}_n, \alpha^{-1}) \tag{2.22}$$

MCMC-FM assumes priors on the model parameters as follows:

$$w_0 \sim \mathcal{N}(w_0|\mu_0, \sigma_0^{-1}), \tag{2.23}$$

$$w_i \sim \mathcal{N}(w_i|\mu_i, \sigma_i^{-1}), \tag{2.24}$$

$$v_{ik} \sim \mathcal{N}(v_{ik}|\mu_{ik}, \sigma_{ik}^{-1}), \tag{2.25}$$

$$\alpha \sim \text{Gamma}(\alpha|\alpha_0, \beta_0). \tag{2.26}$$

On each pair of hyperparameters (μ_i, σ_i) and (μ_{ik}, σ_{ik}) $\forall i, k$ a Gaussian distribution is placed on μ and a gamma distribution is placed on σ as follows:

$$\mu_i \sim \mathcal{N}(\mu_i|\mu_0, (\nu_0\sigma_i)^{-1}), \tag{2.27}$$

$$\sigma_i \sim \text{Gamma}(\sigma_i|\alpha_0^{'}, \beta_0^{'}), \tag{2.28}$$

$$\mu_{ik} \sim \mathcal{N}(\mu_{ik}|\mu_0, (\nu_0\sigma_{ik})^{-1}), \tag{2.29}$$

$$\sigma_{ik} \sim \text{Gamma}(\sigma_{ik}|\alpha_0^{'}, \beta_0^{'}), \tag{2.30}$$

where $\mu_0, \nu_0, \alpha_0^{'}$, and $\beta_0^{'}$ are hyperprior parameters.

MCMC-FM is a closed form Gibbs sampling inference algorithm for the above model. Please refer to [C. Freudenthaler, 2011] for more detailed analysis on MCMC-FM inference equations.

2.3 Probabilistic Modeling and Bayesian Inference

Here we will explain a general probabilistic framework, by which we will describe some of the existing approximate inference techniques. Assume $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) \in \mathbb{R}^{D\times N}$ are the observations and $\boldsymbol{\theta}$ is the set of unknown parameters for the model that generates $\boldsymbol{X}$. For example, assuming $\boldsymbol{X}$ is generated by a Gaussian distribution, $\boldsymbol{\theta}$ would be the mean and the variance of that Gaussian distribution. One of the most popular approaches for parameter estimation is maximum likelihood. In maximum likelihood, the parameters are estimated as:

$$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{X}|\boldsymbol{\theta}) \tag{2.31}$$

The generative model may also include latent or hidden variables. We will denote hidden variables by $\boldsymbol{Z}$. These random variables act as links, that connect the observations to the unknown parameters, and help to explain the data. Given this set up, we aim to find the posterior distribution of latent variables which is written as follows:

$$p\left(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}\right) = \frac{p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\theta})p(\boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{X}|\boldsymbol{\theta})} \tag{2.32}$$

$$= \frac{p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\theta})p(\boldsymbol{Z}|\boldsymbol{\theta})}{\int_{\boldsymbol{Z}} p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\theta})p(\boldsymbol{Z}|\boldsymbol{\theta})d\boldsymbol{Z}} \tag{2.33}$$

But often the denominator in Eq. (2.33) is intractable and hence we need to resort to some approximate inference techniques to calculate the posterior distribution approximately. We explain two popular approximate inference techniques in details below which are used in this thesis.

2.4 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods [Metropolis and Ulam, 1949; Hastings, 1970] are established tools for solving intractable integration problems central to Bayesian statistics. MCMC method was first proposed by Metropolis and Ulam in 1949 [Metropolis and Ulam, 1949] and then generalized to Metropolis-Hastings method [Hastings, 1970]. MCMC method constructs a Markov chain with state space $\boldsymbol{Z}$ and $p\left(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}\right)$ as stationary distribution to sample from $p\left(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}\right)$. The simulated values can be considered as coming from the target distribution if the chain is run for long time. A Markov chain is generated by sampling for a new state of the chain depending on the present state of the chain, ignoring all the past states.

2.4.1 Gibbs Sampling

Gibbs sampling [Geman and Geman, 1984; Gelfand and Smith, 1990] is the most widely applied MCMC method. Gibbs sampling is a powerful tool when we cannot sample directly from the joint-posterior distribution, but when sampling from the conditional distri-

butions of each variable, or set of variables, is possible. Gibbs sampling aims to generate samples from the posterior distribution of $\boldsymbol{Z}$ that is partitioned into M disjoint components $\boldsymbol{Z} = (\boldsymbol{Z}_1, \boldsymbol{Z}_2, ..., \boldsymbol{Z}_M)$. Although it may be hard to sample from the joint-posterior, it is assumed that it is easy to sample from the full conditional distribution of $\boldsymbol{Z}_i$. Initially all the parameters are initialized by random values and then the sampling process goes as follows:

$$
\begin{aligned}
\boldsymbol{Z}_1^{(t+1)}|- &\sim p\left(\boldsymbol{Z}_1|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{Z}_2^{(t)}, \boldsymbol{Z}_3^{(t)}, ..., \boldsymbol{Z}_M^{(t)}\right) \\
\boldsymbol{Z}_2^{(t+1)}|- &\sim p\left(\boldsymbol{Z}_2|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{Z}_1^{(t+1)}, \boldsymbol{Z}_3^{(t)}, ..., \boldsymbol{Z}_M^{(t)}\right) \\
\boldsymbol{Z}_3^{(t+1)}|- &\sim p\left(\boldsymbol{Z}_3|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{Z}_1^{(t+1)}, \boldsymbol{Z}_2^{(t+1)}, \boldsymbol{Z}_4^{(t)}, ..., \boldsymbol{Z}_M^{(t)}\right) \\
&\vdots
\end{aligned}
$$

where $\boldsymbol{Z}_i^t$ is the sample drawn for the i^{th} component in the t^{th} iteration. For more detailed discussion on MCMC methods look into [Metropolis and Ulam, 1949].

2.5 Variational Bayes

Variational methods have their origins on the calculus of variations. Unlike standard calculus, variational calculus considers functional which is defined as a mapping which takes as input a function and output the value of the functional. The functional derivative is defined as the change of the functional for the small change in the input function. Variational inference, an alternative to MCMC sampling, transforms a complex inference problem into a high-dimensional optimization problem [Beal, 2003; Tzikas *et al.*, 2008; Hoffman *et al.*, 2013]. It explores all possible input functions to find the one that maximizes, or minimizes, the functional.

Variational inference optimizes the marginal likelihood function. For our framework, the marginal likelihood function can be written as follows:

$$
\ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q||p), \tag{2.34}
$$

where,

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int_{\boldsymbol{Z}} q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{Z}, \boldsymbol{X}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} \right) d\boldsymbol{Z}, \tag{2.35}$$

$$KL(q||p) = -\int_{\boldsymbol{Z}} q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})} \right) d\boldsymbol{Z}, \tag{2.36}$$

where $q(\boldsymbol{Z})$ is any probability distribution. $KL(q||p)$ is the Kullback-Leibler divergence between $q(\boldsymbol{Z})$ and $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})$, and is always non-negative. Thus $\ln p(\boldsymbol{X}|\boldsymbol{\theta})$ is lower-bounded by the term $\mathcal{L}(q, \boldsymbol{\theta})$, also known as evidence lower bound (ELBO) [Beal, 2003; Tzikas *et al.*, 2008]. We can maximize the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ by optimization with respect to the distribution $q(\boldsymbol{Z})$, which is equivalent to minimizing the KL divergence. Maximum of the lower bound occurs when KL divergence vanishes, which occurs when $q(\boldsymbol{Z})$ equal to $p(\boldsymbol{Z}|\boldsymbol{X}, \theta)$. However, in practice working with the true posterior distribution is intractable. Therefore, a restrictive form of $q(\boldsymbol{Z})$ is considered, and then a member of this family is found which minimizes the KL divergence. Typically, a factorized distribution is considered of the hidden variables $\boldsymbol{Z}$ which factorizes into M disjoint partitions. Also it is assumed that $q(\boldsymbol{Z})$ factorizes with respect to these partitions as follows:

$$q(\boldsymbol{Z}) = \prod_{i=1}^{M} q_i(\boldsymbol{Z}_i) \tag{2.37}$$

Among all distributions $q(\boldsymbol{Z})$ with the form Eq. (2.37), we want to find the distribution for which the lower bound is largest. A free form optimization is performed of $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to all of the distributions $q_i(\boldsymbol{Z}_i)$, which is done by each factors in turn. Let us consider $q_j(\boldsymbol{Z}_j)$ as q_j and using Eq. (2.37) lower bound can be written as:

$$\mathcal{L}(q, \boldsymbol{\theta}) = KL(\tilde{p}||q_j) - \sum_{i \neq j} \int q_i \ln q_i d\boldsymbol{Z}_i, \tag{2.38}$$

where,

$$\ln \tilde{p}(\boldsymbol{X}, \boldsymbol{Z}_j|\boldsymbol{\theta}) = \mathbb{E}_{i \neq j} \left[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) \right] + \text{const} \tag{2.39}$$

Now keeping $\{q_{i \neq j}\}$ fixed and maximization of lower bound with respect to all possible distributions of $q_j(\boldsymbol{Z}_j)$ is equivalent to minimizing KL divergence in Eq. (2.38). So we

get:

$$\ln q_j^*(\boldsymbol{Z}_j) = \ln \tilde{p}(\boldsymbol{X}, \boldsymbol{Z}_j|\boldsymbol{\theta}) = \mathbb{E}_{i\neq j}\left[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})\right] + \text{const} \tag{2.40}$$

The constant term can be calculated using normalization.

$$q_j^*\left(\boldsymbol{Z}_j\right) = \frac{\exp\left(\mathbb{E}_{i\neq j}\left[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})\right]\right)}{\int \exp\left(\mathbb{E}_{i\neq j}\left[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})\right]\right) d\boldsymbol{Z}_j} \tag{2.41}$$

In summary, the variational EM algorithm can be written as:
E-Step: Evaluate $q^{\text{NEW}}(\boldsymbol{Z})$ to maximize $\mathcal{L}(q, \boldsymbol{\theta})$ solving the system of Eq. (2.41).
M-Step: Find $\boldsymbol{\theta}^{\text{NEW}} = \text{argmax}_{\boldsymbol{\theta}}\mathcal{L}(q, \boldsymbol{\theta})$.

Note, that here we do not consider the form of $q_j(\boldsymbol{Z}_j)$, they are found automatically. Also, the lower bound provides another way to approach the variational inference. If the functional form of the factors in the variational posterior distribution are known, then by taking general parametric form of these distributions, we can write the lower bound. Then we can maximize the lower bound with respect to these parameters by setting the derivative of lower bound with respect to these parameters to zero, which gives the re-estimation equations. For more detailed and introductory discussion on variational Bayes look into [Bishop, 2006; Tzikas *et al.*, 2008; Beal, 2003].

2.5.1 Stochastic Variational Inference

Recently stochastic variational inference has been applied in many places such as, stochastic variational inference (SVI) in matrix factorization [Hernández-Lobato *et al.*, 2014], topic modeling [Hoffman *et al.*, 2013] and network modeling [Gopalan *et al.*, 2012]. Specifically, SVI samples a complete data instance, such as a document, and updates all the model parameters. Often, to reduce the variance instead of sampling a single data point to update the parameters, a batch of points are sampled and then the variational parameters associated to this batch of data points are updated. This version of variational inference is called as mini-batch variational inference [Hoffman *et al.*, 2013].

CHAPTER 3

SCALABLE BAYESIAN MATRIX FACTORIZATION

Bayesian Probabilistic Matrix Factorization (BPMF), which is a Markov chain Monte Carlo (MCMC) based Gibbs sampling inference algorithm for matrix factorization, provides state-of-the-art performance. BPMF uses multivariate Gaussian prior on latent factor vector which leads to cubic time complexity with respect to the dimension of latent space. To avoid this cubic time complexity, in this chapter, we develop the Scalable Bayesian Matrix Factorization (SBMF) which considers independent univariate Gaussian prior over latent factors. We also incorporate bias terms in the model which are missing in baseline BPMF model. Similar to BPMF, SBMF is a MCMC based Gibbs sampling inference algorithm for matrix factorization. SBMF has linear time complexity with respect to the dimension of latent space and linear space complexity with respect to the number of non-zero observations.

3.1 Introduction

Factor based models have been used extensively in Recommender Systems (RSs). In a factor based model, preferences of each user are represented by a small number of unobserved latent factors. Matrix factorization [Srebro and Jaakkola, 2003; Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007, 2008; Gopalan *et al.*, 2015] is the simplest and most well studied factor based model and has been applied successfully in several domains. Formally, matrix factorization recovers a low-rank latent structure of a matrix by approximating it as a product of two low-rank matrices.

Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2007] provides a probabilistic interpretation of matrix factorization. In PMF, latent factor variables are assumed to be marginally independent whereas rating variables, given the latent factor variables, are assumed to be conditionally independent.The main drawback of PMF is that

inferring the posterior distribution over the latent factors, given the ratings, is intractable. PMF handles this intractability by providing a maximum a posteriori estimation of the model parameters by maximizing the log-posterior over the model parameters, which is equivalent to minimizing the regularized square error loss. This optimization problem can be solved using stochastic gradient descent (SGD) [Koren *et al.*, 2009]. SGD is an online algorithm which obviates the need to store the entire dataset in the memory. Although SGD is scalable and enjoys local convergence guarantee [Sato, 2001], it often overfits the data and requires manual tuning of the learning rate and regularization parameters. Hence, maximum a posteriori estimation of matrix factorization suffers from the problem of overfitting and entails tedious job of finding the learning rate (if SGD is the choice of optimization) and regularization parameters.

On the other hand, fully Bayesian methods [Salakhutdinov and Mnih, 2008; Beal, 2003; Tzikas *et al.*, 2008; Hoffman *et al.*, 2013] for matrix factorization do not require manual tuning of learning rate and regularization parameters and are robust to overfitting. As direct evaluation of posterior is intractable in practice, approximate inference techniques are adopted to learn the posterior distribution. One of the possible choices to approximate inference is to apply variational approximate inference technique [Beal, 2003; Tzikas *et al.*, 2008]. Bayesian matrix factorization based on the variational approximation [Lim and Teh, 2007; Silva and Carin, 2012; Kim and Choi, 2014; Hoffman *et al.*, 2013] considers a simplified factorized distribution and assumes that the latent factor vectors of users are independent of the latent factor vectors of items while approximating the posterior. But this assumption often leads to over simplification and can produce inaccurate results as shown in [Salakhutdinov and Mnih, 2008]. On the other hand, Markov chain Monte Carlo (MCMC) based approximation method can produce exact results when provided with infinite resources. Bayesian Probabilistic Matrix Factorization (BPMF) [Salakhutdinov and Mnih, 2008] directly approximates the posterior distribution using the MCMC based Gibbs sampling inference mechanism and outperforms the variational based approximation.

In BPMF model, user/item latent factor vectors are assumed to follow a multivariate Gaussian distribution, which results a cubic time complexity with respect to the latent fac-

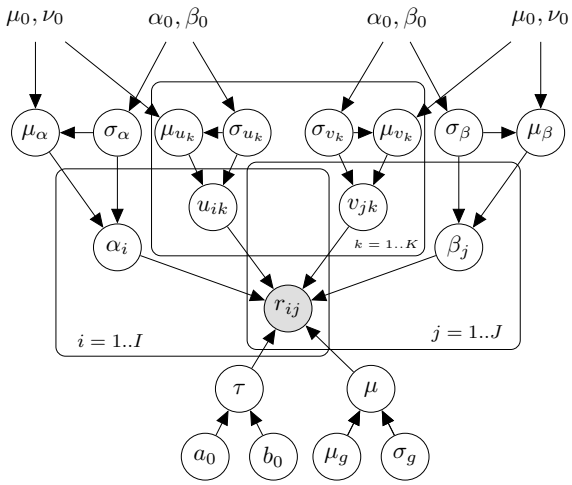


Figure 3.1: Graphical model representation of SBMF.

tor vector dimension. Though BPMF performs well in many applications, this cubic time complexity makes it difficult to apply BPMF on very large datasets. Hence, we propose the Scalable Bayesian Matrix Factorization (SBMF) which considers independent univariate Gaussian prior over latent factors as opposed to multivariate Guassian prior in BPMF. Due to this assumption, the time complexity of SBMF reduces to linear with respect to the dimension of latent space. We also consider user and item bias terms in SBMF model which are missing in BPMF model. These bias terms capture the variation in rating values that are independent of any user-item interaction. Also, the proposed SBMF algorithm is parallelized for multicore environments. We show through extensive experiments on three large scale real world datasets that the adopted univariate approximation in SBMF results in only a small performance loss and provides significant speed up when compared with the baseline method BPMF for higher latent space dimension.

The remainder of the chapter is structured as follows. Section 3.2 presents the SBMF model and its inference mechanism. Section 3.3 evaluates the performance of SBMF. Finally, the summary is presented in Section 3.4.

3.2 Method

3.2.1 Model

Figure 3.1 shows a graphical model representation of SBMF. Consider Ω as the set of observed entries in $\boldsymbol{R}$ provided during the training phase. The observed data r_{ij} is assumed to be generated as follows:

$$r_{ij} \sim \mathcal{N}\left(\mu + \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j, \tau^{-1}\right), \tag{3.1}$$

where τ is the precision parameter, μ is the global bias, α_i is the bias associated with the i^{th} user, β_j is the bias associated with the j^{th} item, $\boldsymbol{u}_i$ is the latent factor vector of dimension K associated with the i^{th}user, and $\boldsymbol{v}_j$ is the latent factor vector of dimension K associated with the j^{th} item. Bias terms are particularly helpful in capturing the individual bias for user/item: a user may have the tendency to rate all the items higher than the other users or an item may get higher ratings if it is perceived better than the others [Koren *et al.*, 2009].

The conditional on the observed entries of $\boldsymbol{R}$ (Likelihood term) can be written as follows:

$$p(\boldsymbol{R}|\boldsymbol{\Theta}) = \prod_{(i,j)\in\Omega} \mathcal{N}(r_{ij}|\mu + \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j, \tau^{-1}), \tag{3.2}$$

where $\boldsymbol{\Theta} = \{\tau, \mu, \{\alpha_i\}, \{\beta_j\}, \boldsymbol{U}, \boldsymbol{V}\}$. We place independent univariate priors on all the model parameters in $\boldsymbol{\Theta}$ as follows:

$$\mu \sim \mathcal{N}(\mu|\mu_g, \sigma_g^{-1}), \tag{3.3}$$

$$\alpha_i \sim \mathcal{N}(\alpha_i|\mu_\alpha, \sigma_\alpha^{-1}), \tag{3.4}$$

$$\beta_j \sim \mathcal{N}(\beta_j|\mu_\beta, \sigma_\beta^{-1}), \tag{3.5}$$

$$\boldsymbol{U} \sim \prod_{i=1}^{I} \prod_{k=1}^{K} \mathcal{N}(u_{ik}|\mu_{u_k}, \sigma_{u_k}^{-1}), \tag{3.6}$$

$$\boldsymbol{V} \sim \prod_{j=1}^{J} \prod_{k=1}^{K} \mathcal{N}(v_{jk}|\mu_{v_k}, \sigma_{v_k}^{-1}), \tag{3.7}$$

$$\tau \sim \text{Gamma}(\tau|a_0, b_0). \tag{3.8}$$

As normal-gamma is the conjugate prior of a normal distribution with unknown mean and precision, similar to BPMF, we place normal-gamma priors on hyperparamters $\mathbf{\Theta}_H = \{\mu_\alpha, \sigma_\alpha, \mu_\beta, \sigma_\beta, \{\mu_{u_k}, \sigma_{u_k}\}, \{\mu_{v_k}, \sigma_{v_k}\}\}$ as follows:

$$\mu_\alpha, \sigma_\alpha \sim \mathcal{NG}\left(\mu_\alpha, \sigma_\alpha | \mu_0, \nu_0, \alpha_0, \beta_0\right), \tag{3.9}$$

$$\mu_\beta, \sigma_\beta \sim \mathcal{NG}\left(\mu_\beta, \sigma_\beta | \mu_0, \nu_0, \alpha_0, \beta_0\right), \tag{3.10}$$

$$\mu_{u_k}, \sigma_{u_k} \sim \mathcal{NG}\left(\mu_{u_k}, \sigma_{u_k} | \mu_0, \nu_0, \alpha_0, \beta_0\right), \tag{3.11}$$

$$\mu_{v_k}, \sigma_{v_k} \sim \mathcal{NG}\left(\mu_{v_k}, \sigma_{v_k} | \mu_0, \nu_0, \alpha_0, \beta_0\right). \tag{3.12}$$

We denote $\{\mu_g, \sigma_g, a_0, b_0, \mu_0, \nu_0, \alpha_0, \beta_0\}$ as $\mathbf{\Theta}_0$ for notational convenience. The joint distribution of the observations and the hidden variables can be written as:

$$p(\boldsymbol{R}, \mathbf{\Theta}, \mathbf{\Theta}_H | \mathbf{\Theta}_0) = p(\boldsymbol{R}|\mathbf{\Theta}) p(\mu) \prod_{i=1}^{I} p(\alpha_i) \prod_{j=1}^{J} p(\beta_j) p(\boldsymbol{U}) p(\boldsymbol{V}) p(\mu_\alpha, \sigma_\alpha)$$
$$p(\mu_\beta, \sigma_\beta) \prod_{k=1}^{K} p(\mu_{u_k}, \sigma_{u_k}) p(\mu_{v_k}, \sigma_{v_k}). \tag{3.13}$$

3.2.2 Inference

Since evaluation of the joint distribution in Eq. (3.13) is intractable, we adopt a MCMC based Gibbs sampling approximate inference technique. As all our model parameters are conditionally conjugate [Hoffman *et al.*, 2013], equations for Gibbs sampling can be written in closed form using the joint distribution as given in Eq. (3.13). Replacing Eq. (3.2)-(3.12) in Eq. (3.13), the sampling distribution of u_{ik} can be written as follows:

$$u_{ik}|- \sim \mathcal{N}\left(u_{ik}|\mu^*, \sigma^*\right), \tag{3.14}$$

where

$$\sigma^* = \left(\sigma_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk}^2\right)^{-1}, \tag{3.15}$$

$$\mu^* = \sigma^* \left(\sigma_{u_k} \mu_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk} \left(r_{ij} - \left(\mu + \alpha_i + \beta_j + \sum_{l=1 \& l \neq k}^{K} u_{il} v_{jl}\right)\right)\right). \tag{3.16}$$

Table 3.1: Time and space complexity comparison between SBMF and BPMF.

Method	Time Complexity	Space Complexity
SBMF	$O(\|\Omega\|K)$	$O((I+J)K)$
BPMF	$O(\|\Omega\|K^2 + (I+J)K^3)$	$O((I+J)K)$

Here, Ω_i is the set of items rated by the i^{th} user in the training set. We sample model parameters in parallel whenever they are independent of each other. Algorithm 2 describes the detailed Gibbs sampling procedure. In each iteration of Gibbs sampling, line 7-12 sample hyperparameters. As line 7-12 are independent for each k, we sample them in parallel. Line 13-18 sample global bias parameter. As the sampling equation of user latent factor vectors are independent to each other, we sample them in line 19-30 in parallel. Similarly, we sample item latent factors in parallel in line 31-42.

3.2.3 Time and Space Complexity

Now, directly sampling u_{ik} from Eq. (3.14) requires $O(K|\Omega_i|)$ complexity. However if we precompute a quantity $e_{ij} = r_{ij} - (\mu + \alpha_i + \beta_j + \boldsymbol{u}_i^T \boldsymbol{v}_j)$ for all $(i,j) \in \Omega$ and write Eq. (3.16) as:

$$\mu^* = \sigma^* \left(\sigma_{u_k} \mu_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk} \left(e_{ij} + u_{ik} v_{jk} \right) \right). \tag{3.17}$$

then the sampling complexity of u_{ik} reduces to $O(|\Omega_i|)$. Table 3.1 shows the space and time complexities of SBMF and BPMF.

3.3 Experiments

3.3.1 Datasets

In this section, we show empirical results on three large real world movie-rating datasets[1,2] to validate the effectiveness of our proposed model. The details of these datasets are pro-

[1] http://grouplens.org/datasets/movielens/
[2] http://www.netflixprize.com/

Algorithm 2 Scalable Bayesian Matrix Factorization (SBMF)

Require: $\mathbf{\Theta}_0$, initialize $\mathbf{\Theta}$ and $\mathbf{\Theta}_H$.
Ensure: Compute e_{ij} for all $(i,j) \in \Omega$

for $t = 1$ **to** T **do**
  // Sample hyperparameters
  $\alpha^* \leftarrow \alpha_0 + \frac{1}{2}(I+1), \beta^* \leftarrow \beta_0 + \frac{1}{2}(\nu_0 (\mu_\alpha - \mu_0)^2 + \sum_{i=1}^{I} (\alpha_i - \mu_\alpha)), \sigma_\alpha | - \sim \text{Gamma}(\alpha^*, \beta^*)$.
  $\sigma^* \leftarrow (\nu_0 \sigma_\alpha + \sigma_\alpha I)^{-1}, \mu^* \leftarrow \sigma^*(\nu_0 \sigma_\alpha \mu_0 + \sigma_\alpha \sum_{i=1}^{I} \alpha_i), \mu_\alpha | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
  $\alpha^* \leftarrow \beta_0 + \frac{1}{2}(J+1), \beta^* \leftarrow \beta_0 + \frac{1}{2}(\nu_0 \left(\mu_\beta - \mu_0\right)^2 + \sum_{j=1}^{J} \left(\beta_j - \mu_\beta\right)), \sigma_\beta | - \sim \text{Gamma}(\alpha^*, \beta^*)$.
  $\sigma^* \leftarrow (\nu_0 \sigma_\beta + \sigma_\beta J)^{-1}, \mu^* \leftarrow \sigma^*(\nu_0 \sigma_\beta \mu_0 + \sigma_\beta \sum_{j=1}^{J} \beta_j), \mu_\beta | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
  for $k = 1$ to K **do in parallel**
    $\alpha^* \leftarrow \alpha_0 + \frac{1}{2}(I+1), \beta^* \leftarrow \beta_0 + \frac{1}{2}(\nu_0 (\mu_{u_k} - \mu_0)^2 + \sum_{i=1}^{I} (u_{ik} - \mu_{u_k})), \sigma_{u_k} | - \sim \text{Gamma}(\alpha^*, \beta^*)$.
    $\sigma^* \leftarrow (\nu_0 \sigma_{u_k} + \sigma_{u_k} I)^{-1}, \mu^* \leftarrow \sigma^*(\nu_0 \sigma_{u_k} \mu_0 + \sigma_{u_k} \sum_{i=1}^{I} u_{ik}), \mu_{u_k} | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
    $\alpha^* \leftarrow \beta_0 + \frac{1}{2}(J+1), \beta^* \leftarrow \beta_0 + \frac{1}{2}(\nu_0 (\mu_{v_k} - \mu_0)^2 + \sum_{j=1}^{J} \left(v_{jk} - \mu_{v_k}\right)), \sigma_{v_k} | - \sim \text{Gamma}(\alpha^*, \beta^*)$.
    $\sigma^* \leftarrow (\nu_0 \sigma_{v_k} + \sigma_{v_k} J)^{-1}, \mu^* \leftarrow \sigma^*(\nu_0 \sigma_{v_k} \mu_0 + \sigma_{v_k} \sum_{j=1}^{J} v_{jk}), \mu_{v_k} | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
  end for
  $a_0^* \leftarrow a_0 + \frac{1}{2}|\Omega|, b_0^* \leftarrow b_0 + \frac{1}{2} \sum_{(i,j) \in \Omega} e_{ij}^2, \tau | - \sim \text{Gamma}(a_0^*, b_0^*)$.
  // Sample model parameters
  $\sigma^* \leftarrow (\sigma_g + \tau |\Omega|)^{-1}, \mu^* \leftarrow \sigma^*(\sigma_g \mu_g + \tau \sum_{(i,j) \in \Omega} (e_{ij} + \mu)), \mu | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
  for $(i,j) \in \Omega$ **do in parallel**
    $e_{ij} \leftarrow e_{ij} + (\mu_{old} - \mu)$
  end for
  for $i = 1$ **to** I **do in parallel**
    $\sigma^* \leftarrow (\sigma_\alpha + \tau |\Omega_i|)^{-1}, \mu^* \leftarrow \sigma^*(\sigma_\alpha \mu_\alpha + \tau \sum_{j \in \Omega_i} (e_{ij} + \alpha_i)), \alpha_i | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
    for $j \in \Omega_i$ **do**
      $e_{ij} \leftarrow e_{ij} + (\alpha_{old} - \alpha_i)$
    end for
    for $k = 1$ **to** K **do**
      $\sigma^* \leftarrow (\sigma_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk}^2)^{-1}, \mu^* \leftarrow \sigma^*(\sigma_{u_k} \mu_{u_k} + \tau \sum_{j \in \Omega_i} v_{jk} \left(e_{ij} + u_{ik} v_{jk}\right)), u_{ik} | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
      for $j \in \Omega_i$ **do**
        $e_{ij} \leftarrow e_{ij} + v_{jk}(u_{ik}^{old} - u_{ik})$
      end for
    end for
  end for
  for $j = 1$ **to** J **do in parallel**
    $\sigma^* \leftarrow (\sigma_\beta + \tau |\Omega_j|)^{-1}, \mu^* \leftarrow \sigma^*(\sigma_\beta \mu_\beta + \tau \sum_{i \in \Omega_j} (e_{ij} + \beta_j)), \beta_j | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
    for $i \in \Omega_j$ **do**
      $e_{ij} \leftarrow e_{ij} + (\beta_{old} - \beta_j)$
    end for
    for $k = 1$ **to** K **do**
      $\sigma^* \leftarrow (\sigma_{v_k} + \tau \sum_{i \in \Omega_j} u_{ik}^2)^{-1}, \mu^* \leftarrow \sigma^*(\sigma_{v_k} \mu_{v_k} + \tau \sum_{i \in \Omega_j} u_{ik} \left(e_{ij} + u_{ik} v_{jk}\right)), v_{jk} | - \sim \mathcal{N}(\mu^*, \sigma^*)$.
      for $i \in \Omega_j$ **do**
        $e_{ij} \leftarrow e_{ij} + u_{ik}(v_{jk}^{old} - v_{jk})$
      end for
    end for
  end for
end for

vided in Table 3.2. Both the Movielens datasets are publicly available and 90:10 split is used to create their train and test sets. For Netflix, the probe data is used as the test set.

3.3.2 Experimental Setup and Parameter Selection

All the experiments are run on an Intel i5 machine with 16GB RAM. We have considered the serial as well as the parallel implementation of SBMF for all the experiments. In the parallel implementation, SBMF is parallelized in multi-core environment using OpenMP library. Although BPMF can also be parallelized, the base paper [Salakhutdinov and Mnih, 2008] and its publicly available code provide only the serial implementation. So in our experiments, we have compared only the serial implementation of BPMF against the serial and the parallel implementations of SBMF. Serial and parallel versions of the SBMF are denoted as SBMF-S and SBMF-P respectively. Since performance depends on the number of factors K, it is necessary to investigate how the models work with different values of K, for both SBMF and BPMF. Hence, three sets of experiments are run for each dataset corresponding to $K = \{50, 100, 200\}$ for SBMF-S, SBMF-P, and BPMF. As our main aim is to validate that SBMF is more scalable as compared to BPMF under same conditions, we choose 50 burn-in iterations for all the experiments of SBMF-S, SBMF-P, and BPMF. In Gibbs sampling process burn-in refers to the practice of discarding an initial portion of a Markov chain sample, so that the effect of initial values on the posterior inference is minimized. Note that, if SBMF takes less time than BPMF for a particular burn-in period, then increasing the number of burn-in iterations will make SBMF more scalable as compared to BPMF. Additionally, we allow the methods to have 100 collection iterations where collection iterations are the ones that come after the burn-in iterations and contribute to the sample collections for the variables.

In SBMF, we initialize parameters in $\mathbf{\Theta}$ using a Gaussian distribution with zero mean and 0.01 variance. All the parameters in $\mathbf{\Theta}_H$ are set to zero. Also, $a_0, b_0, \nu_0, \alpha_0$, and β_0 are set to one, and μ_0 and μ_g are set to zero. σ_g is initialized to 0.01. For BPMF, we use standard parameter setting as provided in the paper [Salakhutdinov and Mnih, 2008]. We collect samples of user and item latent factor vectors and bias terms from the collection

Table 3.2: Description of the datasets.

Dataset	No. of users	No. of movies	No. of ratings
Movielens 10m	71567	10681	**10m**
Movielens 20m	138493	27278	**20m**
Netflix	480189	17770	**100m**

iterations and approximate a rating r_{ij} as:

$$\hat{r}_{ij}^{t} = \frac{1}{C}\sum_{c=1}^{C}\left(\mu^{c} + \alpha_{i}^{c} + \beta_{j}^{c} + (\boldsymbol{u}_{i}^{c})^{\intercal}\boldsymbol{v}_{j}^{c}\right), \tag{3.18}$$

where $\boldsymbol{u}_i^c$ and $\boldsymbol{v}_j^c$ are the c^{th} drawn samples of the i^{th} user and the j^{th} item latent factor vectors respectively, μ^c, α_i^c, and β_j^c are the c^{th} drawn samples of the global bias, the i^{th} user bias, and the j^{th} item bias respectively. C is the number of drawn samples. Then the Root mean square error (RMSE) [Koren *et al.*, 2009] is used as the evaluation metric for all the experiments.

3.3.3 Results

In Figure 3.2, for all the graphs, x-axis represents the time elapsed since the starting of an experiment and y-axis presents the RMSE value. Since we allow 50 burn-in iterations for all the experiments and each iteration of BPMF takes more time than SBMF-P's, collection iterations of SBMF-P begin earlier than BPMF's and thus we get the initial RMSE value of SBMF-P earlier. In SBMF-S also, iterations take less time as compared to BPMF's iterations, except for $K = \{50, 100\}$ in the Netflix dataset, where iterations of SBMF-S take more time than iterations of BPMF. We believe that in Netflix dataset, for $K = 50$ and 100, BPMF takes less time than SBMF-S because BPMF is implemented in Matlab where matrix computations are efficient. On the other hand, SBMF is implemented in C++ where the matrix storage is unoptimized. As the Netflix data is large with respect to the number of entries as well as the number of users and items, number of matrix operations are more in it as compared to other datasets. So for lower values of K, the cost of matrix operations for SBMF-S dominates the cost incurred due to $O(K^3)$ complexity of BPMF, thus BPMF takes less time than SBMF-S. But with large values of K, BPMF start taking more time

Table 3.3: Results comparison between SBMF and BPMF.

		$K = 50$		$K = 100$		$K = 200$	
Dataset	Method	RMSE	Time(Hr)	RMSE	Time(Hr)	RMSE	Time(Hr)
Movielens 10m	BPMF	**0.8629**	1.317	**0.8638**	3.517	**0.8651**	22.058
	SBMF-S	0.8655	1.091	0.8667	2.316	0.8654	5.205
	SBMF-P	0.8646	**0.462**	0.8659	**0.990**	0.8657	**2.214**
Movielens 20m	BPMF	**0.7534**	2.683	**0.7513**	6.761	**0.7508**	45.355
	SBMF-S	0.7553	2.364	0.7545	5.073	0.7549	11.378
	SBMF-P	0.7553	**1.142**	0.7545	**2.427**	0.7551	**5.321**
Netflix	BPMF	0.9057	11.739	**0.9021**	28.797	**0.8997**	150.026
	SBMF-S	0.9048	17.973	0.9028	40.287	0.9017	89.809
	SBMF-P	**0.9047**	**7.902**	0.9026	**16.477**	0.9017	**34.934**

as the $O(K^3)$ complexity of BPMF becomes dominating. We leave the task of optimizing the code of SBMF, to decrease the runtime of SBMF, as future work.

We can observe from Figure 3.2 that SBMF-P takes much less time in all the experiments than BPMF and incurs only a small loss in the performance. Similarly, SBMF-S also takes less time than the BPMF (except for $K = \{50, 100\}$ in Netflix dataset) and incurs only a small performance loss. Important point to note is that total time difference between both of the variants of SBMF and BPMF increases with latent factor dimension and the speedup is significantly high for $K = 200$. Table 3.3 shows the final RMSE values and the total time taken corresponding to each dataset and K. We find that the RMSE values for SBMF-S and SBMF-P are very close for all the experiments. We also observe that increasing the latent dimension reduces the RMSE value in the Netflix dataset. Note, in past it has been shown that increasing the number of latent factors improves RMSE [Koren, 2009; Salakhutdinov and Mnih, 2008]. With high latent dimension, the running time for BPMF is significantly high due to its cubic time complexity with respect to the latent dimension and it takes approximately 150 hours on Netflix dataset with $K = 200$. However, SBMF has linear complexity with respect to the latent dimension and SBMF-P and SBMF-S take only 35 and 90 hours (approximately) respectively on the Netflix dataset with $K = 200$. Thus SBMF is more suited for large datasets with large factor dimensions. Similar speed up patterns are found on the other datasets also.

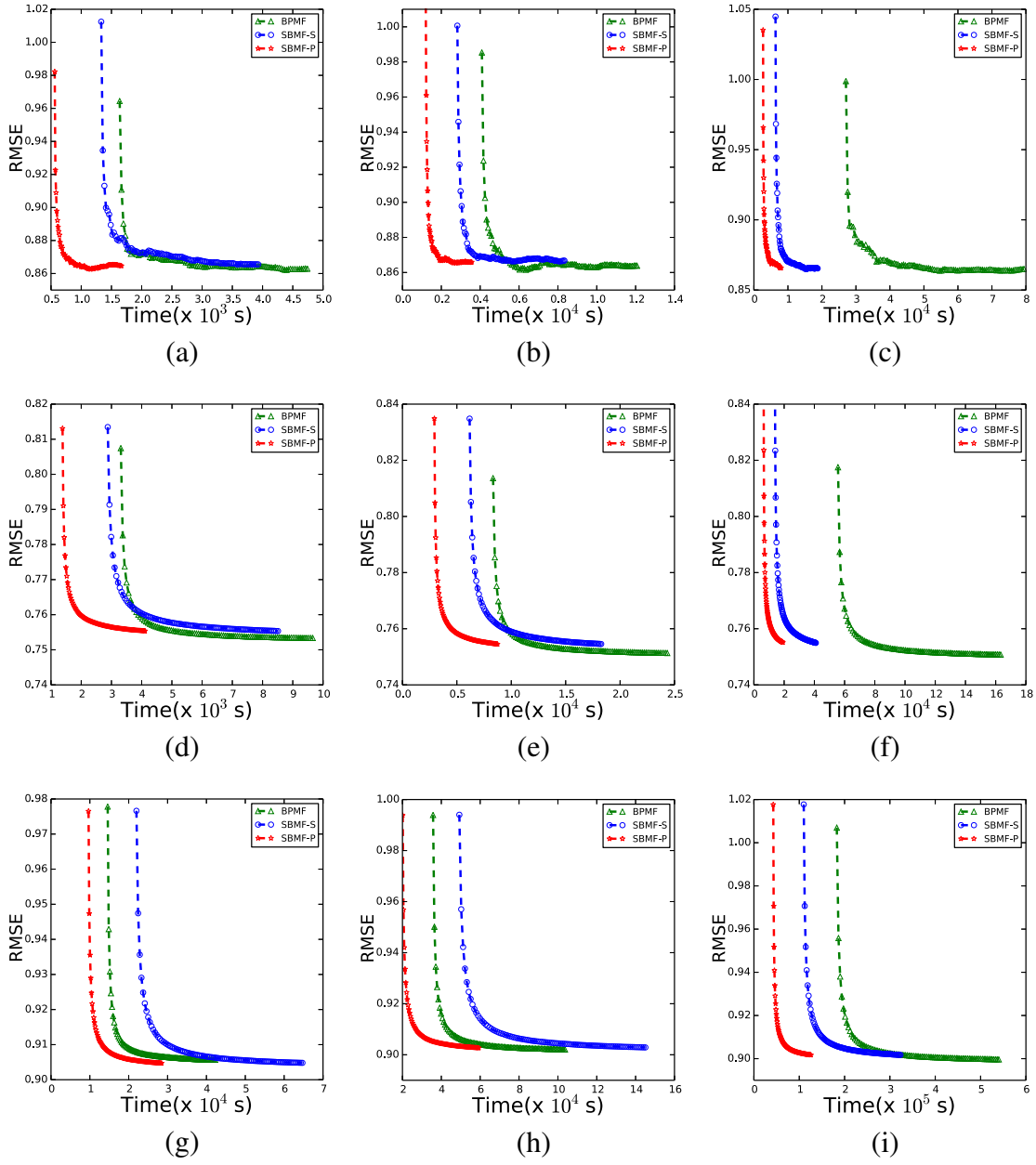


Figure 3.2: Left, middle, and right columns correspond to the results for $K = 50$, 100, and 200 respectively. {a,b,c}, {d,e,f}, and {g,h,i} are results on Movielens 10m, Movielens 20m, and Netflix datasets respectively.

3.4 Summary

We have proposed the Scalable Bayesian Matrix Factorization (SBMF), which is a Markov chain Monte Carlo based Gibbs sampling algorithm for matrix factorization, has linear time complexity with respect to the target rank and linear space complexity with respect to the number of non-zero observations. Also, we show using extensive experiments on

three sufficiently large real world datasets that SBMF incurs only a small loss in the performance and takes much less time as compared to the baseline Bayesian Probabilistic Matrix Factorization (BPMF) for higher latent space dimension. It is worth while to note that with small latent space dimension we should use BPMF. However for higher latent space dimension, SBMF is preferred.

CHAPTER 4

SCALABLE VARIATIONAL BAYESIAN FACTORIZATION MACHINE

In this chapter, we develop the Variational Bayesian Factorization Machine (VBFM) which is a scalable variational Bayesian inference algorithm for Factorization Machine (FM). VBFM converges faster than the existing state-of-the-art Markov chain Monte Carlo (MCMC) based Gibbs sampling inference algorithm for FM while providing similar performance. Additionally, for large-scale learning, we propose the Online Variational Bayesian Factorization Machine (OVBFM) which utilizes stochastic gradient descent (SGD) to optimize the lower bound in variational approximation. OVBFM outperforms existing online algorithm for FM as validated by extensive experiments performed on numerous large-scale real world datasets.

4.1 Introduction

Feature based methods, such as Support Vector Machines (SVMs), are one of the standard approach in machine learning which work on the generic features extracted from the data. Standard tools like LIBSVM [Chang and Lin, 2011], SVM-Light [Joachims, 2002] for SVMs can be applied for feature based methods. These approach do not require expert's intervention to extract information from the data. However, feature based methods fail in domains with very sparse and high dimensional data, where instead, a class of algorithms called factorization model is widely used due to its prediction quality and scalability. Matrix factorization [Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007, 2008; Gopalan *et al.*, 2015] is the simplest and most well studied factorization model. Though factorization models have been successful due to their simplicity, performance and scalability in several domains, deploying these models to new prediction problems is non-trivial and requires

– 1) designing of the model and feature representation for the specific application; 2) deriving learning or inference algorithm; and 3) implementing the approach for that specific application – all of which are time consuming and often call for domain expertise.

Factorization Machine (FM) [Rendle, 2010] is a generic framework which combines high prediction quality of factorization model with the flexibility of feature engineering. FM represents data as real-valued features like standard machine learning approaches, such as SVMs, and uses interactions between each pair of variables in a low-dimensional latent space. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [Koren *et al.*, 2009], SVD++ [Koren, 2008], TimeSVD++ [Koren, 2009], Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme, 2010], and factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2011*a*].

One of popular learning algorithm for FM is SGD-FM [Rendle, 2010] which uses stochastic gradient descent (SGD) to learn the model. Though SGD is scalable and enjoys local convergence guarantee [Sato, 2001], it often overfits the data and requires manual tuning of learning rate and regularization parameters [Salakhutdinov and Mnih, 2007]. Alternative methods to solve FM include MCMC-FM [C. Freudenthaler, 2011] which provides state-of-the-art performance using Markov chain Monte Carlo (MCMC) based Gibbs sampling as the inference mechanism and avoids expensive manual tuning of the learning rate and regularization parameters. However, MCMC-FM is a batch learning method and is less straight forward to apply to datasets as large as the KDD music dataset [Dror *et al.*, 2012]. Also, it is difficult to preset the values of burn-in and collection iterations and gauge the convergence of the MCMC inference framework, a known problem with sampling based techniques.

Variational inference, an alternative to MCMC sampling, transforms a complex inference problem into a high-dimensional optimization problem [Beal, 2003; Tzikas *et al.*, 2008; Hoffman *et al.*, 2013]. Typically, the optimization is solved using a coordinate ascent algorithm and hence is more scalable compared to MCMC sampling [Hoffman *et al.*, 2010]. Motivated by the scalability of variational methods, we propose a batch Variational Bayesian Factorization Machine (VBFM). Empirically, VBFM is found to converge faster than MCMC-FM and performs as good as MCMC-FM asymptotically. The convergence

is also easy to track when the objective associated with the variational approximation in VBFM stops changing significantly. Additionally, the Online Variational Bayesian Factorization Machine (OVBFM) is introduced which uses SGD for maximizing the lower bound obtained from the variational approximation and performs much better than the existing online algorithm of FM that uses SGD. To summarize, the chapter makes the following contribution:

1. The VBFM is proposed which converges faster than MCMC-FM.
2. The OVBFM is introduced which exploits the advantages of online learning and performs much better than SGD-FM.
3. Extensive experiments on real-world datasets validate the superiority of both VBFM and OVBFM.

The remainder of the chapter is structured as follows. Section 4.2 presents the model description. A detailed description of the inference mechanism for both VBFM and OVBFM is provided in Section 4.3. Section 4.4 evaluates the performance of both VBFM and OVBFM on several real world datasets. Finally, the summary is presented in Section 4.5.

4.2 Model

Consider a training set containing N tuples of the form $(\boldsymbol{x}_n, y_n)_{n=1}^N$ where each tuple consists of the covariate $\boldsymbol{x}_n$ and the response variable y_n. The data is further represented in the form of Figure 2.1 where the i^{th} column represents the i^{th} variable. For detailed description of FM please refer to Section 2.2.4. Similar to Eq. (2.19), the observed data y_n is assumed to be generated as follows:

$$y_n \sim \mathcal{N}\left(w_0 + \sum_{i=1}^{D} x_{ni} w_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni} x_{nj} \boldsymbol{v}_i^{\intercal} \boldsymbol{v}_j, \alpha^{-1}\right), \tag{4.1}$$

where α is the precision parameter, w_0 is the global bias, w_i is the bias associated with the i^{th} variable, and $\boldsymbol{v}_i$ is the latent factor vector of dimension K associated with the i^{th}

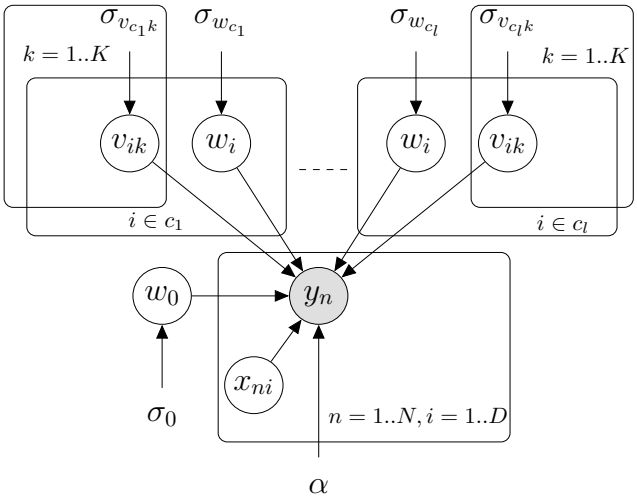


Figure 4.1: Graphical model representation of VBFM.

variable. Independent prior is imposed on each of the latent variables as follows:

$$w_0 \sim \mathcal{N}(w_0|0, \sigma_0^{-1}), \tag{4.2}$$

$$w_i \sim \mathcal{N}(w_i|0, \sigma_{w_{c_i}}^{-1}), \tag{4.3}$$

$$v_{ik} \sim \mathcal{N}(v_{ik}|0, \sigma_{v_{c_i k}}^{-1}), \tag{4.4}$$

where c_i denotes the group in which the i^{th} variable belongs to. For example, in Figure 2.1, users, songs, and genres form three separate groups. If users form the first group, then according to this notation, $\{u_1, u_2, \cdots\}$ belong to group c_1. Figure 4.1 shows a graphical model for VBFM with l different groups.

Note that unlike MCMC-FM, VBFM does not incorporate hyperpriors over the model hyperparamters.

4.3 Approximate Inference

4.3.1 Batch Variational Inference

Let $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ be the sparse feature representation of input data and $\boldsymbol{y}$ be the vector of corresponding response variables, as shown in Figure 2.1. For notational convenience, the

set of parameters in the model is denoted by $\boldsymbol{\theta} = \{\alpha, \sigma_0, \{\sigma_{w_{c_i}}\}_i, \{\sigma_{v_{c_i k}}\}_{i,k}\}$ and the set of latent variables is represented concisely by $\boldsymbol{Z} = \{w_0, \{w_i\}_i, \boldsymbol{V}\}$.

Given a training set, the objective is to maximize the likelihood of the observations given by $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$. The marginal log-likelihood can be written as:

$$\ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q||p), \tag{4.5}$$

where

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int_Z q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{Z}, \boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})} \right) d\boldsymbol{Z}, \tag{4.6}$$

$$KL(q||p) = -\int q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})} \right) d\boldsymbol{Z}, \tag{4.7}$$

where $q(\boldsymbol{Z})$ is any probability distribution. $KL(q||p)$ is the Kullback-Leibler divergence between $q(\boldsymbol{Z})$ and $p(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})$, and is always non-negative. Thus $\ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ is lower-bounded by the term $\mathcal{L}(q, \boldsymbol{\theta})$, also known as evidence lower bound (ELBO) [Beal, 2003; Tzikas *et al.*, 2008]. While maximizing the ELBO $\mathcal{L}(q, \boldsymbol{\theta})$, note that the optimum is achieved at $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})$. However, in practice working with the true posterior distribution is intractable. Therefore, a restrictive form of $q(\boldsymbol{Z})$ is considered, and then a member of this family is found which minimizes the KL divergence. For large-scale applications, a fully factorized variational distribution is considered:

$$q(\boldsymbol{Z}) = q(w_0) \prod_{i=1}^{D} q(w_i) \prod_{i=1}^{D} \prod_{k=1}^{K} q(v_{ik}), \tag{4.8}$$

where

$$q(w_0) = \mathcal{N}(w_0|\mu_0^{'}, \sigma_0^{'}), \tag{4.9}$$

$$q(w_i) = \mathcal{N}(w_i|\mu_{w_i}^{'}, \sigma_{w_i}^{'}), \tag{4.10}$$

$$q(v_{ik}) = \mathcal{N}(v_{ik}|\mu_{v_{ik}}^{'}, \sigma_{v_{ik}}^{'}). \tag{4.11}$$

The ELBO in Eq. (4.6) can be calculated as follows:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{n=1}^{N} F_n + F^0 + \sum_{i=1}^{D} F_i^w + \sum_{i=1}^{D}\sum_{k=1}^{K} F_{ik}^v, \tag{4.12}$$

where

$$F_n = E_q\left[\sum_{n=1}^{N} \ln \mathcal{N}(y_n|\boldsymbol{x}_n, \boldsymbol{Z}, \alpha^{-1})\right] = -\frac{1}{2}\ln 2\pi\alpha^{-1} - \frac{\alpha}{2}((y_n - \bar{y}_n)^2 + T_n), \tag{4.13}$$

$$\bar{y}_n = \mu_0^{'} + \sum_{i=1}^{D}\mu_{w_i}^{'}x_{ni} + \sum_{i=1}^{D}\sum_{j=i+1}^{D} x_{ni}x_{nj}\sum_{k=1}^{K}\mu_{v_{ik}}^{'}\mu_{v_{jk}}^{'}, \tag{4.14}$$

$$\begin{aligned} T_n = {} & \sigma_0^{'} + \sum_{i=1}^{D}\sigma_{w_i}^{'}x_{ni}^2 + \sum_{i=1}^{D}\sum_{j=i+1}^{D} x_{ni}^2x_{nj}^2\sum_{k=1}^{K}\left(\mu_{v_{ik}}^{'2}\sigma_{v_{jk}}^{'} + \mu_{v_{jk}}^{'2}\sigma_{v_{ik}}^{'} + \sigma_{v_{ik}}^{'}\sigma_{v_{jk}}^{'}\right) \\ & + 2\sum_{i=1}^{D} x_{ni}^2\sigma_{v_{ik}}^{'}\sum_{j=1}^{D}\sum_{(j'=j+1)\&(j\&j'\neq i)}^{D} x_{nj}x_{nj'}\mu_{v_{jk}}^{'}\mu_{v_{j'k}}^{'} \end{aligned} \tag{4.15}$$

$$F^0 = E_q[\ln\mathcal{N}(w_0|0, \sigma_0^{-1}) - \ln\mathcal{N}(w_0|\mu_0^{'}, \sigma_0^{'})] = \frac{1}{2} + \frac{1}{2}\ln\sigma_0\sigma_0^{'} - \frac{\sigma_0}{2}(\mu_0^{'2} + \sigma_0^{'}), \tag{4.16}$$

$$\begin{aligned} F_i^w = {} & E_q[\ln\mathcal{N}(w_i|0, \sigma_{w_{c_i}}^{-1}) - \ln\mathcal{N}(w_i|\mu_{w_i}^{'}, \sigma_{w_i}^{'})] = \frac{1}{2} + \frac{1}{2}\ln\sigma_{w_i}^{'}\sigma_{w_{c_i}} \\ & - \frac{\sigma_{w_{c_i}}}{2}(\mu_{w_i}^{'2} + \sigma_{w_i}^{'}), \end{aligned} \tag{4.17}$$

$$\begin{aligned} F_{ik}^v = {} & E_q[\ln\mathcal{N}(v_{ik}|0, \sigma_{v_{c_ik}}^{-1}) - \ln\mathcal{N}(v_{ik}|\mu_{v_{ik}}^{'}, \sigma_{v_{ik}}^{'})] = \frac{1}{2} + \frac{1}{2}\ln\sigma_{v_{ik}}^{'}\sigma_{v_{c_ik}} \\ & - \frac{\sigma_{v_{c_ik}}}{2}(\mu_{v_{ik}}^{'2} + \sigma_{v_{ik}}^{'}). \end{aligned} \tag{4.18}$$

Let $\mathcal{Q}$ be the set of all distributions having a fully factorized form as given in Eq. (4.8). The optimal distribution that produces the tightest possible lower bound $\mathcal{L}$ is given by:

$$q^* \quad = \quad \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q(\boldsymbol{Z})||p(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})). \tag{4.19}$$

The update rule corresponding to a variational parameter describing q^* can be obtained by setting the derivative of Eq. (4.12) with respect to that variational parameter to zero. For example, the update equation for the variational parameters associated with $q^*(v_{ik})$ can be

written as follows:

$$\sigma^{'}_{v_{ik}} = \left(\sigma_{v_{c_i k}} + \alpha \sum_{n=1}^{N} x_{ni}^2 \left(\left(\sum_{l=1 \& l \neq i}^{D} x_{nl} \mu^{'}_{v_{lk}} \right)^2 + \sum_{l=1 \& l \neq i}^{D} x_{nl}^2 \sigma^{'}_{v_{lk}} \right) \right)^{-1}, \tag{4.20}$$

$$\mu^{'}_{v_{ik}} = \sigma^{'}_{v_{ik}} \alpha \sum_{n=1}^{N} x_{ni} \sum_{l=1 \& l \neq i}^{D} x_{nl} \mu^{'}_{v_{lk}} \left(y_n - \bar{y}_n + x_{ni} \mu^{'}_{v_{ik}} \sum_{l=1 \& l \neq i}^{D} x_{nl} \mu^{'}_{v_{lk}} \right). \tag{4.21}$$

Straightforward implementation of Eq. (4.20) and (4.21) requires $O(KN_iD)$ complexity, where N_i is a set of indices n for which x_{ni} is non-zero. However, using a simple trick we can reduce the complexity to $O(N_iD)$. To that end, we first show the calculation of $\bar{y}_n$:

$$\bar{y}_n = \mu^{'}_0 + \sum_{i=1}^{D} \mu^{'}_{w_i} x_{ni} + \frac{1}{2} \sum_{k=1}^{K} \left(\left(\sum_{i=1}^{D} x_{ni} \mu^{'}_{v_{ik}} \right)^2 - \sum_{i=1}^{D} x_{ni}^2 \mu^{'2}_{v_{ik}} \right) \tag{4.22}$$

Now, $\bar{y}_n$ can be computed in $O(KD)$ time. However, the complexity to update Eq. (4.20) and (4.21) is still $O(KN_iD)$. We can reduce the complexity to $O(N_iD)$ by pre-computing the quantities $R_n = (y_n - \bar{y}_n)$ for all the training points. The update equations, with R_n, can be written as follows:

$$\sigma^{'}_{v_{ik}} = \left(\sigma_{v_{c_i k}} + \alpha \sum_{n=1}^{N} x_{ni}^2 \left(S_1(i,k)^2 + S_2(i,k) \right) \right)^{-1}, \tag{4.23}$$

$$\mu^{'}_{v_{ik}} = \sigma^{'}_{v_{ik}} \alpha \sum_{n=1}^{N} x_{ni} S_1(i,k) \left(R_n + x_{ni} \mu^{'}_{v_{ik}} S_1(i,k) \right), \tag{4.24}$$

where $S_1(i,k) = \sum_{l=1}^{D} x_{nl} \mu^{'}_{v_{lk}} - x_{ni} \mu^{'}_{v_{ik}}$ and $S_2(i,k) = \sum_{l=1}^{D} x_{nl}^2 \sigma^{'}_{v_{lk}} - x_{ni}^2 \sigma^{'}_{v_{ik}}$. The same trick works for all other parameters as well. R_n is updated iteratively as and when each hyperparameter is updated. The detail procedure is presented in Algorithm 3. In each iteration, Algorithm 3 updates all the parameters in turn. In line 3-9, variational parameters of the global bias term are updated. Line 11-31 update the variational parameters correspond to the individual bias term and the pairwise interaction term. Then all the model hyperparamters are updated in line 33-42. This procedure is then repeated for M number of iterations.

Algorithm 3 Variational Bayesian Factorization Machine (VBFM)

Require: $\alpha, \sigma_0, \sigma_{w_{c_i}}, \sigma_{v_{c_i,k}} \; \forall i, k$
Ensure: Randomly Initialize $\sigma_0^{'}$, $\mu_0^{'}$, $\sigma_{w_i}^{'}$, $\mu_{w_i}^{'}$, $\sigma_{v_{ik}}^{'}$, $\mu_{v_{ik}}^{'} \; \forall i, k$
Ensure: Compute R_n for all the training data points.
for $t = 1$ **to** M **do**
  // Update w_0's parameter
  $\sigma_{old} \leftarrow \sigma_0^{'}$
  $\mu_{old} \leftarrow \mu_0^{'}$
  $\sigma_0^{'} \leftarrow (\sigma_0 + \alpha N)^{-1}$
  $\mu_0^{'} \leftarrow \sigma_0^{'} \alpha \sum_{n=1}^{N} \left(R_n + \mu_0^{'}\right)$
  for $n = 1$ **to** N **do**
    $R_n \leftarrow R_n + \mu_{old} - \mu_0^{'}$
  end for
  // Update w_i's parameter
  for $i = 1$ **to** D **do**
    $\sigma_{old} \leftarrow \sigma_{w_i}^{'}$
    $\mu_{old} \leftarrow \mu_{w_i}^{'}$
    $\sigma_{w_i}^{'} \leftarrow \left(\sigma_{w_{c_i}} + \alpha \sum_{n=1}^{N} x_{ni}^2\right)^{-1}$
    $\mu_{w_i}^{'} \leftarrow \sigma_{w_i}^{'} \alpha \sum_{n=1}^{N} x_{ni} \left(R_n + x_{ni} \mu_{w_i}^{'}\right)$
    for n in Ω_i **do**
      $R_n \leftarrow R_n + x_{ni}(\mu_{old} - \mu_{w_i}^{'})$
    end for
  end for
  // Update v_{ik}'s parameter
  for $k = 1$ **to** K **do**
    for $i = 1$ **to** D **do**
      $\sigma_{old} \leftarrow \sigma_{v_{ik}}^{'}$
      $\mu_{old} \leftarrow \mu_{v_{ik}}^{'}$
      $\sigma_{v_{ik}}^{'} \leftarrow \left((\sigma_{v_{c_i k}} + \alpha \sum_{n=1}^{N} x_{ni}^2 (S_1(i,k)^2 + S_2(i,k))\right)^{-1}$
      $\mu_{v_{ik}}^{'} \leftarrow \sigma_{v_{ik}}^{'} \alpha \sum_{n=1}^{N} x_{ni} S_1(i,k) [R_n + x_{ni} \mu_{v_{ik}} S_1(i,k)]$
      for n in Ω_i **do**
        $R_n \leftarrow R_n + x_{ni} S_1(i,k)(\mu_{old} - \mu_{v_{ik}}^{'})$
      end for
    end for
  end for
  // Update hyperparameters
  $\alpha \leftarrow \frac{N}{\sum_{n=1}^{N} R_n^2 + T_n}$
  $\sigma_0 \leftarrow \frac{1}{\mu_0^2 + \sigma_0}$
  for $i = 1$ **to** $|c|$ **do**
    $\sigma_{w_{c_i}} \leftarrow \frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{w_j}^{'2} + \sigma_{w_j}^{'}\right)}$
  end for
  for $k = 1$ **to** K **do**
    for $i = 1$ **to** $|c|$ **do**
      $\sigma_{v_{c_i k}} \leftarrow \frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{v_{jk}}^{'2} + \sigma_{v_{jk}}^{'}\right)}$
    end for
  end for
end for

4.3.2 Stochastic Variational Inference

Stochastic approximation methods follow noisy gradient of a target function with decreasing step sizes. Such noisy gradient is calculated only for the sub-sampled data (data generated from the original dataset according to a sampling mechanism), the computation of which is often cheaper. Scaling of the objective function is necessary in such case to ensure that the expectation of the noisy gradient is equal to the gradient of the original target func-

tion. Therefore, in the implementation, after sub-sampling a data instance n uniformly at random from the given dataset, the noisy estimate of $\mathcal{L}(q, \boldsymbol{\theta})$ can be computed as follows:

$$\mathcal{L}_{\text{noisy}}(q, \boldsymbol{\theta}) = {s_n}^{-1} F_n + F^0 + \sum_{i=1; i \in n}^{D} F_i^w + \sum_{i=1; i \in n}^{D} \sum_{k=1}^{K} F_{ik}^v, \tag{4.25}$$

where s_n is the rescaling constant. Eq. (4.25) is the rescaled version of Eq. (4.12). Rescale factors for w_0, w_i and v_{ik} are set to N, $|\Omega_i|$, and $|\Omega_i|$ respectively. The variational parameters associated with $q(\boldsymbol{Z})$ are updated by making a small step in the direction of the gradient of Eq. (4.25). Since natural gradient leads to faster convergence [Amari, 1998; Hoffman *et al.*, 2013], natural parameters of $q(\boldsymbol{Z})$ are considered for the updates. The natural gradient of a function accounts for the information geometry of its parameter space. The classical gradient method for maximization tries to find a maximum of a function by taking small steps in the direction of the gradient. The gradient (when it exists) points in the direction of steepest ascent. However, the Euclidean metric might not capture a meaningful notion of distance [Hoffman *et al.*, 2013]. The natural gradient corrects for this issue by redefining the basic definition of the gradient [Amari, 1998]. While the Euclidean gradient points in the direction of steepest ascent in Euclidean space, the natural gradient points in the direction of steepest ascent in the Riemannian space, that is, the space where local distance is defined by KL divergence rather than the L2 norm.

Natural parameters are represented as: $\bar{v}_{ik} = \frac{\mu'_{v_{ik}}}{\sigma'_{v_{ik}}}$, $\hat{v}_{ik} = \frac{1}{\sigma'_{v_{ik}}}$ and $\mathring{v}_{ik} = \{\bar{v}_{ik}, \hat{v}_{ik}\}$, with $\mathring{v}_{ik}$ denoting the natural parameter corresponding to v_{ik}. Also, the natural gradient of $\mathcal{L}_{\text{noisy}}(q, \boldsymbol{\theta})$ with respect to $\mathring{v}_{ik}$ is given by $\nabla\mathcal{L}'(\mathring{v}_{ik})$. As the model is conditionally conjugate [Hoffman *et al.*, 2013], $\nabla\mathcal{L}'(\mathring{v}_{ik}) = \mathring{v}^*_{ik} - \mathring{v}_{ik}$, where $\mathring{v}^*_{ik} = \{\bar{v}^*_{ik}, \hat{v}^*_{ik}\}$ is the value of $\mathring{v}_{ik}$ that maximizes Eq. (4.25). Therefore, update equation for $\mathring{v}_{ik}$ can be written as:

$$\mathring{v}_{ik}^{\text{new}} = \mathring{v}_{ik}^{\text{old}} + \eta_v^i(\mathring{v}^*_{ik} - \mathring{v}_{ik}^{\text{old}}) = (1 - \eta_v^i)\mathring{v}_{ik}^{\text{old}} + \eta_v^i \mathring{v}^*_{ik}, \tag{4.26}$$

where η_v^i is the step size corresponding to $\mathring{v}_{ik}$. Step sizes η_w^0, η_w^i and η_v^i are updated each time the corresponding parameters get updated using Robbins-Monro conditions [Hoffman *et al.*, 2013] which ensures convergence. In particular, let t_{w_0}, t_{w_i} and $t_{v_{ik}}$ be the number of times corresponding parameters get updated. Then the update rules can be written as

Algorithm 4 Online Variational Bayesian Factorization Machine (OVBFM)

Require: $\alpha, \sigma_0, \sigma_{w_{c_i}}, \sigma_{v_{c_ik}}, \eta^i \ \ \forall i, k$
Ensure: Randomly Initialize $\sigma_0^{'}$, $\mu_0^{'}$, $\sigma_{w_i}^{'}$, $\mu_{w_i}^{'}$, $\sigma_{v_{ik}}^{'}$, $\mu_{v_{ik}}^{'}$ $\forall i, k$
for $t = 1$ **to** M **do**
  for $s \in B$ **do**
    Compute $R_n \ \forall n \in s$
    // Update w_0's parameter
    for $i = 1$ **to** n **do**
      Update $\mathring{w}_0^{\text{avg}}$ using (4.31)
    end for
    Update $\mathring{w}_0$ using (4.29)
    Update R_n like algorithm 3 for current batch
    // Update w_i's parameter
    for $i = 1$ **to** D **do**
      for $n = 1$ **to** Ω_i **do**
        Update $\mathring{w}_i^{\text{avg}}$ using (4.32)
      end for
      Update $\mathring{w}_i$ using (4.29)
      Update R_n like algorithm 3 for current batch
    end for
    // Update v_{ik}'s parameter
    for $k = 1$ **to** K **do**
      for $i = 1$ **to** D **do**
        for $n = 1$ **to** Ω_i **do**
          Update $\mathring{v}_{ik}^{\text{avg}}$ using (4.33)
        end for
        Update $\mathring{v}_{ik}$ using (4.29)
        Update R_n like algorithm 3 for current batch
      end for
    end for
    Update η_w^0, η_w^i, η_v^i using (4.27) and (4.28)
    // Update hyperparameters
    $\alpha \leftarrow (1 - \eta_w^0)\alpha + \eta_w^0(\frac{|s|}{\sum_{n=1}^{|s|} R_n^2 + T_n})$
    $\sigma_0 \leftarrow (1 - \eta_w^0)\sigma_0 + \eta_w^0(\frac{1}{\mu_0^2 + \sigma_0})$
    for $i = 1$ **to** $|c|$ **do**
      $\sigma_{w_{c_i}} \leftarrow (1 - \eta_w^i)\sigma_{w_{c_i}} + \eta_w^i(\frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{w_j}^{'2} + \sigma_{w_j}^{'}\right)})$
    end for
    for $k = 1$ **to** K **do**
      for $i = 1$ **to** $|c|$ **do**
        $\sigma_{v_{c_ik}} \leftarrow (1 - \eta_v^i)\sigma_{v_{c_ik}} + \eta_v^i(\frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{v_{jk}}^{'2} + \sigma_{v_{jk}}^{'}\right)})$
      end for
    end for
  end for
end for

follows:

$$\eta_w^0 = (1 + t_{w_0})^{-\lambda}, \eta_w^i = (1 + t_{w_i})^{-\lambda} \ \forall i \in \{1, 2, \cdots, D\}, \tag{4.27}$$

$$\eta_v^i = (1 + t_{v_{ik}})^{-\lambda} \ \forall i \in \{1, 2, \cdots, D\} \text{ and } \forall k \in \{1, 2, \cdots, K\}, \tag{4.28}$$

where $\lambda \in (0.5, 1)$. For all the experiments, minimum value of λ produced best results. Therefore λ is set at 0.5.

To reduce the variance due to noisy estimate of the gradient, a mini-batch version is considered with a batch size of s number of points. To update a parameter, for example $\mathring{v}_{ik} = \{\bar{v}_{ik}^*, \hat{v}_{ik}^*\}$ are computed and stored for all the data instances with non-zero feature

values in the i^{th} column. The update of $\mathring{v}_{ik}$ can then be derived as follows:

$$\mathring{v}_{ik}^{\text{new}} = (1 - \eta_v^i)\mathring{v}_{ik}^{\text{old}} + \eta_v^i \mathring{v}_{ik}^{\text{avg}}, \tag{4.29}$$

where,

$$\mathring{v}_{ik}^{\text{avg}} = \sum_{n=1}^{n_i} \mathring{v}_{ik}^{*,n}/n_i. \tag{4.30}$$

Here n_i is the number of non-zero entries in the i^{th} column of the design matrix constructed from the current batch and $\mathring{v}_{ik}^{*,n}$ is the value of $\mathring{v}_{ik}^{*}$ produced when the n^{th} data point is considered. Detailed update equations of parameters set $\{\mathring{w}_0^*, \mathring{w}_i^*, \mathring{v}_{ik}^*\}$, which are used in Eq. (4.29) to calculate the variational parameters $\{\mathring{w}_0, \mathring{w}_i, \mathring{v}_{ik}\}$, are as follows.

- Update rule for the parameters of $\mathring{w}_0^* = \{\bar{w}_0^*, \hat{w}_0^*\}$ given the n^{th} data point is as follows:

$$\begin{aligned} \hat{w}_0^* &= \sigma_0 + N\alpha, \\ \bar{w}_0^* &= N\alpha(R_n + \mu_0^{'}) \end{aligned} \quad . \tag{4.31}$$

- Update rule for the parameters of $\mathring{w}_i^* = \{\bar{w}_i^*, \hat{w}_i^*\}$ given the n^{th} data point is as follows:

$$\begin{aligned} \hat{w}_i^* &= (\sigma_{w_{c_i}} + |\Omega_i|\alpha x_{ni}^2), \\ \bar{w}_i^* &= |\Omega_i|\alpha x_{ni}\left(R_n + x_{ni}\mu_{w_i}^{'}\right). \end{aligned} \tag{4.32}$$

- Update rule for the parameters of $\mathring{v}_{ik}^* = \{\bar{v}_{ik}^*, \hat{v}_{ik}^*\}$ given the n^{th} data point is as follows:

$$\begin{aligned} \hat{v}_{ik}^* &= \sigma_{v_{c_i k}} + |\Omega_i|\alpha x_{ni}^2\left(S_1(i,k)^2 + S_2(i,k)\right), \\ \bar{v}_{ik}^* &= |\Omega_i|\alpha x_{ni} S_1(i,k)\left(R_n + x_{ni}\mu_{v_{ik}}^{'} S_1(i,k)\right). \end{aligned} \tag{4.33}$$

Algorithm 4 describes the detail procedure of OVBFM. In each iteration of OVBFM, the dataset is partitioned into B random batches. Then in each iteration, OVBFM loops through these B batches sequentially. For a given batch, line 3-27 update all the natural parameters, and line 28-39 update all the model hyperparameters. These steps are then repeated for B batches which completes a full iteration of OVBFM. These steps are then repeated for M iterations.

Table 4.1: Description of the datasets.

Dataset	No. of User	No. of Movie	No. of Entries
Movielens 1m	6040	3900	**1m**
Movielens 10m	71567	10681	**10m**
Netflix	480189	17770	**100m**
KDD Music	1000990	624961	**263m**

4.4 Experiments

4.4.1 Dataset

In this section, empirical results on four real-world datasets are presented that validate the effectiveness of the proposed models. Except Netflix, all the other datasets are publicly available. The details of these datasets are provided in Table 4.1. For Movielens 1m and 10m datasets, the train-test split provided by Movielens is used for the experiments. For Netflix, the probe data is used as the test set. In case of KDD Music dataset, standard train-test split is used for analysis.

4.4.2 Methods of Comparison

VBFM and OVBFM are compared against MCMC-FM [C. Freudenthaler, 2011] and SGD-FM [Rendle, 2012]. A version of MCMC-FM is also considered as another baseline which allows 20 burn-in iterations for the sampler and is named as MCMC-Burnin-FM.

4.4.3 Parameter Selection and Experimental Setup

The variational parameters $\{\mu_0^{'}, \mu_{w_i}^{'}, \mu_{v_{ik}}^{'}\}$ are initialized using a standard normal distribution and $\{\sigma_0^{'}, \sigma_{w_i}^{'}, \sigma_{v_{ik}}^{'}\}$ are initialized to 0.02 for both VBFM and OVBFM. The parameters $\boldsymbol{\theta}$ of the model are initialized to 1.0 for both VBFM and OVBFM. Additionally, in OVBFM, all the η's are initialized to 1.0 and decayed further using Robbins-Monro sequence for all the experiments. The number of batches for OVBFM is chosen by cross validation. In MCMC-FM and MCMC-Burnin-FM, the parameters are initialized according to the values suggested in [C. Freudenthaler, 2011]. In SGD-FM, MCMC-FM and

MCMC-Burnin-FM, parameters in $\boldsymbol{Z}$ are initialized using a normal distribution with zero mean and standard deviation of value 0.01. As performance of SGD-FM is susceptible to the learning rate and regularization parameter, they are chosen by cross-validation. In Movielens 1m, 10m, and Netflix, the best performances are achieved with the following values of learning rate and regularization parameter – $(0.001, 0.01)$, $(0.0001, 0.01)$, and $(0.001, 0.01)$ respectively. For KDD music dataset, experiments are run for SGD-FM with three different learning rates 0.0001, 0.00005, and 0.00001, but the regularization parameter is kept fixed at 0.01. Each of the algorithm are run for 100 iterations.

In past, it is shown that increasing the number of latent factor improves the RMSE [Koren, 2009; Salakhutdinov and Mnih, 2008]. Hence, it is necessary to investigate how the models work with different values of K. So three sets of experiments are run for each of Movielens 1m, 10m, and Netflix datasets corresponding to three different values of $K \in \{20, 50, 100\}$. As we run experiments on an Intel I5 machine with 16GB RAM, employing batch algorithms (MCMC-FM, MCMC-Burnin-FM, and VBFM) on KDD music data is not possible due to memory limitation. Hence, for KDD music dataset, we only compare performances of SGD-FM and OVBFM for $K = 20$ and $K = 50$. All of the proposed and baseline methods are allowed to run for 100 iterations. In case of MCMC-Burnin-FM, 20 burn-in iterations are followed by 80 collection iterations. Root Mean Square Error (RMSE) [Koren *et al.*, 2009] is used as the evaluation metric for all the experiments.

4.4.4 Results

Left, middle and right columns in Figure 4.2 show results for $K = 20$, 50, and 100 respectively on Movielens 1m, 10m and Netflix datasets. In Figure 4.3, left and right columns show results on KDD music dataset with $K = 20$ and 50 respectively. In all the plots, $x-$axis represents the number of iterations and the $y-$axis presents the RMSE value. Each iteration of VBFM and MCMC-FM takes almost equal time. However, SGD-FM is faster than VBFM and MCMC-FM as it is online algorithm. In case of OVBFM, we consider mini batch. hence, it is slower than SGD-FM but faster than VBFM and MCMC-FM.

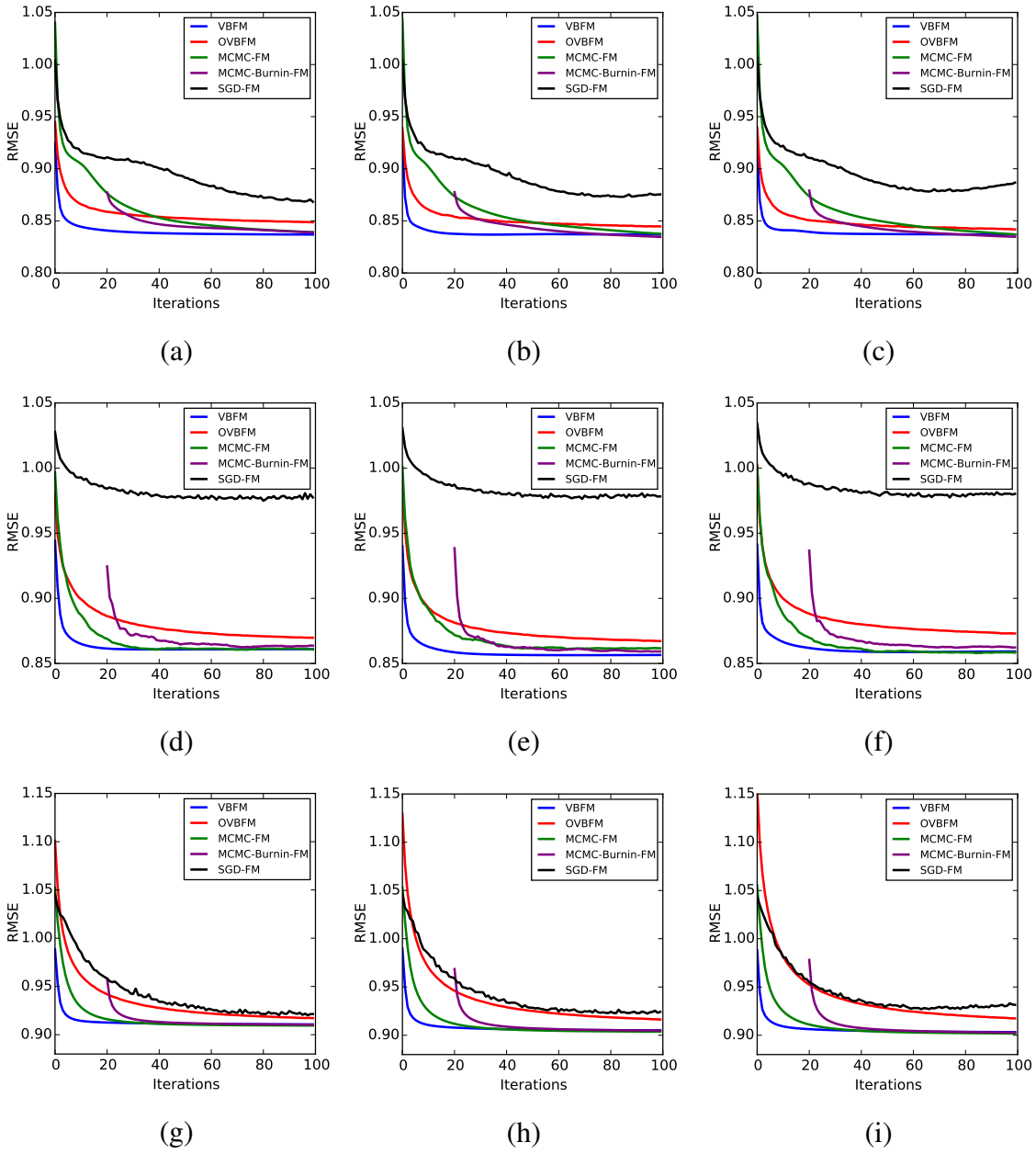


Figure 4.2: Left, middle, and right columns correspond to the results for $K = 20$, 50, and 100 respectively. {a,b,c}, {d,e,f}, and {g,h,i} are results on Movielens 1m, Movielens 10m, and Netflix datasets respectively.

Though the over-all performance varies for VBFM, MCMC-FM, and MCMC-Burnin-FM depending on the datasets and the number of latent factors used, the differences among their asymptotic behaviors are negligible. For all the experiments on Movielens 1m, 10m, and Netflix dataset, VBFM is found to converge faster than both MCMC-FM and MCMC-Burnin-FM. The convergence of VBFM is faster probably due to the fact that MCMC-FM is a hierarchical model. On the other hand, VBFM is a single level model where

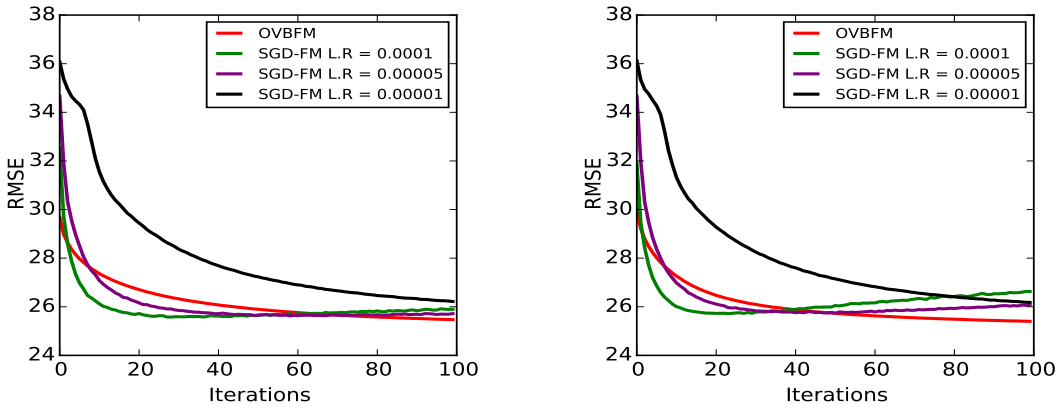


Figure 4.3: Left and right columns show results on KDD music dataset for $K = 20$ and 50 respectively.

hyperpriors are not considered. For all the experiments on Movielens 1m, 10m, and Netflix dataset, OVBFM performs much better than SGD-FM. Also it is evident from the graph that SGD overfits the data quite often. In KDD music dataset, OVBFM performs better than SGD-FM and the gap in RMSE is more significant for $K = 50$. Overfitting with SGD is more problematic with higher values of K in the KDD music dataset. Also for SGD, there is an additional difficulty of tuning the learning rate for each dataset. For very small values of learning rate, SGD underfits the data and for very large values it overfits the data. On the contrary, we tried OVBFM with different learning rates and the performance of OVBFM is quite robust with respect to the learning rate. In Netflix dataset, for VBFM and MCMC-FM $K = 100$ gives approximately 1% lift over $K = 20$. For other datasets the lift is less. SGD-FM performs worse with higher K due to overfitting and OVBFM gives small lift in some cases. Note, we experimented with different values of K to be consistent with the literature [Koren, 2009; Salakhutdinov and Mnih, 2008].

4.5 Summary

We have proposed the Variational Bayesian Factorization Machine (VBFM) which is a scalable variational Bayesian approach for Factorization Machine (FM). VBFM converges faster than the existing state-of-the-art Markov chain Monte Carlo (MCMC) based Gibbs sampling inference algorithm for FM while providing similar performance. Additionally,

the Online Variational Bayesian Factorization Machine (OVBFM) is introduced, which uses SGD for maximizing the lower bound obtained from the variational approximation, and performs much better than the existing online algorithm of FM that uses SGD.

CHAPTER 5

NONPARAMETRIC POISSON FACTORIZATION MACHINE

In this chapter, we develop the Nonparametric Poisson Factorization Machine (NPFM), which models count data using the Poisson distribution, provides both modeling and computational advantages for sparse data. The ideal number of latent factors is estimated from the data itself. We also consider a special case of NPFM, the Parametric Poisson Factorization Machine (PPFM), that considers a fixed number of latent factors. Both NPFM anf PPFM have linear time and space complexity with respect to the number of non-zero observations. Extensive experiments on four different movie review datasets show that both NPFM and PPFM outperform two competitive baseline methods by huge margin.

5.1 Introduction

Factorization models have received extensive attention in the data mining community recently for certain problems characterized by high-dimensional, sparse matrices, due to their simplicity, prediction quality, and scalability. One of the most successful domains for factorization models have been Recommender Systems(RSs). Perhaps the most well studied factorization model is matrix factorization [Srebro and Jaakkola, 2003; Koren *et al.*, 2009; Salakhutdinov and Mnih, 2007, 2008; Gopalan *et al.*, 2015] using the Frobenius norm as the loss function. Tensor factorization [Xiong *et al.*, 2010; Ho *et al.*, 2014; Chi and Kolda, 2012] methods have also been developed for multi-modal data. Several specialized factorization models have been proposed specific to particular problems, for instance: SVD++ [Koren, 2008], TimeSVD++ [Koren, 2009], Factorizing Personalized Markov Chains (FPMC) [Rendle *et al.*, 2011*a*], and Bayesian Probabilistic Tensor Factorization (BPTF) [Xiong *et al.*, 2010]. Also many learning and inference methods have been

studied for factorization models, for example: stochastic gradient descent (SGD) [Koren *et al.*, 2009], alternating least-squares [Zhou *et al.*, 2008], variational Bayes [Lim and Teh, 2007; Kim and Choi, 2014; Silva and Carin, 2012], and Markov Chain Monto Carlo (MCMC) Gibbs sampling inference [Salakhutdinov and Mnih, 2008].

In more challenging prediction scenarios where additional "side-information" is available and/or higher order features may be needed, new challenges due to feature engineering needs arise. The side-information may include user specific features such as age, gender, demographics, and network information and item specific information such as product descriptions. While interaction terms are typically desired in such scenarios, the number of such terms grows very quickly. This dilemma is cleverly addressed by the Factorization Machine (FM) [Rendle, 2010], which combines high prediction quality of factorization models with the flexibility of feature engineering. FM represents data as real-valued features like standard machine learning approaches, such as Support Vector Machines (SVMs), and uses interactions between each pair of variables as well but constrained to a low-dimensional latent space. By restricting the latent space, the number of parameters needed to be determined is kept manageable. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [Koren *et al.*, 2009], SVD++ [Koren, 2008], TimeSVD++ [Koren, 2009], Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme, 2010], and factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2011*a*].

A stochastic gradient descent (SGD) based algorithm SGD-FM [Rendle, 2010] is usually used to learn FM. Though SGD is scalable and enjoys local convergence guarantees [Sato, 2001], it requires manual tuning of learning rate and regularization parameters, and may overfit the data. Alternative methods to solve FM include MCMC-FM [C. Freudenthaler, 2011] which provides state-of-the-art performance using Markov chain Monte Carlo (MCMC) based Gibbs sampling as the inference mechanism and avoids expensive manual tuning of the learning rate and regularization parameters. However, MCMC-FM assumes that the observations are generated from a Gaussian distribution which, obviously, is not a good fit for count data. Additionally, for both SGD-FM and MCMC-FM, one needs to solve an expensive model selection problem to identify the optimal number of latent fac-

tors. We note that alternative models for count data have recently emerged that use discrete distributions, provide better interpretability and scale only with the number of non-zero elements [Gopalan *et al.*, 2014*b*; Zhou and Carin, 2015; Zhou *et al.*, 2012; Acharya *et al.*, 2015].

This chapter proposes a Nonparametric Poisson Factorization Machine (NPFM) to overcome the limitations of the existing inference techniques for FM mentioned above. The specific advantages of NPFM include:

- NPFM provides a more interpretable model with a better fit for count dataset.

- NPFM is a nonparametric approach and avoids the costly model selection procedure by automatically finding the number of latent factors suitable for modeling the pairwise interaction matrix in FM.

- NPFM takes advantages of the Poisson distribution [Gopalan *et al.*, 2015] and considers only sampling over the non-zero entries. On the other hand, existing FM methods, which assume a Gaussian distribution, must iterate over both positive and negative samples in the implicit setting. Such iteration is expensive for large datasets and often needs to be solved using a costly positive and negative data sampling approach instead. NPFM can take advantage of natural sparsity of the data which existing inference technique of FM fails to exploit.

The remainder of the chapter is structured as follows. Section 5.2 presents pertinent background and related work. A detailed description of NPFM and associated inference mechanisms is provided in Section 5.3. Section 5.4 demonstrates the performance of NPFM on both simulated and real world datasets. Finally, the conclusions are summarized in Section 5.5.

5.2 Background and Related Work

This section presents the related literature and the background materials that are useful for understanding the framework described in Section 5.3.

5.2.1 Negative Binomial Distribution

The negative binomial (NB) distribution $m \sim \text{NB}(r, p)$, with probability mass function (PMF) $P(M = m) = \frac{\Gamma(m+r)}{m!\Gamma(r)} p^m (1-p)^r$ for $m \in \mathbb{Z}$, can be augmented into a gamma-Poisson construction as $m \sim \text{Pois}(\lambda),\ \lambda \sim \text{Gamma}(r, p/(1-p))$, where Pois and Gamma represent Poisson and gamma distribution respectively, and the gamma distribution is parameterized by its shape r and scale $p/(1-p)$. It can also be augmented under a compound Poisson representation as $m = \sum_{t=1}^{l} u_t, u_t \overset{iid}{\sim} \text{Log}(p), l \sim \text{Pois}(-r\ln(1-p))$, where $u \sim \text{Log}(p)$ is the logarithmic distribution [Johnson *et al.*, 2005]. Consequently, we have the following Lemma.

Lemma 5.2.1 ([Zhou and Carin, 2012])**.** *If $m \sim \text{NB}(r, p)$ is represented under its compound Poisson representation, then the conditional posterior of l given m and r has PMF:*

$$P(l = j|m, r) = \frac{\Gamma(r)}{\Gamma(m+r)} |s(m, j)| r^j,\ j = 0, 1, \cdots, m, \tag{5.1}$$

where $|s(m, j)|$ are unsigned Stirling numbers of the first kind. We denote this conditional posterior as $(l|m, r) \sim \text{CRT}(m, r)$, a Chinese restaurant table (CRT) count random variable, which can be generated via $l = \sum_{n=1}^{m} z_n, z_n \sim \text{Bernoulli}(r/(n-1+r))$.

Lemma 5.2.2. *Let $X = \sum_{k=1}^{K} x_k$, $x_k \sim \text{Pois}(\zeta_k)\ \forall k$, and $\zeta = \sum_{k=1}^{K} \zeta_k$. If $(y_1, \cdots, y_K|X) \sim \text{Mult}(X, \zeta_1/\zeta, \cdots, \zeta_K/\zeta)$ and $X \sim \text{Pois}(\zeta)$, then the following holds:*

$$P(X, x_1, \cdots, x_K) = P(X, y_1, \cdots, y_K). \tag{5.2}$$

Lemma 5.2.3. *If $x_i \sim \text{Pois}(m_i\lambda),\ \lambda \sim \text{Gamma}(r, 1/c)$, then $x = \sum_i x_i \sim \text{NB}(r, p)$, where $p = (\sum_i m_i)/(c + \sum_i m_i)$.*

Lemma 5.2.4. *If $x_i \sim \text{Pois}(m_i\lambda),\ \lambda \sim \text{Gamma}(r, 1/c)$, then*

$$(\lambda|\{x_i\}, r, c) \sim \text{Gamma}\left(r + \sum_i x_i, \frac{1}{c + \sum_i m_i}\right). \tag{5.3}$$

Lemma 5.2.5. *If $r_i \sim \text{Gamma}(a_i, 1/b)\ \forall i,\ b \sim \text{Gamma}(c, 1/d)$, then we have:*

$$(b|\{r_i, a_i\}, c, d) \sim \text{Gamma}\left(\sum_i a_i + c, \frac{1}{\sum_i r_i + d}\right). \tag{5.4}$$

The proofs of Lemmas 5.2.3, 5.2.4 and 5.2.5 follow from the definitions of gamma, Poisson and negative binomial distributions.

Lemma 5.2.6. *If* $x_i \sim \text{Pois}(m_i r_2),\ r_2 \sim \text{Gamma}(r_1, 1/d),\ r_1 \sim \text{Gamma}(a, 1/b),$ *, then* $(r_1|-) \sim \text{Gamma}(a + \ell, 1/(b - log(1-p)))$ *where* $(\ell|x, r_1) \sim \text{CRT}(\sum_i x_i, r_1), p = \sum_i m_i/(d + \sum_i m_i)$.

The proof and illustration of Lemma 5.2.6 can be found in Section 3.3 of [Acharya *et al.*, 2015].

5.2.2 Gamma Process

The gamma process [Zhou and Carin, 2015] $G \sim \Gamma\text{P}(c, G_0)$ is a completely random measure defined on the product space $\mathbb{R}_+ \times \Omega$, with scale parameter $\frac{1}{c}$ and a finite and continuous base measure G_0 over a complete separable metric space Ω, such that $G(A_i) \sim \text{Gamma}(G_0(A_i), 1/c)$ for each $A_i \subset \Omega$. The Lévy measure of the gamma process can be expressed as $\nu(drd\omega) = r^{-1}e^{-cr}dr(d\omega)$. Since the Poisson intensity $\nu^+ = \nu(\mathbb{R}_+ \times \Omega) = \infty$ and the value of $\int\int_{\mathbb{R}_+\times\Omega} r\nu(drd\omega)$ is finite, a draw from the gamma process consists of countably infinite atoms, which can be expressed as follows:

$$G = \sum_{k=1}^{\infty} r_k \delta_{\omega_k},\ (r_k, \omega_k) \overset{iid}{\sim} \pi(drd\omega),\ \pi(drd\omega)\nu^+ \equiv \nu(drd\omega). \tag{5.5}$$

5.2.3 Poisson Factor Analysis

Since the pairwise interaction matrix in NPFM is modeled using a Poisson factorization, some discussion on Poisson factor analysis is necessary. A large number of discrete latent variable models for count matrix factorization can be united under Poisson factor analysis (PFA) [Zhou *et al.*, 2012; Zhou and Carin, 2015; Acharya *et al.*, 2015], which factorizes a count matrix $\mathbf{Y} \in \mathbb{Z}^{D\times V}$ under the Poisson likelihood as $\mathbf{Y} \sim \text{Pois}(\mathbf{\Phi\Theta})$, where

$\mathbf{\Phi} \in \mathbb{R}_+^{D \times K}$ is the factor loading matrix or dictionary, $\mathbf{\Theta} \in \mathbb{R}_+^{K \times V}$ is the factor score matrix. A wide variety of algorithms, although constructed with different motivations and for distinct problems, can all be viewed as PFA with different prior distributions imposed on $\mathbf{\Phi}$ and $\mathbf{\Theta}$. For example, non-negative matrix factorization [Cemgil, 2009], with the objective to minimize the Kullback-Leibler divergence between $\mathbf{N}$ and its factorization $\mathbf{\Phi\Theta}$, is essentially PFA solved with maximum likelihood estimation. LDA [Blei *et al.*, 2003] is equivalent to PFA, in terms of both block Gibbs sampling and variational inference [Gopalan *et al.*, 2014*b*, 2015], if Dirichlet distribution priors are imposed on both $\boldsymbol{\phi}_k \in \mathbb{R}_+^D$, the columns of $\mathbf{\Phi}$, and $\boldsymbol{\theta}_k \in \mathbb{R}_+^V$, the columns of $\mathbf{\Theta}$. The gamma-Poisson model [Canny, 2004; Titsias, 2007] is PFA with gamma priors on $\mathbf{\Phi}$ and $\mathbf{\Theta}$. A family of negative binomial (NB) processes, such as the beta-NB [Zhou *et al.*, 2012] and gamma-NB processes [Zhou and Carin, 2012, 2015], impose different gamma priors on $\{\theta_{vk}\}$, the marginalization of which leads to differently parameterized NB distributions to explain the latent counts. Both the beta-NB and gamma-NB process PFAs are nonparametric Bayesian models that allow K to grow without limits [Hjort, 1990].

5.3 Nonparametric Poisson Factorization Machine

5.3.1 Model

Consider a training data consisting of N tuples of the form $(\boldsymbol{x}_n, y_n)_{n=1}^N$ where $\boldsymbol{x}_n$ is the feature representation as shown in Figure 2.1 and y_n is the associated response variable for the n^{th} training instance. In NPFM, the response variable $y_n \in \mathbb{Z}$ is assumed to be linked to the covariate $\boldsymbol{x}_n \in \mathbb{R}_+^D$ as:

$$y_n \sim \text{Pois}\left(w_0 + \boldsymbol{w}^\intercal \boldsymbol{x}_n + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni} x_{nj} \sum_{k=1}^{\infty} r_k v_{ik} v_{jk}\right). \tag{5.6}$$

According to the standard terminology in the literature of recommender systems, w_0 denotes the global bias and is sampled as $w_0 \sim \text{Gamma}(\alpha_0, 1/\beta_0)$. $\boldsymbol{w} = (w_i)_{i=1}^D \in \mathbb{R}_+^D$ and w_i indicates the strength of the i^{th} variable (the i^{th} column from Figure 2.1) and

can be thought of as the bias corresponding to the i^{th} feature. w_i is modeled as $w_i \sim \text{Gamma}(\alpha_i, 1/\beta_i)\ \forall i \in \{1, 2, \cdots, D\}$. A gamma process $G \sim \Gamma\text{P}(c, G_0)$ is further introduced, a draw from which is expressed as $G = \sum_{k=1}^{\infty} r_k \delta_{\boldsymbol{v}_k}$, where $\boldsymbol{v}_k = (v_{ik})_{i=1}^{D}$ is an atom drawn from a D-dimensional base distribution as $\boldsymbol{v}_k \sim \prod_{i=1}^{D} \text{Gamma}(\zeta_i, 1/\delta_k)$ and $r_k = G(\boldsymbol{v}_k)$ is the associated weight. The i^{th} column in Figure 2.1 is associated with the infinite dimensional latent vector $\boldsymbol{v_i}$. The objective is to learn a distribution over the weights w_0, $\boldsymbol{w}$, $\{r_k, \boldsymbol{v}_k\}$ based on the training observations. To complete the generative process, gamma priors are imposed on the parameters as:

$$\alpha_0 \sim \text{Gamma}(a_0, 1/b_0),\ \beta_0 \sim \text{Gamma}(c_0, 1/d_0),$$

$$\alpha_i \sim \text{Gamma}(a_i, 1/b_i),\ \beta_i \sim \text{Gamma}(c_i, 1/d_i),\ \zeta_i \sim \text{Gamma}(e_i, 1/f_i),$$

$$\delta_k \sim \text{Gamma}(g_k, 1/h_k).$$

The pairwise interaction matrix $\boldsymbol{W}$ (please refer to Section 2.2.4 for more details) is modeled little differently in NPFM compared to other existing formulations of FM. In NPFM, $\boldsymbol{W}$ is factored as $w_{ij} \sim \text{Pois}\left(\sum_k r_k v_{ik} v_{jk}\right)$, where r_k denotes the strength of the k^{th} latent factor and v_{ik} denotes the affinity of the i^{th} variable towards the k^{th} latent factor. Note that such assumptions have been successfully used already for network analysis [Zhou, 2015] and count data modeling [Acharya *et al.*, 2015] and is similar to using eigen decomposition of the matrix $\boldsymbol{W}$ with integer entries. However, unlike in eigen decomposition, the factors v_{ik}'s are neither normalized nor do they form an orthogonal set of basis vectors. Of course, by sampling the entries of $\boldsymbol{W}$ from a Poisson distribution, we do restrict these entries to integers, which is yet another departure from the standard formulation of FM. However, the empirical results reveal that this is not at all an unreasonable assumption. The gamma process $G = \sum_{k=1}^{\infty} r_k \delta_{\boldsymbol{v}_k}$ allows to estimate the ideal number of latent factors from the data itself, without any need to cross-validate the performance with varying number of latent factors.

The factor specific variable r_k adds more flexibility to the model. For example, if there is a need for modeling temporal count datasets, such as in a recommender system that evolves over time, r_k's can be linked across successive time stamps using a gamma

Markov chain [Acharya *et al.*, 2015]. This would imply that a certain combination of user-item pair changes its characteristics over time. Since in practical recommender systems, such evolution is very natural, we prefer to maintain these latent variables. In Section 5.3.3, we illustrate PPFM, a simplified version of NPFM, which does not use the variables r_k's at all. In Section 5.4, we see that the performances of NPFM and PPFM are comparable, implying that the added flexibility does not hurt the predictive performance.

5.3.2 Inference Using Gibbs Sampling

Though NPFM supports countably infinite number of latent factors, in practice, it is impossible to instantiate all of them. Instead of marginalizing out the underlying stochastic process [Blackwell and MacQueen, 1973] or using slice sampling [Walker, 2007] for non-parametric modeling, for simplicity, a finite approximation of the infinite model is considered by truncating the number of latent factors K, by letting $r_k \sim \text{Gamma}(\gamma_0/K, 1/c)$. Such an approximation approaches the original infinite model as K approaches infinity. Further, gamma priors are imposed on both γ_0 and c as:

$$\gamma_0 \sim \text{Gamma}(e_0, 1/f_0),\ c \sim \text{Gamma}(g_0, 1/h_0).$$

With such approximation, the graphical model of NPFM is displayed in Figure 5.1.

For each $(\boldsymbol{x}_n, y_n)$, consider a vector of latent count variables $\boldsymbol{z}_n$, which is assumed to consist of three parts:

$$\boldsymbol{z}_n = \left(z_{1n}, (z_{2ni})_{i=1}^{D}, (z_{3nijk})_{(i,j):i<j;k}\right),$$

where $z_{1n} \sim \text{Pois}(w_0)$, $z_{2ni} \sim \text{Pois}(x_{ni}w_i)$ and $z_{3nijk} \sim \text{Pois}(x_{ni}x_{nj}r_k v_{ik}v_{jk})$. These latent variables are incorporated to make the model conjugate [Zhou and Carin, 2012]. As a sum of Poisson random variables is itself a Poisson with rate equal to the sum of the rates, one gets the following:

$$y_n = z_{1n} + \sum_{i=1}^{D} z_{2ni} + \sum_{(i,j):i<j;k} z_{3nijk}.$$

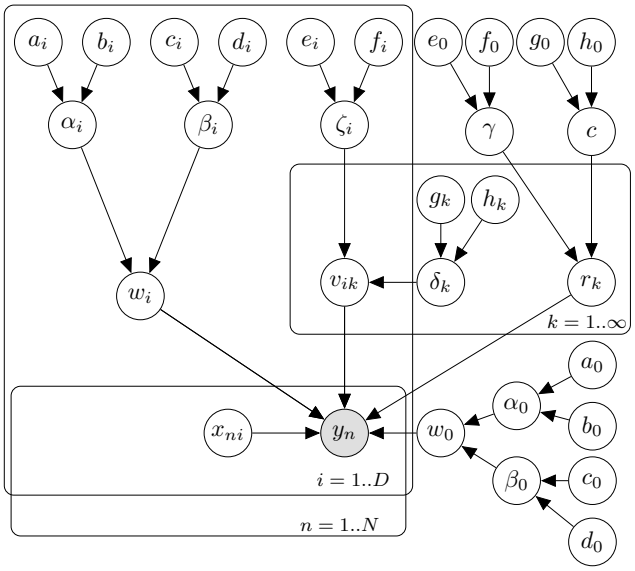


Figure 5.1: Graphical Model representation of NPFM where y_n are target values drawn from a Poisson distribution; all other intermediate variables are drawn from gamma distributions.

Note that when $y_n = 0$, $\boldsymbol{z}_n = 0$ with probability 1. Hence, the NPFM inference procedure needs to consider $\boldsymbol{z}_n$ only when $y_n > 0$. Using Lemma 5.2.2, the conditional posterior of these latent counts can be expressed as follows:

$$\boldsymbol{z}_n|- \sim \text{Mult}\left(\frac{w_0 + (w_i x_{ni})_{i=1}^{D}, (x_{ni}x_{nj}r_k v_{ik}v_{jk})_{(i,j):i<j;k}}{w_0 + \sum_i w_i x_{ni} + \sum_{(i,j):i<j} x_{ni}x_{nj}\sum_{k=1}^{K} r_k v_{ik}v_{jk}}; y_n\right). \tag{5.7}$$

Sampling of w_0 : Using Lemma 5.2.4, the conditional posterior of w_0 can be expressed as:

$$w_0|- \sim \text{Gamma}\left(\alpha_0 + z_{1.}, 1/(\beta_0 + N)\right). \tag{5.8}$$

Sampling of w_i : Using Lemma 5.2.4, the conditional posterior of w_i can be expressed as:

$$w_i|- \sim \text{Gamma}\left(\alpha_i + z_{2.i}, 1/(\beta_i + x_{.i})\right). \tag{5.9}$$

Sampling of v_{ik} : Using Lemma 5.2.4, the conditional posterior of v_{ik} can be expressed as:

$$v_{ik}|- \sim \text{Gamma}\left(\zeta_i + z_{3.ik}, 1/(\delta_k + \sum_{n=1}^{N} r_k x_{ni} \sum_{j \neq i} x_{nj} v_{jk}\right). \tag{5.10}$$

Sampling of r_k : Using Lemma 5.2.4, the conditional posterior of r_k can be expressed as:

$$r_k|- \sim \text{Gamma}\left(\gamma_0/K + q_k, 1/(c + s_k)\right), \tag{5.11}$$
$$q_k = \sum_{n=1}^{N} \sum_{(i,j):i<j} z_{3nijk},\ s_k = \sum_{n=1}^{N} \sum_{(i,j):i<j} x_{ni} x_{nj} v_{ik} v_{jk}.$$

Sampling of β_0 : Using Lemma 5.2.5, the conditional posterior of β_0 can be expressed as:

$$\beta_0|- \sim \text{Gamma}(c_0 + \alpha_0, 1/(d_0 + w_0). \tag{5.12}$$

Sampling of β_i : Using Lemma 5.2.5, the conditional posterior of β_i can be expressed as:

$$\beta_i|- \sim \text{Gamma}(c_i + \alpha_i, 1/(d_i + w_i)). \tag{5.13}$$

Sampling of δ_k : Using Lemma 5.2.5, the conditional posterior of δ_k can be expressed as:

$$\delta_k|- \sim \text{Gamma}(g_k + \zeta_., 1/(h_k + v_{.k})). \tag{5.14}$$

Sampling of c : Using Lemma 5.2.5, the conditional posterior of c can be expressed as:

$$c|- \sim \text{Gamma}(\gamma_0 + g_0, 1/(h_0 + r_.)). \tag{5.15}$$

Sampling of α_0 : Straightforward sampling of α_0 is not possible as this is the shape parameter of a gamma distribution and the prior and the likelihood are not conjugate. However, from the generative assumptions, we have $z_{1.} \sim \text{Pois}(Nw_0)$. Using Lemma 5.2.3, one can have $w_0 \sim \text{NB}(\alpha_0, N/(N + \beta_0))$. Now augment l_0 as $l_0 \sim \text{CRT}\left(z_{1.}, \alpha_0\right)$, and using Lemma 5.2.6 one can sample:

$$\alpha_0|- \sim \text{Gamma}(a_0 + l_0, 1/(b_0 - \log(\beta_0/(\beta_0 + N)))). \tag{5.16}$$

Sampling of α_i : Straightforward sampling of α_i is not possible as this is the shape parameter of a gamma distribution and the prior and the likelihood are not conjugate. However, from the generative assumptions, we have $z_{2.i} \sim \text{Pois}(w_i m_i)$, where $m_i = \sum_n x_{ni}$. Using Lemma 5.2.3, one can have $w_i \sim \text{NB}(\alpha_i, m_i/(m_i + \beta_i))$. Now augment l_i as $l_i \sim \text{CRT}\,(z_{2.i}, \alpha_i)$, and using Lemma 5.2.6 one can sample:

$$\alpha_i|- \sim \text{Gamma}(a_i + l_i, 1/(b_i - \log(1 - p_i))), \tag{5.17}$$

where $p_i = m_i/(\beta_i + m_i)$.

Sampling of ζ_i : Since $z_{3.ik} \sim \text{Pois}(m_{ik} v_{ik})$ where $m_{ik} = \sum_n x_{ni} r_k \sum_{j \neq i} x_{nj} v_{jk}$ and $v_{ik} \sim \text{Gamma}(\zeta_i, 1/\delta_k)$, integrating out v_{ik} and using Lemma 5.2.3, one has $z_{3.ik} \sim \text{NB}\,(\zeta_i, p_{ik})$ $\forall k \in \{1, 2, \cdots, K\}$ where $p_{ik} = \frac{m_{ik}}{\delta_k + m_{ik}}$. Augment $\ell_{ik} \sim \text{CRT}\,(z_{3.ik}, \zeta_i)$ and using Lemma 5.2.6 sample ζ_i as follows:

$$\zeta_i|- \sim \text{Gamma}(e_i + \sum_k \ell_{ik}, 1/(f_i - \sum_k \log(1 - p_{ik}))). \tag{5.18}$$

Sampling of γ_0 : Since $z_{3\ldots k} \sim \text{Pois}(r_k m_k)$ where $m_k = \sum_{n,\{(i,j):i<j\}} x_{ni} x_{nj} v_{ik} v_{jk}$ and $r_k \sim \text{Gamma}(\gamma_0/K, 1/c)$, integrating out r_k and using Lemma 5.2.3, one has $z_{3..k} \sim \text{NB}\,(\gamma_0/K, p_k)$ where $p_k = m_k/(c + m_k)$. Augment $\ell_k \sim \text{CRT}\,(z_{3\ldots k}, \gamma_0/K)$ and using Lemma 5.2.6 sample,

$$\gamma_0|- \sim \text{Gamma}(e_0 + \sum_k \ell_k, 1/(f_0 - 1/K \sum_k \log(1 - p_k))). \tag{5.19}$$

5.3.3 Parametric Version

We also consider a special case of NPFM, Parametric Poisson Factorization Machine (PPFM), as a baseline to compare against. The key difference between NPFM and PPFM is that in NPFM, one does not need to tune over the number of latent factors. Even with a finite approximation of NPFM, it is sufficient to set K at a high value and the inference itself predicts the ideal number of latent factors from the data. On the other hand, in PPFM, one needs to choose the number of latent factors by a cross-validation process which is

time-consuming and computationally intensive. Additionally, to make the baseline comparable against other existing formulations of FM (see Eq. (2.17)), the term r_k is left out. To be more precise, in PPFM, we consider the following generative process for the label y_n:

$$y_n \sim \text{Pois}\left(w_0 + \boldsymbol{w}^\intercal \boldsymbol{x} + \sum_{i=1}^{D}\sum_{j=i+1}^{D} x_{ni}x_{nj}\sum_{k=1}^{K} v_{ik}v_{jk}\right). \tag{5.20}$$

Here, $w_0 \sim \text{Gamma}(\alpha_0, 1/\beta_0)$, $\boldsymbol{w} = (w_i)_{i=1}^{D} \in \mathbb{R}_+^D$ and $w_i \sim \text{Gamma}(\alpha_i, 1/\beta_i)\ \forall i \in \{1, 2, \cdots, D\}$. Also $v_{ik} \sim \text{Gamma}(\zeta_i, 1/\delta_k)$. Similar to NPFM, gamma prior are imposed on other parameters as:

$$\alpha_i \sim \text{Gamma}(a_i, 1/b_i),\ \beta_i \sim \text{Gamma}(c_i, 1/d_i),\ \zeta_i \sim \text{Gamma}(e_i, 1/f_i),$$

$$\alpha_0 \sim \text{Gamma}(a_0, 1/b_0),\ \beta_0 \sim \text{Gamma}(c_0, 1/d_0),\ \delta_k \sim \text{Gamma}(g_k, 1/h_k).$$

Sampling equations for these parameters are similar to those of NPFM and follow from the Lemmas listed in Section 5.2.1.

5.3.4 Prediction

Let $\boldsymbol{\theta}$ denote the set of all the variables that are sampled in NPFM, other than the $\boldsymbol{z}_n$'s. In the training phase, we store the average values of these variables from T number of collection iterations. While predicting on the held-out set or missing entires, one just needs to sample the $\boldsymbol{z}_n$'s according to Eq. (5.7) given $\boldsymbol{X}$ and $\boldsymbol{\theta}$ fixed at the value obtained from the training phase. One should note that, while training, the calculation of the summary statistics gets affected by the presence of missing/held-out entries. Such adaptation affects directly the sampling of variables which are used in generating $\boldsymbol{z}_n$ according to Eq. (5.7), such as w_0, w_i's, r_k's and v_{ik}'s. In the updates of these variables in the training phase, one just needs to exclude the contribution from the missing entries and the equations work out similar to the case of sampling without missing entries.

5.3.5 Tme and Space Complexity

NPFM has linear time complexity with respect to the number of non-zero training instances. If S is the number of non-zero training instances in the data, maximum number of latent factors used be K, and D be the feature dimension, then NPFM has time complexity of O(SKD). Direct implementation NPFM is infeasible because one needs to store all the latent count variables for all the observations in training set which leads to a space complexity of O(D^2SK), which is clearly prohibitive for large datasets like Netflix with 100 million observations. To avoid such space complexity, the implementation does not store the latent count variables at all, but instead stores the summary statistics required in the updates for the Gibbs sampler, such as $z_{1.}$, $z_{2.i}$, and $z_{3.ik}$. This representation helps maintain space complexity of O(DK).

5.4 Experimental Results

We first validate the NPFM model using a simple synthetic dataset. Using a count matrix with two clearly separable clusters we show that NPFM recovers the clusters accurately.

5.4.1 Generating Synthetic Data

We apply NPFM on a synthetically generated count matrix with 60 users and 90 items. The whole matrix is divided into four parts with indices ranges as $([0, 20), [0, 30))$, $([0, 20), [30, 90))$, $([20, 60), [0, 30))$, and $([20, 60), [30, 90))$. We populate the first and fourth parts by 5s, and the other two parts by 0s. So 44% entries in the matrix are 0. Then, we randomly choose 20% entries as held-out set. First subplot in Figure 5.2 shows the count matrix generated by this process, where the black dots represent the missing entries. The red and blue areas are separate clusters with red and blue denoting a rating of 5 and 0 respectively.

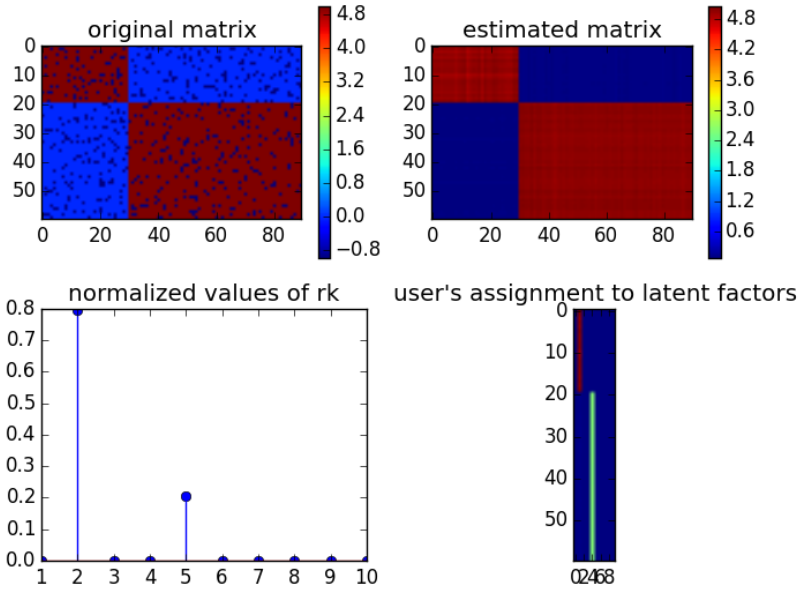


Figure 5.2: Results of NPFM on Synthetic dataset. In both first and second sub-plot, x-axis and y-axis represent users and items respectively. First represent the original count matrix whereas second one is the estimated matrix using the NPFM. Third sub-plot x-axis and y-axis are latent dimension and normalized value assigned to a latent dimension. Fourth subplot shows users' assignment to latent factors.

5.4.2 Simulation

We apply NPFM on the synthetically generated matrix, with maximum latent dimension set to 10. Figure 5.2 shows the output of NPFM on this synthetic count matrix. Subplot 2 in Figure 5.2 shows the matrix estimated by NPFM. It is evident from Figure 5.2 that NPFM is able to recover the count data accurately. Further, Subplot 3 shows the weight assigned to different r_k's. As there are only two clusters most of the mass is assigned to just two factors. The last subplot represents the user's assignment to the latent factors. Thus this experiment demonstrates that NPFM is not only able to recover the count data but also model the latent structure present in the data effectively.

Table 5.1: Description of the datasets.

Dataset	No. of User	No. of Movie	No. of Entries
Movielens 100k	943	1682	**100k**
Movielens 1m	6040	3900	**1m**
Movielens 10m	71567	10681	**10m**
Netflix	480189	17770	**100m**

5.4.3 Real World Datasets

We now validate both the performance and the runtime of our model on four different movie rating datasets[1,2] popularly used in the recommender system literature. The detailed statistics of these datasets are given in Table 5.1.

5.4.4 Metric

We evaluate our method on both accuracy and time. To evaluate accuracy, we consider the recommendation problem as a ranking problem and use Mean Precision, Mean Recall, and F-measure as the metrics [Gopalan *et al.*, 2014*a*,*b*, 2015]. We also use Mean Normalized Discounted Cumulative Gain (Mean NDCG) as the performance measure to capture the positional importance of retrieved items. We use time per iteration for measuring the time of execution. For all the datasets, we measure the accuracy of prediction on a held out set consisting of 20% data instances randomly selected. Training is done on the remaining 80% of the data instances. During the training phase, the held out set is considered as missing data.

We structure the evaluation along the lines of Ref. [Gopalan *et al.*, 2014*b*]. For all the datasets, utmost 10000 users were selected at random. For Movielens 100k and Movielens 1M, which had fewer than 10000 users, we used all the users. Let's denote this randomly selected user set by U. Let $Relevant_u$ and $Retrieved_u$ be the set of relevant and retrieved items for user u. Further, let M be the set of all movies, T_u be the set of movies present in the training set and rated by user u and P_u be the set of movies present in the held out set and rated by user u. Note that, $Relevant_u = P_u$. The set of unconsumed items for

[1] http://grouplens.org/datasets/movielens/
[2] http://www.netflixprize.com/

user u is given by $M - T_u$. For each user u in U, we predict the rating score for all her unconsumed items and select top P recommendations. These top P items are considered as the $Retrieved$ set for that user. We set P to 100 in our experiments. Using these definitions of $Relevant_u$ and $Retrieved_u$, we calculate the Precision$_u$-at-P and Recall$_u$-at-P for each user u in U as follows:

$$\text{Precision}_u\text{-at-}P = \frac{Relevant_u \cap Retrieved_u}{P}, \tag{5.21}$$

$$\text{Recall}_u\text{-at-}P = \frac{Relevant_u \cap Retrieved_u}{|Relevant_u|}. \tag{5.22}$$

We calculate Mean Precision and Mean Recall by averaging the precision and recall score from all the users as follows:

$$\text{Mean Precision} = \frac{\sum_{u \in U} \text{Precision}_u\text{-at-}P}{|U|}, \tag{5.23}$$

$$\text{Mean Recall} = \frac{\sum_{u \in U} \text{Recall}_u\text{-at-}P}{|U|}. \tag{5.24}$$

The F-measure is calculated using,

$$\text{F-measure} = \frac{2 \times \text{Mean Precision} \times \text{Mean Recall}}{\text{Mean Precision} + \text{Mean Recall}}. \tag{5.25}$$

We calculate Discounted Cumulative Gain$_u$-at-P as follows:

$$\text{DCG}_u\text{-at-}P = rel(1) + \sum_{k=2}^{P} \frac{rel(k)}{\log_2(k)}, \tag{5.26}$$

where, $rel(i)$ is 1 if the item is present in the test set, otherwise 0. We consider the test set as the ideal ordering for users and calculate Ideal Discounted Cumulative Gain-at-P (IDCG-at-P) based on that. Then we calculate Normalized Discounted Cumulative Gain-at-P (NDCG$_u$-at-P) for each user u in U as:

$$\text{NDCG}_u\text{-at-}P = \frac{\text{DCG}_u\text{-at-}P}{\text{IDCG}_u\text{-at-}P}. \tag{5.27}$$

Finally, we calculate Mean NDCG as follows:

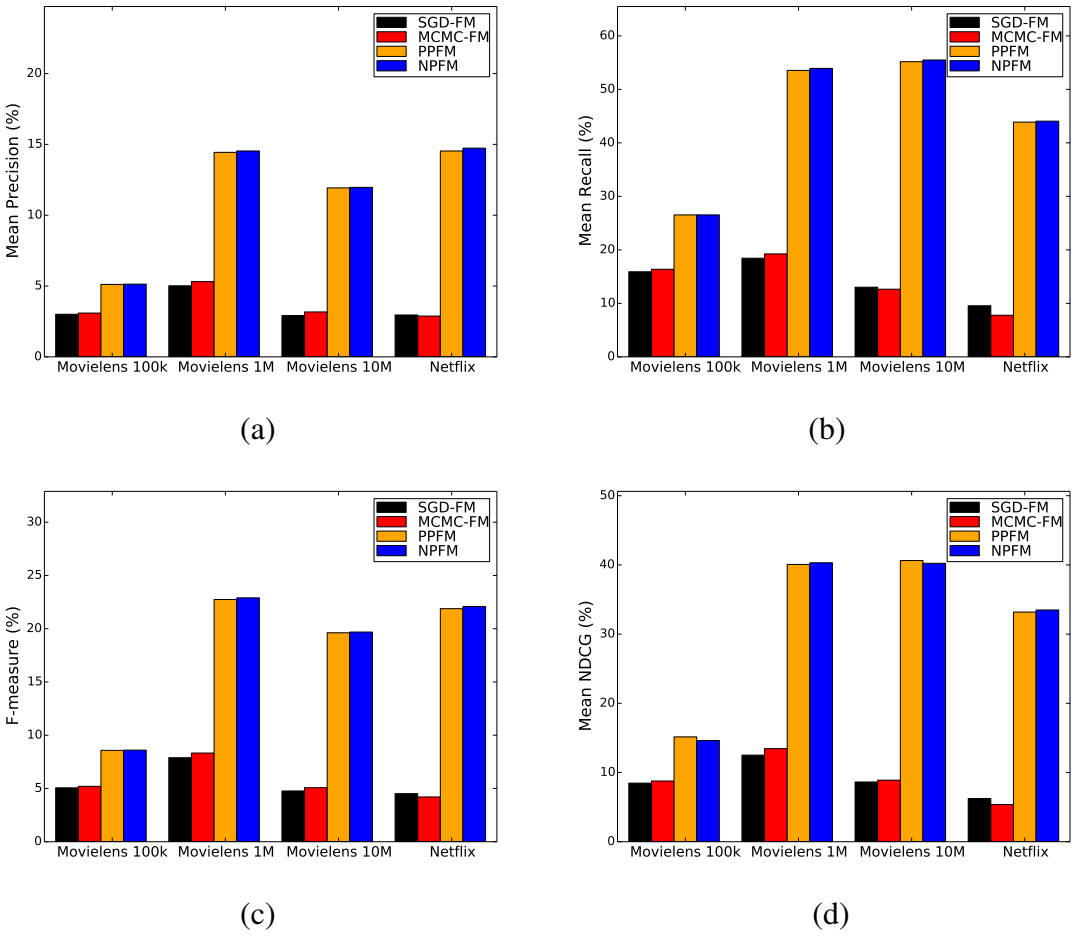


Figure 5.3: (a), (b), (c), and (d) show Mean Precision, Mean Recall, F-measure, and Mean NDCG comparison on different datasets for different algorithms respectively.

$$\text{Mean NDCG} = \sum_{u \in U} \frac{\text{NDCG}_u\text{-at-}P}{|U|}. \tag{5.28}$$

5.4.5 Baseline

We compare NPFM with two strong baseline methods: stochastic gradient descent for FM (SGD-FM) [Rendle, 2010] and Markov chain Monte Carlo FM (MCMC-FM) [C. Freudenthaler, 2011]. Note that in both the SGD-FM and MCMC-FM, Gaussian distributional assumption is made to represent the data likelihood.

5.4.6 Experimental Setup and Parameter Selection

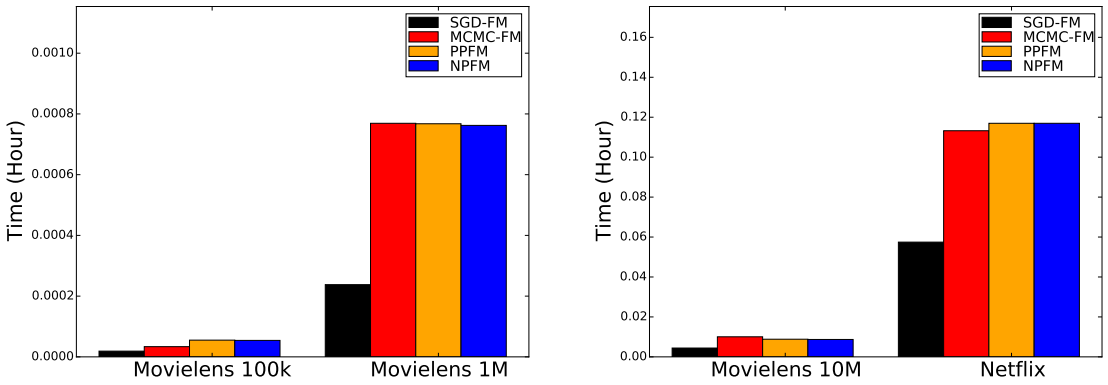
NPFM selects all the model parameters automatically, but we set the maximum number of factors as 50 for all the experiments. We chose the same latent factor dimension for all the experiments with PPFM, SGD-FM, and MCMC-FM, as well. NPFM, PPFM, and MCMC-FM are based on Gibbs sampling and they need an initial burn-in phase. We used 1500 burn-in iterations and 1000 collection iterations except in the Netflix dataset where 100 burn-in and 100 collection iterations were used due to time constraints. For SGD-FM we found that 300 iterations were sufficient for convergence. So, we ran SGD-FM for 300 iterations for all the datasets except the Netflix dataset where only 200 iterations were considered. Though due to the time constraints for Netflix dataset, we have considered less number of iterations, the number of iterations was sufficient for convergence.

In NPFM, all the hyperprior parameters ($a_0, b_0, c_0, d_0, \{a_i, b_i, e_i, f_i\}, h_0, \gamma_0$) are initialized to 1. We initialize w, v, r to a small positive value and all other parameters to 0. Likewise, for PPFM, all hyperprior parameters are initialized to 1, w and v are initialized to a small positive value and all other parameters are initialized to 0. We select the learning rate and regularization parameters of SGD-FM using cross validation. We found that SGD-FM performs best with 0.001 learning rate and 0.01 regularization. So we stick to this value for all the experiments of SGD-FM. For SGD-FM, we initialize w_0, w, and v using a Gaussian distribution with 0 mean and 0.01 variance. For MCMC-FM, we use standard parameter setting provided in the paper [C. Freudenthaler, 2011]: we set all the hyperprior parameters to 1, w_0, w, v are initialized using a Gaussian distribution with 0 mean and 0.01 variance and all other parameters are set to 0.

5.4.7 Results

Accuracy

Figure 5.3 shows the results on three Movielens data sets with 100k, 1 million, and 10 million ratings and the Netflix dataset with 100 million ratings. For all the datasets, NPFM and PPFM perform much better than the baseline method SGD-FM and MCMC-FM. We



(a) (b)

Figure 5.4: (a) and (b) show time per iteration comparison for different algorithms on Movielens 100k & Movielens 1M and Movielens 10M & Netflix datasets respectively.

observe that the Mean Recall scores (Figure 5.3 subplot (b)) and Mean NDCG values (Figure 5.3 subplot (d)) on all the datasets for NPFM and PPFM are much higher than the other methods. MCMC-FM and SGD-FM perform very poorly for all the datasets primarily due to the inappropriateness of the Gaussian assumption for count data.

Time

Figure 5.4 shows time per iteration for different methods. As the four datasets have different time scales, for visibility we group Movielens 100K and Movielens 1M together; and Movielens 10M and Netflix together. In all the datasets SGD-FM takes much less time than the other three methods. This is not surprising since SGD-FM is an online method and updates model parameters for each data instance, whereas the other three methods are batch algorithms. Though in Movielens 100K, MCMC-FM takes less time than the NPFM and PPFM, as dataset size increases, sparsity in the dataset also increases and as a result MCMC-FM, NPFM, and PPFM take almost equal time to run on the two larger datasets. Thus we are able to get the power of Poisson modelling without too much additional computational overhead. What is heartening to note is that NPFM and PPFM perform similarly both in terms of accuracy and time. This indicates that we are able to avoid setting the latent factor dimension apriori but still do not pay much of a cost in terms of performance.

This is particularly important when working with datasets where the appropriate dimension is hard to determine ahead of time.

5.5 Summary

This chapter describes Nonparametric Poisson Factorization Machine (NPFM), an alternative for formulating Factorization Machine for count data. The model exploits the natural sparsity of the data, has linear time and space complexity with respect to the number of non-zero observations and predicts the ideal number of latent factors for modeling the pairwise interaction in Factorization Machine. We also consider a special case of NPFM, the Parametric Poisson Factorization Machine (PPFM), that considers a fixed number of latent factors. Both NPFM and PPFM outperform some of the existing baselines by significant margin on several real-world datasets. Though we have found that NPFM works well when it mimic matrix factorization, but NPFM in the presence of "side-information" is needed to be validated. The side-information may include user specific features such as age, gender, demographics, and network information and item specific information such as product descriptions.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Factorization models are an important class of algorithms for Recommender Systems (RSs). In spite of having a vast literature on factorization models, several problems exist with different factorization model algorithms. In this thesis, we have taken a probabilistic approach to develop several factorization models. We adopt a fully Bayesian treatment of these models and develop scalable approximate inference algorithms for them.

Firstly, we develop the Scalable Bayesian Matrix Factorization (SBMF) which considers independent univariate Gaussian prior over latent factors as opposed to multivariate Guassian prior in Bayesian Probabilistic Matrix Factorization (BPMF) [Salakhutdinov and Mnih, 2008]. SBMF uses Markov chain Monte Carlo (MCMC) based Gibbs sampling inference mechanism to approximate the posterior, and has linear time and space complexity. SBMF has competitive performance along with the scalibility as validated by experiments on several real world datasets.

We then develop the Variational Bayesian Factorization Machine (VBFM) which is a batch scalable variational Bayesian inference algorithm for Factorization Machine (FM) [Rendle, 2010]. Additionally for large scale learning, we develop the Online Variational Bayesian Factorization Machine (OVBFM) which utilizes stochastic gradient descent to optimize the lower bound in variational approximation. The efficacy of both VBFM and OVBFM has been validated using experiments on several real world datasets.

Finally, we propose the Nonparametric Poisson Factorization Machine (NPFM), which models data using a Poisson distribution, and where the number of latent factors are theoretically unbounded and is estimated while computing the posterior distribution. We also consider a special case of NPFM, the Parametric Poisson Factorization Model (PPFM), that considers a fixed number of latent factors. Both PPFM and NPFM have linear time and space complexity with respect to the number of observations. Extensive experiments

on four different movie review datasets show that our methods outperform two strong baseline methods by large margins.

There are several potential future directions:

- It would be interesting to extend SBMF to applications like matrix factorization with "side-information", where the time complexity is cubic with respect to the number of features (which can be very large in practice). The side-information may include user specific features such as age, gender, demographics, and network information and item specific information such as product descriptions.
- OVBFM is an online algorithm. Hence, it would be interesting to extend OVBFM to applications where one can actively query labels for few data instances [Silva and Carin, 2012], leading to solve the cold-start problem.
- In NPFM, firstly it is worth while to further increase the scalability. To accomplish this, borrowing ideas from stochastic gradient Langevin dynamics [Welling and Teh, 2011], one can propose an online inference technique with parallel steps and reduce the convergence time further. Such adaptation may help apply NPFM to very large datasets like the KDD music dataset [Dror *et al.*, 2012]. On the other hand, NPFM can be applied to other types of factorization models as well apart from basic matrix factorization. We note that the latter part is an important future work to validate the efficacy of NPFM.

REFERENCES

Acharya, A., **J. Ghosh**, and **M. Zhou**, Nonparametric bayesian factor analysis for dynamic count matrices. *In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*. 2015.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, **10**(2), 251–276.

Arora, S., **R. Ge**, and **A. Moitra**, Learning topic models - going beyond SVD. *In 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*. 2012.

Beal, M. J., Variational algorithms for approximate Bayesian inference. *In PhD. Thesis, Gatsby Computational Neuroscience Unit, University College London.*. 2003.

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

Bishop, C. M., *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

Blackwell, D. and **J. MacQueen** (1973). Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, **1**, 353–355.

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

Burke, R. (2000). Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Science*, **69**(32), 4:2–4:2.

Burke, R. D. (2002). Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, **12**(4), 331–370.

C. Freudenthaler, S. R., L. Schmidt-Thieme, Bayesian factorization machines. *In Proc. of NIPS Workshop on Sparse Representation and Low-rank Approximation*. 2011.

Canny, J. F., Gap: a factor model for discrete data. *In SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*. 2004.

Cemgil, A. T. (2009). Bayesian inference for nonnegative matrix factorisation models. *Intell. Neuroscience*, **2009**, 4:1–4:17.

Chang, C. and **C. Lin** (2011). LIBSVM: A library for support vector machines. *ACM TIST*, **2**(3), 27.

Chi, E. C. and **T. G. Kolda** (2012). On tensors, sparsity, and nonnegative factorizations. *SIAM J. Matrix Analysis Applications*, **33**(4), 1272–1299.

Connor, M. and **J. Herlocker** (2001). Clustering items for collaborative filtering.

Dror, G., **N. Koenigstein**, **Y. Koren**, and **M. Weimer**, The yahoo! music dataset and kdd-cup '11. *In Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011*. 2012.

Gelfand, A. E. and **A. F. M. Smith** (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, **85**(410), 398–409.

Geman, S. and **D. Geman** (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6**(6), 721–741.

Gopalan, P., **L. Charlin**, and **D. M. Blei**, Content-based recommendations with poisson factorization. *In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014*a*.

Gopalan, P., **J. M. Hofman**, and **D. M. Blei**, Scalable recommendation with hierarchical poisson factorization. *In Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*. 2015.

Gopalan, P., **D. M. Mimno**, **S. Gerrish**, **M. J. Freedman**, and **D. M. Blei**, Scalable inference of overlapping communities. *In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*. 2012.

Gopalan, P., **F. J. Ruiz**, **R. Ranganath**, and **D. M. Blei**, Bayesian nonparametric poisson factorization for recommendation systems. *In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. 2014*b*.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**(1), 97–109.

Hernández-Lobato, J. M., **N. Houlsby**, and **Z. Ghahramani**, Stochastic inference for scalable probabilistic modeling of binary matrices. *In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 2014.

Hjort, N. L. (1990). Nonparametric Bayes estimators based on beta processes in models for life history data. *Ann. Statist.*.

Ho, J. C., **J. Ghosh**, and **J. Sun**, Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. *In The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 2014.

Hoffman, M. D., **D. M. Blei**, and **F. R. Bach**, Online learning for latent dirichlet allocation. *In Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, December 2010, Vancouver, British Columbia, Canada.*. 2010.

Hoffman, M. D., **D. M. Blei**, **C. Wang**, and **J. W. Paisley** (2013). Stochastic variational inference. *Journal of Machine Learning Research*, **14**(1), 1303–1347.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, **22**(1), 89–115.

Joachims, T. (2002). SVM light, http://svmlight.joachims.org.

Johnson, N. L., **A. W. Kemp**, and **S. Kotz**, *Univariate Discrete Distributions*. John Wiley & Sons, 2005.

Kim, Y. and **S. Choi**, Scalable variational bayesian matrix factorization with side information. *In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. 2014.

Koren, Y., Factorization meets the neighborhood: a multifaceted collaborative filtering model. *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. 2008.

Koren, Y., Collaborative filtering with temporal dynamics. *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*. 2009.

Koren, Y., **R. M. Bell**, and **C. Volinsky** (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, **42**(8), 30–37.

Lee, J., **M. Sun**, and **G. Lebanon** (2012). A comparative study of collaborative filtering algorithms. *CoRR*, **abs/1205.3193**.

Lim, Y. and **Y. Teh**, Variational Bayesian approach to Movie Rating Prediction. *In Proc. of KDDCup*. 2007.

Metropolis, N. and **S. Ulam** (1949). The monte carlo method. *Journal of the American statistical Association*, **44**(247), 335–341.

Miyahara, K. and **M. J. Pazzani**, Collaborative filtering with the simple bayesian classifier. *In PRICAI*. 2000.

Rendle, S., Factorization machines. *In ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 2010.

Rendle, S. (2012). Factorization machines with libfm. *ACM TIST*, **3**(3), 57.

Rendle, S., **Z. Gantner**, **C. Freudenthaler**, and **L. Schmidt-Thieme**, Fast context-aware recommendations with factorization machines. *In Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. 2011*a*.

Rendle, S., **Z. Gantner**, **C. Freudenthaler**, and **L. Schmidt-Thieme**, Fast context-aware recommendations with factorization machines. *In Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. 2011*b*.

Rendle, S. and **L. Schmidt-Thieme**, Pairwise interaction tensor factorization for personalized tag recommendation. *In Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. 2010.

Salakhutdinov, R. and **A. Mnih**, Probabilistic matrix factorization. *In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. 2007.

Salakhutdinov, R. and **A. Mnih**, Bayesian probabilistic matrix factorization using markov chain monte carlo. *In Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. 2008.

Salakhutdinov, R., **A. Mnih**, and **G. E. Hinton**, Restricted boltzmann machines for collaborative filtering. *In Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*. 2007.

Sato, M. (2001). Online model selection based on the variational bayes. *Neural Computation*, **13**(7), 1649–1681.

Silva, J. G. and **L. Carin**, Active learning for online bayesian matrix factorization. *In The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. 2012.

Srebro, N. and **T. S. Jaakkola**, Weighted low-rank approximations. *In Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. 2003.

Su, X. and **T. M. Khoshgoftaar** (2009). A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, **2009**, 421425:1–421425:19.

Titsias, M. K., The infinite gamma-poisson feature model. *In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. 2007.

Tzikas, D., **A. Likas**, and **N. Galatsanos** (2008). The variational approximation for bayesian inference. *IEEE Signal Processing Magazine*, **25**(6), 131–146.

Walker, S. G. (2007). Sampling the Dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, **36**(1), 45–54.

Welling, M. and **Y. W. Teh**, Bayesian learning via stochastic gradient langevin dynamics. *In Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 2011.

Xiong, L., **X. Chen**, **T. Huang**, **J. G. Schneider**, and **J. G. Carbonell**, Temporal collaborative filtering with bayesian probabilistic tensor factorization. *In Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*. 2010.

Xue, G., **C. Lin**, **Q. Yang**, **W. Xi**, **H. Zeng**, **Y. Yu**, and **Z. Chen**, Scalable collaborative filtering using cluster-based smoothing. *In SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. 2005.

Zhou, M., Infinite edge partition models for overlapping community detection and link prediction. *In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*. 2015.

Zhou, M. and **L. Carin**, Augment-and-conquer negative binomial processes. *In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*. 2012.

Zhou, M. and **L. Carin** (2015). Negative binomial process count and mixture modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, **37**(2), 307–320.

Zhou, M., **L. Hannah**, **D. B. Dunson**, and **L. Carin**, Beta-negative binomial process and poisson factor analysis. *In Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*. 2012.

Zhou, Y., **D. M. Wilkinson**, **R. Schreiber**, and **R. Pan**, Large-scale parallel collaborative filtering for the netflix prize. *In Algorithmic Aspects in Information and Management, 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings*. 2008.

LIST OF PAPERS BASED ON THESIS

1. **Saha, A., A. Acharya, B. Ravindran, and J. Ghosh**, Nonparametric Poisson Factorization Machine, *2015 IEEE International Conference on Data Mining*, (ICDM 2015), 967-972.

2. **Saha, A., R. Misra, and B. Ravindran**, Scalable Bayesian Matrix Factorization, *In Proceedings of the 6th International Workshop on Mining Ubiquitous and Social Environments*, (MUSE 2015), 43-54.

3. **Saha, A., J. Rajendran, S. Shekhar, and B. Ravindran**, How Popular Are Your Tweets?, *In Proceedings of the 2014 Recommender Systems Challenge*, RecSysChallenge'14, 2014, 66:66-66:69.

Curriculum Vitae

Name	Avijit Saha
Date of Birth	10^{th} December 1990
Address	Vill - Daharthuba, PO - Hatthuba, PS - Habra, DIST - North 24 Parganas, STATE - West Bengal, PIN - 743269
Education	B-Tech (CSE), West Bengal University of Technology, MS by Research (CSE), IIT Madras

GTC Members

Chairman	Dr. Shukhendu Das Department of Computer Science and Engineering, Indian Institute of Technology, Madras.
Guide	Dr. Balaraman Ravindran Department of Computer Science and Engineering, Indian Institute of Technology, Madras.
Members	Dr. Shankar Balachandran Department of Computer Science and Engineering, Indian Institute of Technology, Madras.
	Dr. Krishna Jagannathan Department of Electrical Engineering, Indian Institute of Technology, Madras.