


November 2020

Multimodal Data Fusion and Attack Detection in Recommender Systems

Mehmet Aktukmak
University of South Florida

Follow this and additional works at: <https://digitalcommons.usf.edu/etd>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Library and Information Science Commons](#), and the [Statistics and Probability Commons](#)

Scholar Commons Citation

Aktukmak, Mehmet, "Multimodal Data Fusion and Attack Detection in Recommender Systems" (2020).
USF Tampa Graduate Theses and Dissertations.
<https://digitalcommons.usf.edu/etd/9519>

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact scholarcommons@usf.edu.

Multimodal Data Fusion and Attack Detection in Recommender Systems

by

Mehmet Aktukmak

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Co-Major Professor: Ismail Uysal, Ph.D.
Co-Major Professor: Yasin Yilmaz, Ph.D.
Kwang-Cheng Chen, Ph.D.
Alfredo Weitzenfeld, Ph.D.
Samuel Mercier, Ph.D.

Date of Approval:
November 5, 2020

Keywords: probabilistic graphical models, machine learning, multimodal data fusion, sequential attack detection, variational inference, recommender systems

Copyright © 2020, Mehmet Aktukmak

Dedication

To my family, for supporting me through all of life hardships.

To my friends and teachers, for inspiring and encouraging my pursuit of knowledge.

Acknowledgments

Respect and thanks, to my sister Yagmur, my mother Perihan, and my father Ali Aktukmak, for keeping up with this legacy by devoting their lives to surrounding me with top-notch environments. Your unconditional love is unexampled, and your sacrifices cannot be repaid. Thanks to all my family for dropping everything they had and being there for me with their whole network whenever I needed support or guidance, and always loving me plenty. My family is my highest luck and wealth.

A greetings to all my friends who pulled me through life's hardships and promoted the better in me at all times. The influence of you all on who I am today cannot be underestimated. No space could accommodate the names of all the wonderful companions. I thank my friends Berker Perkoz, Selahattin Burak Sarsilmaz, Kadriye Merve Dogan, Ali Fatih Demir, Cagri Cetin, Pallavi Singh, Pavia Bera, Santiago Hernandez, and Ahmet Manisali for all the emotional support and friendship over my graduate school years. I will remain indebted to my colleagues Ismail Ulutürk, Rania Elashmawy, Ria Kanjilal for their friendships, kindness, and their technical and non-technical insights. I will remain grateful to Dr. Samuel Mercier for helping me to get started in this field.

Special thanks to Dr. Kwang-Cheng Chen, Dr. Alfredo Weitzenfled for their commitment to my growth. Special thanks to Dr. Yasin Yilmaz for academic support and valuable technical guidance. Special thanks to Dr. Ismail Uysal for believing in me, providing me the opportunity and the means to achieve this level, and having the patience to allow me to develop my skills.

Table of Contents

List of Tables	iv
List of Figures	v
Abstract	vi
Chapter 1: Introduction	1
1.1 Recommender Systems	1
1.2 Matrix Completion	3
1.3 Beyond Rating Matrix	4
1.4 Security and Robustness	5
1.5 Contributions	7
1.5.1 An Alternating Neural Network Model	7
1.5.2 Probabilistic Fusion of Multiple Diverse Information	7
1.5.3 Sequential Attack Detection	8
1.6 Outline	9
Chapter 2: Matrix Completion	10
2.1 Introduction	10
2.2 Base Models	12
2.2.1 Latent Variable Models	12
2.2.2 Regression Models	14
2.2.3 Bayesian Methods	15
2.2.4 Deep Learning Models	17
2.3 Alternating Neural Network Model	18
2.4 Comparative Study	22
2.4.1 Experimental Setup	22
2.4.2 Implementation Details	23
2.4.3 Cross-validation and Evaluation Procedure	25
2.4.4 Results and Discussion	26
2.5 Conclusions and Future Work	27
Chapter 3: Heterogeneous Data Fusion	30
3.1 Introduction	30
3.2 Related Work	32
3.2.1 Baseline MF Models	33

3.2.2	Prior Based Models	34
3.2.3	Regression-Based Models	34
3.2.4	Disjoint Models	35
3.3	Proposed Model	37
3.3.1	Model Definition	37
3.3.2	Inference	39
3.3.3	Computational Complexity	44
3.4	Experimental Results	45
3.4.1	Evaluation Models	45
3.4.2	Datasets	49
3.4.3	Evaluation Metrics	50
3.4.4	Splitting the Dataset for Training and Testing	51
3.4.5	Simulation Study	52
3.4.6	Movie Recommendation Study	53
3.4.7	A Discussion on Deep Learning Methods	62
3.5	Conclusion	64
3.6	Limitations and Future Research Directions	65
Chapter 4: Sequential Attack Detection		67
4.1	Introduction	67
4.2	Related Work	71
4.2.1	Attack Types	71
4.2.2	Static Detection Algorithms	72
4.2.3	Temporal Detection Algorithms	73
4.3	Proposed Framework	73
4.3.1	Latent Variable Model	74
4.3.2	Model Optimization	78
4.3.3	Attack Detection	81
4.4	Experimental Study	88
4.4.1	Test Environment	89
4.4.2	Robustness Assessment	96
4.4.3	Profile Detection Performance	100
4.4.4	Sequential Detection Performance	103
4.5	Conclusion	104
Chapter 5: Concluding Remarks		106
5.1	Summary	106
5.2	Insights and Possible Future Work	108
References		110
Appendix A: Copyright Permissions		127
Appendix B: Closed Form Expression of Rating Likelihood		131
Appendix C: Variational Inference of Latent Variable Model		133

About the Author End Page

List of Tables

Table 2.1: MSE comparison on small datasets	26
Table 2.2: MSE comparison on medium datasets	27
Table 3.1: MovieLens dataset statistics	50
Table 3.2: Sample users from MovieLens 100K	56
Table 3.3: Sample movie groups from MovieLens 100K	57
Table 3.4: MSE comparison with standard benchmark algorithms on the warm setting . .	60
Table 3.5: Performance comparison with recent state-of-the-art algorithms	62
Table 3.6: The performance analysis of various deep learning algorithms	64
Table 4.1: Dataset statistics	88
Table 4.2: Test setup	99

List of Figures

Figure 2.1: Graphical model of FSNMC for two layer MLP configuration	20
Figure 2.2: Average Mean Square Error	29
Figure 3.1: Graphical model representation of the proposed MF-MSI method	38
Figure 3.2: MSE vs missing value fraction on synthetic dataset	52
Figure 3.3: Convergence analysis on synthetic dataset	54
Figure 3.4: Recall performances on MovieLens datasets when $L = 2$	58
Figure 3.5: Recall performances for the varying number of recommended movies	59
Figure 4.1: Graphical model representation of the linear latent variable model.	74
Figure 4.2: Different characteristic of the two variants of proposed algorithm with respect to varying obfuscation of the attack and the sparsity of the dataset.	92
Figure 4.3: Average AUC scores when mixed push/nuke/obfuscated attacks are performed with different attack configurations, i.e, filler and attack size, to the MovieLens dataset.	95
Figure 4.4: Average AUC scores when mixed push/nuke/obfuscated attacks are performed with different attack configurations, i.e, filler and attack size, to the three different datasets.	97
Figure 4.5: Sample decision statistics when push attack is injected for item #294 of Movie-lens dataset beginning from the user index 85.	100
Figure 4.6: Sequential detection performance when mixed push/nuke/obfuscated attacks are performed to the three different datasets.	101

Abstract

The commercial platforms that use recommender systems can collect relevant information to produce useful recommendations to the platform users. However, these sources usually contain missing values, imbalanced and heterogeneous data, and noisy observations. Such characteristics render the process of exploiting the information nontrivial, as one should carefully address them during the data fusion process. In addition to the degenerative characteristics, some entries can be fake, i.e., they can be the outcomes of malicious intents to manipulate the system. These entries should be eliminated before incorporation to any recommendation task. Detecting such malicious attacks quickly and accurately and then mitigating them is vital to enhance the trustworthiness and robustness of the system, which is another non-trivial process. Recent advances in probabilistic data fusion pave the way for addressing such issues. Such problems can be handled in a principled way by developing proper latent variable models that consider the different nature of the observed data types, and training the latent variable models with computationally efficient learning algorithms. This dissertation aims to develop such latent variable models to effectively fuse multiple heterogeneous information sources to improve the accuracy, robustness, and safety of the recommender systems. First, we propose a latent variable model that can fuse the categorical and numerical information sources containing missing values such as the rating matrix, user attributes, and item features. We demonstrate its superior performance over existing collaborative filtering algorithms that only use the rating matrix, and the more recent algorithms that incorporate the side information, i.e., the user attributes and item features, via different techniques. Subsequently, by exploiting the proposed latent variable model, we design a sequential attack detection framework. By extracting uni-variate statistics from the latent space of our model, and using in a sequential change detection algorithm, we obtain a framework that can use multiple diverse information sources to improve the

attack detection performance. The experimental results demonstrate the enhancements over other baseline algorithms that only use the rating patterns of the profiles, in terms of both detection rate and sequential detection performance.

Chapter 1: Introduction

1.1 Recommender Systems

The Internet makes it possible to access an unprecedented amount of consumer content. Naturally, as the number and diversity of the data sources increase, the information retrieval process becomes a challenge of its own. The management of big data has been an active research topic in the machine learning community. For instance, recommender systems personalize content delivery for popular applications such as streaming devices, e-commerce, and online media. Within this context, a successful recommender system should accurately and efficiently guide the consumers to the products and information they are looking for.

The users of these platforms provide feedback as they interact with the content. The feedback can be explicit or implicit based on the type of interaction. Explicit feedbacks are the ones the users directly provide they give information about their preferences. Implicit feedbacks are not direct. Instead, the recommendation system learns implicit feedback from the user's browsing/clicking patterns or transaction history. As an example of explicit feedback, in a movie platform, after a user watches a movie, she can enter a rating from an integer interval ranging from 1 to 5. This information explicitly reflects this user's degree of preference, making this rating a useful source of information for the recommendation engine. Another example can be the reactions of the users on social media posts such as Facebook videos. The like/dislike information is useful explicit feedback for recommending new video posts favorable to that user's taste. As an example of implicit feedback, a user can browse Amazon to look for a specific product. As she takes a view of similar products, the system can mine the information and infer her preference and recommend similar items to make it easier for the user to reach what she is trying to find.

The interaction history (both implicit and explicit) can be packed into a rating matrix. The rows and columns correspond to the users and the items respectively. Naturally, for almost all the platforms, the rating matrix is very sparse and imbalanced. Some users might be very active and provide entries for their corresponding rows while some other users might be inactive or newly registered so that their information content is limited, i.e., the corresponding rows are very sparse. A successful recommender system is expected to produce reasonable recommendations for all types of users. Among the recommender system algorithms, the collaborative filtering algorithms [1] have been popular in the last two decades. The main intuition behind these algorithms is to transfer knowledge from one user/item to other users/items by using the dependencies between them. That results in a collaborative approach to alleviate the effects of sparse rows. For example, the dependencies can be inferred and similar users can be found by computing a similarity metric between the rating vectors of user pairs. Then, for a user with a sparse rating vector, the recommendations can be generated not only by using the user's interaction history but also by the interaction histories of the users with similar preference patterns.

Besides collaborative filtering algorithms, the other two commonly used methods are the knowledge-based and content-based methods [2, 3]. For example, a platform for purchasing expensive items such as a car or a home, in which it is very hard to collect enough ratings for a data-driven model, knowledge-based algorithms are very suitable. The user can provide specific attributes of the item to explore the inventory and search for the item that meets her requirements. Content-based methods are useful when the item/product features are available for the recommender system, especially, for the cases in which the item is new and does not have enough ratings for collaborative filtering approaches. In this study, we will focus on a unified approach that combines collaborative filtering and content-based methods. In this context, our main focus will be to fuse diverse information sources such as user attributes, item features, as well as rating matrix.

1.2 Matrix Completion

The main goal of a recommender system is to provide predictions for unknown interactions. That means predicting missing entries of the rating matrix which defines the problem as matrix completion. However, in this case, the matrix is very large in both dimensions, i.e, in a typical commercial platform, the number of users and the items can go beyond millions. In addition to size, another challenge is the extreme sparsity of the matrix, which is typically larger than 99%. These challenges make it hard to use most of the matrix completion algorithms in the literature without major modifications to the existing ones.

Matrix completion can be regarded as generalization of regression/classification models. Although the target feature is fixed in the latter models, it changes from user to user in the matrix completion task. For example, in a multiple-choice integer rating system, a straightforward approach could be to fill the missing values with a basic imputation algorithm, then use a regression model by fixing the target variable and separating the rest as dependent variables/regressors. By using the regression model, one can update the missing entries which reside in the target variable column by assuming fully observed regressors, then move on by fixing the next variable as the target. With this iterative structure, all the missing entries can be updated. This alternating imputation procedure can continue until the changes in the values of missing entries converge. In Chapter 2, we will introduce an example of this approach by using a neural network as the regression model, and then discuss the advantages and disadvantages of such methods.

A more popular family of methods includes completing the rating matrix by using latent factor models. These methods assume underlying latent variables for each user and item. Since the information content of the rating matrix is limited compared to its size, low dimensional latent variables can represent the characteristics of the users and items efficiently. For example, in a platform with tens of thousands of items, one can use a latent variable model with size on the order of tens to represent a user's preference. These low dimensional vectors are typically chosen as constrained real-valued dense vectors. The constraints are usually set in order to avoid over-fitting or just to increase the interpretability of the vectors. In this study, our focus will be on such latent

variable models since they allow a probabilistic design with multiple diverse information sources. They are also computationally efficient in training and testing, which makes them more suitable for online applications. The challenge of these models is to incorporate additional sources of information beyond the rating matrix in an efficient manner.

1.3 Beyond Rating Matrix

In addition to being very sparse, the rating matrix is also imbalanced, i.e, the sparsity changes from row to row (user to user) and column to column (item to item). The main reason is the differences between user activities and item popularities. The users who have rated a very small number of items, labeled as cold users, can be regarded as inactive (or can also be new), and they will have a more sparse rating vector than the population average. The users who have rated a decent number of items are called warm users. The same analogy is true for the items which differ in terms of popularity. They can be categorized as warm and cold items. A recommender system is expected to produce reasonable recommendations for both cold and warm users/items. This can be regarded as another challenge for the matrix completion task in addition to the challenges coming from the sparsity and large size of the rating matrix.

Fortunately, different sources of information exist for most of the platforms that can be used to alleviate the aforementioned challenges. These sources can include i) the demographic information of the users, such as age, occupation, gender, zip code, ii) the item features such as movie genre, release date, plot, book author, publisher, language, video tags, iii) the social relationships/trust between the users such as friendship network, reputation rating, iv) the network structures connecting the users/items based on specific criteria, v) textual reviews of the users, and vi) cross-domain knowledge from other related platforms, i.e, using the rating matrices from distinct domains. If one can overcome the difficulties of efficiently incorporating these sources of information, the recommendation quality can be greatly improved compared to using only the rating matrix.

However, the incorporation of these sources is not straightforward. The challenges are i) the sources can conflict with each other which may result in performance degradation in recommendation quality or erroneous recommendations after incorporation, ii) the sources can include, noisy, erroneous, fake entries, iii) the sources can have missing entries, not only in the rating matrix, but in the other sources as well, iv) the sources can include diverse data types, i.e, categorical, numerical or textual data. The algorithm to fuse the sources should address all the aforementioned concerns. The precision should be modeled to down-weight or eliminate noisy, fake, erroneous sources. Missing entries should be handled in a principled way without discarding. The data types should be modeled according to the nature of the data. Probabilistic models are very suitable, especially the latent variable models which can address these concerns with a proper design and a computationally efficient training algorithm.

1.4 Security and Robustness

Another aspect studied in this thesis is the security of the recommender systems. As these systems typically serve global commercial platforms with high cash flow, they are very attractive to people with malicious intent. They can try to manipulate the system to gain some benefits. One type of motivation can be to promote an item in an e-commerce platform to increase its sales. The promotion can be done by injecting fake entries with high ratings for a target item. Hence, the system recommends that item more and makes it more popular. Another motivation can be to demote (nuke) an item to decrease its sales. This item is usually a rival item so that demoting it can increase the sales of the competing item. Disturbing the recommendation system can be motivation alone. By flooding the system with fake entries, the operation of the system can be distracted, i.e. the efficiency and accuracy can be decreased. For example, such an attack can originate from a rival platform or hackers with diverse intentions. It is known that most of the recommender systems are prone to these types of attacks [2, 3], i.e., one can obtain a reliable prediction shift on the rating of a target item by entering fake ratings and can alter the average ranking of the item.

Attack detection algorithms have been developed to detect and mitigate these types of attacks in the last two decades. By analyzing the rating matrix, one can form distinctive features to separate genuine and fake profiles. In reality, although we can obtain a large set of genuine users from the pool of already registered users in the platform, it is very hard to get information about the rating behaviors of the fake profiles that can register in the future. This problem corresponds to the case in which no a priori information exists about the anomalies which promotes an unsupervised approach for the attack detection framework as a more general solution, as the focus of this thesis. As discussed in Section 1.3, we have different sources of information that can alleviate problems due to the imbalanced rating matrix to increase recommendation accuracy. These sources can also help to increase the performance of attack detection. For example, a recommendation algorithm can relate the user demographic information to the rating behaviors by learning from the ratings and attributes of the genuine users in a training set. Then, if the newly registered users significantly deviate from this hidden pattern, it can be a sign of anomalous behavior. Hence, this study will also focus on how to use the side information of the users to improve the attack detection performance.

In a traditional rating matrix, one user can enter only one rating for a specific item. Therefore, the attacker should inject many fake profiles to affect the system and to create a prediction shift. In a realistic setting, we would expect these fake profiles to rate the item sequentially in a short time interval, to get quick and effective results. This setting requires sequential attack detection in recommender systems, where the anomalies coming from the ratings and other information sources, e.g, the user attributes, can be aggregated in a temporal way to increase the detection performance. The advantages of the sequential approach can be summarized as follows: i) There will be newly registered users having diverse preferences/tastes from the users in the training set, which may cause frequent false alarms. By taking the temporal rate of anomalies, the false alarms can be reduced since these users will not be flagged as fake profiles as long as they are rare events. ii) The attackers can generate stealthy attacks by hiding their attack in small anomalies in their sequential fake profiles, and distributing the attack impact over a longer time. While the small anomalies in individual fake profiles can easily bypass non-sequential detection algorithms, they

can be accurately detected by sequential algorithms that aggregate the small persistent anomaly evidence over time to detect these intelligent attacks. This can further increase the true positive rates of the detector.

1.5 Contributions

In this section, we summarize the contributions of this thesis. The details of the studies can be found in the corresponding chapters.

1.5.1 An Alternating Neural Network Model

We present a novel framework for matrix completion in Chapter 2, where the missing values are accumulatively estimated with feature-based neural networks. Specifically, for each individual feature, a different model is trained to predict the missing values in the observations using the remaining features as input to provide an initial estimate. These estimates are then used to initiate the next round of model training to converge iteratively to the final prediction of the missing values. The weight parameters of the networks are propagated through these accumulative iterations which leads to a computationally efficient algorithm where training times become progressively shorter with each round. Results show that the proposed algorithm outperforms the prior work in 80% of the test scenarios when compared to four universally accepted methods on a combination of different datasets and missing data ratios.

1.5.2 Probabilistic Fusion of Multiple Diverse Information

As mentioned before, recommender systems that exclusively rely on past interactions between the users and the items underperform in settings with very few observations. Their performance at such extreme sparsity can be improved by exploiting the side information of the users and the items including the demographics and the item descriptions. However, such side information is mostly heterogeneous and multi-modal, including both numerical and categorical features to yield a nontrivial incorporation process. Researchers have addressed this problem mainly by converting

the categorical features into numerical ones or forming numerical similarity matrices. In Chapter 3, we present a different approach in the form of a novel Bayesian probabilistic generative framework which can effectively incorporate multi-modal side information into the matrix factorization (MF-MSI) model. With the help of local quadratic bounds on the categorical likelihoods, we derive a scalable and computationally efficient iterative optimization method based on the Variational EM technique to learn the posterior distributions of latent variables associated with each user and item. A comprehensive experimental study on both simulated and real benchmark datasets demonstrate the proof-of-concept where the additional side information improves both the prediction accuracy and the ranking performance over more than a dozen popular baseline models. The proposed MF-MSI model claims state-of-the-art performance in the majority of the test scenarios when compared to more recently introduced recommender systems which can also utilize the side information via different techniques.

1.5.3 Sequential Attack Detection

The last contribution considers the security aspect with quick and accurate detection of attacks by observing the newly created profiles sequentially to prevent the damage which may be incurred by the injection of new profiles with dishonest ratings. We propose a framework in Chapter 4, which consists of a latent variable model and, is trained by a variational EM algorithm followed by a sequential detection algorithm. The latent variable model generates homogeneous representations of the users given their rating history and the mixed datatype attributes such as age and gender. The representations are then exploited to generate uni-variate statistics to be efficiently used in a sequential detection algorithm that can accurately detect persistent attacks while maintaining low false alarm rates. We apply our framework to three different real-world datasets. We first demonstrate a robust performance over different attacking strategies and configurations, and then compare the proposed framework with several existing baseline algorithms for the tasks of attack profile detection and sequential detection. Our results indicate a superior algorithm especially as the complexity of attack types increases.

1.6 Outline

In Chapter 2, we discuss the base algorithms that are used in collaborative filtering and matrix completion. Then, we discuss the connections with the deep learning methods, and finally present our first contribution, an alternating neural network model, followed by the experimental studies that demonstrate its performance. In Chapter 3, we present the details of our probabilistic fusion algorithm, Matrix Factorization with Multi-modal Side Information (MF-MSI). First, we give the mathematical definition of the model. Then, we derive the corresponding inference algorithm and present its computational complexity for training, followed by the experimental results. In Chapter 4, we introduce the attack detection problem in recommender systems. First, we give the details of the latent variable model and the sequential detector used as the main components of the framework in detecting fake attack profiles. Then we present the experimental study that shows the effectiveness of the proposed framework. In Chapter 5, we provide the concluding remarks by highlighting the novel contributions of this dissertation and possible future work.

Chapter 2: Matrix Completion

2.1 Introduction

Missing value presence is a common problem which degrades the quality of any dataset and impacts data analysis. Missingness mechanism may depend on either the data collection or generation process. Since the missing values in a dataset make data processing and analysis more difficult, matrix completion becomes an important preprocessing step. Inference methods based on machine learning have shown significant promise for matrix completion which includes wide-ranging applications from recommender systems [4] to operations research [5], from image processing [6] to product development [7] and high failure rate experiments [8].¹

Among matrix completion algorithms, unsupervised linear latent models have been widely used to estimate the missing values. Finding the latent factors and associated coefficients imitate the data generation process so that the missing entries are inferred easily from the known entries of the observations via the trained model. Principle Component Analysis (PCA) [10] and matrix factorization [11] are the most popular linear latent models for missing variable estimation even today. In the case of incomplete datasets, the methods estimate model parameters iteratively with gradient descent or alternating least squares until a convergence criterion is met. Observed and unobserved variables are regarded as random where introducing priors for latent variables makes the model less prone to overfitting especially for sparse datasets. Variational inference [10] or stochastic gradient descent [12] can further be used to infer the model parameters.

Another popular approach is to define a nonlinear mapping between the latent and observed variables with an unsupervised model. In this context, Kernel PCA [13, 14], auto-encoder [15, 16] and deep auto-encoder [17] based methods have been proposed to increase the model capacity in

¹Part of this chapter was published in [9]. Permission is included in Appendix A.

order to represent more complex relations. Application areas that show promising results include DNA micro-arrays [13], traffic flow [14], metabolite data [15], recommender systems [16] and even oil production [18]. Nonlinear methods are prone to over-fitting and parameter shrinking with L2 norm constraint is the most common regularization technique for that purpose. Probabilistic version of auto-encoders called the variational auto-encoders are probabilistic interpretations of the standard autoencoders which use prior for latent variables and estimates the posterior distribution of latent variables by assuming Gaussian latent space [19].

Introducing pseudo supervision to linear latent models has first been introduced in [20]. The model estimates the latent factors by building a PCA model given completed dataset from the previous iteration. Model parameters are trimmed according to known and unknown variables for each observation to form a linear regression model from observed to missing variables. Model over-fitting is avoided by using early stopping in [7] which has also shown promising results for several small scale datasets.

Nonlinear supervised methods are natural extensions of linear supervised methods. However, building separate complex regression models for each feature significantly increases the number of parameters to be estimated which can cause over-fitting, especially in the case of very sparse datasets [21]. On the other hand, in particular cases, mostly associated with scientific experimental datasets with relatively smaller fractions of missing values where the number of observations is several orders of magnitude larger than the number of features, they can provide higher accuracy. Building independent models associated with each feature will also allow parallel optimization of the models which can ultimately decrease computational burden and/or processing times. In this chapter, we propose a nonlinear supervised matrix completion algorithm based on multi-layer perception (MLP) specifically tailored for scientific experimental datasets. We assign an independent regression model for each feature and infer the parameters by using a novel iterative alternating method. At each iteration, model parameters are inferred by using the completed matrix from the previous iteration. Afterwards, missing variables are estimated given these newly inferred model parameters where the cycle continues until a convergence criterion is met. We also introduce

additional steps to ensure a robust learning process with fast convergence with good generalization. Our results show better accuracy for small to medium fractions of missing values for scientific experimental datasets with varying sizes.

This chapter is organized as follows. The details of the previous models proposed for matrix completion task are provided as background in Section 2.2. Then, we explain the proposed approach and discuss the differences with the baseline algorithms in Section 2.3. In Section 2.4.1, the experimental setup including the dataset descriptions, the definitions of the competitive algorithms are presented in detail. The procedure to optimize and tune the models and the evaluation metrics are introduced in Section 2.4.3. The chapter is concluded with the experimental results and discussions.

2.2 Base Models

2.2.1 Latent Variable Models

Latent variable models have been widely used for matrix completion problems in recent years. Matrix factorization is a simple latent variable method used in collaborating filtering applications to complete sparse matrices [12]. In this method, the features are represented with latent factors and the instances are represented with the factor coefficients. The factors are assumed to be lying in a lower dimensional latent factor space so that the observed matrix is assumed to be generated with the multiplication of those latent factors with their associated coefficients as follows:

$$x_i = Wz_i + m + \epsilon_i, \quad (2.1)$$

where W corresponds to the factor loading matrix in which the latent factors of each feature are stacked together in order. $z_i \in \mathbb{R}^K$ is latent variable of instance i (the factor coefficient) so that the multiplication with factor loading matrix gives the prediction of the variables of i th instance with the addition of the column wise mean vector m . ϵ is the error induced by the model during projection of data on a lower dimensional space, which is modeled as zero mean Gaussian random

variable. If we denote the the dimension of latent factor space as K and the dimension of the matrix to be completed as $N \times D$, the corresponding dimensions of the variables for a single instance i , will be in the forms $x_i \in \mathbb{R}^D$, $z_i \in \mathbb{R}^K$, $W \in \mathbb{R}^{D \times K}$, and $m, \epsilon \in \mathbb{R}^D$ respectively.

The main goal is to infer z_i , W and m given observed variables of incomplete matrix X . The objective of this optimization problem can be defined as

$$\min_{W,Z} \sum_{(i,j) \in \Omega} (x_{ij} - w_j z_i - m_j)^2 + \lambda(\|w_j\|^2 + \|z_i\|^2), \quad (2.2)$$

in which the right hand-side of the equation with coefficient λ corresponds to the regularization term to prevent the model from over-fitting that should be optimized via cross-validation as a hyper-parameter. Although Eq. 2.2 uses mean square error metric and L2 norm regularization for the reconstructions of known values in the matrix, several different metrics and regularization terms have been proposed for different tasks to improve the model [12, 22]. Since Singular Value Decomposition(SVD) is not straightforward due to the presence of missing values in the matrix, the researchers mostly prefer to use stochastic gradient or alternating least square methods to infer the z_i , m and W especially for big and sparse datasets [12].

Building latent variable models in the presence of missing values have also been studied over PCA models [10]. In those approaches, the model building and exploiting steps are separated to enable the application of SVD to the incomplete matrix. Model building step is performed to infer the latent variables and factor loading matrix with SVD by using the old predictions of missing values. Associated PCA model equation is given as

$$x_{i,t} = W_t z_{i,t} + m_t + e_{i,t}, \quad (2.3)$$

where t refers to the iteration number. Model building step includes inferring the parameters and the latent variables by using the old predictions of the missing values. On the other hand, model exploiting step performs predictions for the unobserved variables, whose indices are kept in set

$\bar{\Omega}$, by using the inferred parameter values z_i , W and m at iteration t , and then replaces the old predictions with the new estimations as follows:

$$x_{ij,(t+1)} = w_{j,t}z_{i,t} + m_{j,t}, (i, j) \in \bar{\Omega} \quad (2.4)$$

To initialize the algorithm, the missing variables are filled with feature-wise means of the observed variables to enable initial SVD. Consecutive model building and exploiting steps are iterated until the convergence of either the predictions or the model parameters.

2.2.2 Regression Models

Another common approach to matrix completion is to combine regression framework with the latent variable models to increase the accuracy of the predictions [20]. The missing variables of each observation is inferred within a regression model by using the predicted factor loading matrix and the known variable of that observation. After the column-wise mean vector is subtracted, i.e., ($y_{i,t} = x_{i,t} - m_t$), where m_t is calculated by using the known values of the matrix, then PCA decomposition is applied to the model;

$$y_{i,t} = z_{i,t}W_t^T + e_{i,t} \quad (2.5)$$

For regression framework, as preprocessing step, W is reordered and splitted for each observation with missing values such as $W_t = [W_t^* \ W_t^\#]$ in which W_t^* corresponds to the factor loadings of the known variables of the observation and $W_t^\#$ corresponds to factor loadings of the unknown variables. The regression from known variables to unknown variables is performed as

$$y_{i,t}^\# = W_t^\#(W_t^{*T}W_t^*)^{-1}W_t^{*T}y_{i,t}^* \quad (2.6)$$

The regression framework is improved in TSR algorithm [7] to use trimmed scores such as

$$\mathbf{y}_{i,t}^{\#} = \mathbf{S}_t^{\#*} \mathbf{W}_t^* \left(\mathbf{W}_t^{*T} \mathbf{S}_t^{**} \mathbf{W}_t^* \right)^{-1} \mathbf{W}_t^{*T} \mathbf{y}_{i,t}^*, \quad (2.7)$$

where $(\mathbf{S}_t^{\#*} = \mathbf{Y}_t^{\#T} \mathbf{Y}_t^* / (N - 1))$ and $(\mathbf{S}_t^{**} = \mathbf{Y}_t^{*T} \mathbf{Y}_t^* / (N - 1))$.

2.2.3 Bayesian Methods

Continuous latent variable models [23] are generalization of both PCA and matrix factorization models in which the latent variables are treated as random variables. For real valued datasets, it is convenient to use Gaussian distributions for the representation of both latent and observed variables, which results in linear Gaussian framework. This framework is regarded as a probabilistic version of PCA. The main advantage of this approach is to allow EM to be used for parameter estimation which has many benefits such as computational efficiency and allowance to avoid calculation of sample covariance matrix which makes it suitable for big datasets. Since the graphical model is directed from the latent variables to observed variables, sample generation and the missing value inference can easily be performed. It also leads to estimation of the number of latent variables of the model without cross validation within a Bayesian framework [23]. Continuous latent variable model in case of numerical observations is defined as

$$p(\mathbf{x}_i | \mathbf{z}_i, \Theta) = \mathcal{N}(\mathbf{x}_i | \mathbf{W} \mathbf{z}_i + \mathbf{m}, \Sigma_x), \quad (2.8)$$

where Σ_x is the noise covariance matrix of the observed variables. Note that $\Sigma_x = \mathbf{0}_K$ corresponds to the classical PCA model [23]. The prior distribution over the latent variables is defined as normal distribution, i.e., $p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i | \mathbf{0}_K, \mathbf{I}_K)$. In this setting, the posterior distribution of the latent variables is Gaussian such that

$$p(\mathbf{z}_i | \mathbf{x}_i, \Theta) = \mathcal{N}(\mathbf{z}_i | \boldsymbol{\mu}_i, \Sigma_i) \quad (2.9)$$

EM algorithm can efficiently be used to estimate the parameters of posterior distribution $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1:N}$ as well as the model parameters $\Theta = \{\mathbf{Z}, \mathbf{W}, \mathbf{m}\}$ by using the known variables of the matrix.

VBPCA is fully Bayesian treatment of the linear latent variable model defined above, specifically proposed for missing value estimation [10]. Unlike PPCA models [23], VBPCA algorithm treats all the model parameters as random variables in addition to the factors. By using the proper prior distributions, the model itself penalizes more complex distributions which naturally avoids over-fitting. Gaussian priors for the parameters are convenient to be used for real valued datasets.

$$p(\mathbf{m}) = \mathcal{N}(\mathbf{m}|0, v_m \mathbf{I}_K) \quad (2.10)$$

$$p(\mathbf{W}) = \prod_{j=1}^K \mathcal{N}(\mathbf{w}_j|0, v_{m,j} \mathbf{I}_K) \quad (2.11)$$

The model assumes factorization over all the factor loadings those are packed in matrix \mathbf{W} . The parameter set $\Theta = \{\mathbf{Z}, \mathbf{W}, \mathbf{m}\}$ and hyper-parameter set $\zeta = (v_y, v_m, v_{m,j})$ are to be inferred given the observed variables.

After treating model parameters as random variables, posterior of the parameters is not tractable anymore. Instead of trying to infer exact the posterior $p(\Theta|\mathbf{X}, \zeta)$, it is approximated through a simple distribution $q(\Theta)$, which is a form of variational inference. Cost function of the model that should be minimized is given as

$$\begin{aligned} C(q(\Theta), \zeta) &= \int q(\Theta) \log \frac{q(\Theta)}{p(\mathbf{X}, \Theta|\zeta)} d\Theta \\ &= \int q(\Theta) \log \frac{q(\Theta)}{p(\Theta|\mathbf{X}, \zeta)} d\Theta - \log p(\mathbf{X}|\Theta), \end{aligned} \quad (2.12)$$

where the first term on the right hand side of the equation is KL divergence between the exact and the approximate posterior while the second term is the likelihood of the model that is to be maximized. Since the KL divergence is always positive, it forms a lower bound for the model likelihood. VBPCA algorithm introduces factorized form for the approximate posterior as follows:

$$q(\Theta) = \prod_{j=1}^D q(m_j) \prod_{j=1}^D q(\mathbf{w}_j) \prod_{i=1}^N q(\mathbf{z}_i) \quad (2.13)$$

Each factor in the parameter set is assumed to be a Gaussian distributed posterior in which the parameters of these distributions are inferred from the observed variables by using EM algorithm. EM steps are performed given the values in the set Ω to calculate the means and covariance/variance of the distributions of the parameters as well as point estimates of the hyper-parameters.

2.2.4 Deep Learning Models

Auto-encoders are popular deep neural network models to find the lower dimensional representations of the observations [24]. Traditional auto-encoders does not have probabilistic interpretation since the latent variables are assumed to be deterministic thus estimated point wise. For example, in a one hidden layer network, the mapping from the observed space to the latent space is defined over the parameter set $\Theta_e = \{\mathbf{V} \in \mathbb{R}^{K \times D}, \mathbf{b} \in \mathbb{R}^{1 \times K}\}$, which is called encoding parameters. Similarly, the mapping from the latent space to the observed space is defined over $\Theta_d = \{\mathbf{W} \in \mathbb{R}^{D \times K}, \mathbf{m} \in \mathbb{R}^{1 \times D}\}$ which is called decoding parameters. A non-linear activation function is used to introduce non-linearity to the mappings as follows:

$$\mathbf{x}_i = f(\mathbf{z}_i, \Theta_d) \quad (2.14)$$

$$\mathbf{z}_i = g(\mathbf{x}_i, \Theta_e) \quad (2.15)$$

Reconstruction of an observation under trained model parameters is obtained by passing through the encoder and the decoder respectively.

$$r(\mathbf{x}_i | \Theta_d, \Theta_e) = f(g(\mathbf{x}_i, \Theta_e), \Theta_d) \quad (2.16)$$

AutoREC is an auto-encoder based algorithm proposed to deal with missing values in a dataset [16]. The parameters of the model $\Theta = \{\Theta_d, \Theta_e\}$ are inferred by using the observed

variables of the input matrix thus the objective is defined with the addition of L2 norm regularization on the parameters as follows

$$\min_{\Theta} \sum_{i,j \in \Omega} (x_{ij} - r(x_{ij}|\Theta))^2 + \lambda h(\Theta) \quad (2.17)$$

Variational auto-encoders are probabilistic extensions of traditional auto-encoder which assumes a similar graphical model with the linear latent variable models but the mappings are non-linear through MLPs. The main objective is still to maximize the likelihood of data given the model parameters i.e. $p(\mathbf{X}|\Theta)$. The distribution of the observed variables are modeled as Gaussian.

$$p(\mathbf{x}_i|\mathbf{z}_i, \Theta_f) = \mathcal{N}(\mathbf{x}_i|f(\mathbf{z}_i, \Theta_d), \Psi), \quad (2.18)$$

where $\Psi = \sigma^2 \mathbf{I}_D$. Mean of the distribution of each observation is linked with corresponding latent variables through a non-linear mapping called as decoder, that is implemented by using MLP. The prior distribution over the latent variables is the same as the continuous latent linear models such as $p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I}_K)$. The cost function that should be minimized to maximize the data likelihood is the same as the VBPCA algorithm which is given by Eq. 2.12. The main difference is on the definition of approximate posterior over the latent variables. VBPCA uses factorized Gaussian distributions, however VAE uses MLP, which is called the encoder network defined as

$$q(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}(\mathbf{x}_i, \Theta_{\mu}), \boldsymbol{\Sigma}(\mathbf{x}_i, \Theta_{\Sigma})) \quad (2.19)$$

The parameters of the networks $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and Θ_d are optimized by using stochastic gradient descent [19] although VBPCA uses EM algorithm to infer.

2.3 Alternating Neural Network Model

FSNMC is a MLP based approach developed specifically for matrix completion task [9]. We introduce D MLPs, where D is number of observed features, with independent parameters by

aiming to increase the model capacity for better accuracy compared to the models described above. Each of the MLPs is associated with a single feature in which the parameters are optimized within a regression framework. It uses an alternating scheme, like PCA methods described in Section 2.2.1, such as the parameters are optimized given the values of the observed variables as the first step, then the parameters are exploited to estimate the associated missing variables in the second step. This iterative scheme is performed until convergence of the missing variable estimates.

We initialize D independent MLPs associated with each feature denoted as f_j . The associated variable of the network is removed from the input variable vector so that the number of input nodes for each network becomes $D - 1$. The removed variable becomes a single output node to create a regression model. The corresponding model is given as

$$x_{ij} = f_j(\mathbf{x}_{i(\sim j)}, \Theta_j), \quad (2.20)$$

where $j = 1, \dots, D$ corresponds to the network index for that feature. The model assumes the variable x_{ij} is linked with the other variables $\mathbf{x}_{i(\sim j)}$ for that observation through the parameters Θ_j . Since the algorithm has an alternating scheme, the model parameters are optimized at each iteration given the completed matrix from the previous iteration. Therefore, we use $\Theta_{j,t}$ for the parameters of the j th network during the iteration t . Objective is to find the optimal $\Theta_t = \{\Theta_{j,t}\}_{j=1, \dots, D}$ that minimizes the cost function below at each iteration.

$$\min_{\Theta_{j,t}} \sum_{(i,j) \in \Omega} (x_{ij,t} - f_j(\mathbf{x}_{i(\sim j),t}, \Theta_{j,t}))^2 + \lambda h(\Theta_{j,t}) \quad (2.21)$$

For example, for a two layer MLP network structure, the parameter set that is to be optimized will be $\Theta_{j,t} = (\mathbf{W}_{1j,t} \in \mathbb{R}^{D \times M}, \mathbf{W}_{2j,t} \in \mathbb{R}^{M \times 1}, \mathbf{m}_{1j,t} \in \mathbb{R}^{M \times 1}, \mathbf{m}_{2j,t} \in \mathbb{R}^{1 \times 1})$. The graphical model that represents this structure is given in Figure 2.1. Nonlinear activation functions of MLPs can be chosen as sigmoid or RELU. The cost function is similar to the one of AutoREC. However, unlike AutoREC, we optimize the network parameters iteratively after updating the missing values. The cost function includes a regularization term to avoid over-fitting with strength parameter λ .

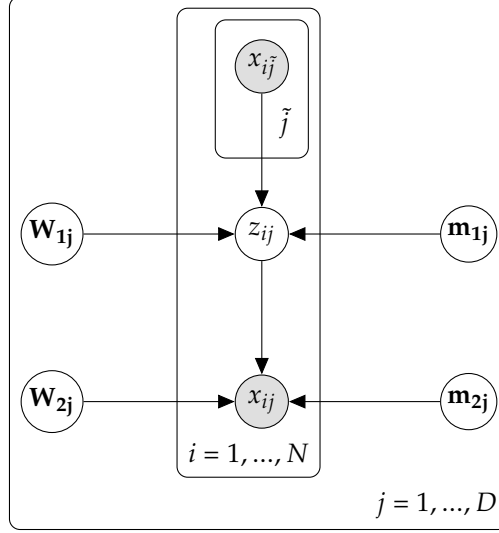


Figure 2.1: Graphical model of FSNMC for two layer MLP configuration

A convenient choice is to use L2 norm on the weight parameters that shrinks them towards small numbers.

$$h(\Theta_{j,t}) = \|\mathbf{W}_{1j,t}\|^2 + \|\mathbf{W}_{2j,t}\|^2 \quad (2.22)$$

After optimizing the parameters of the networks at iteration t , the model parameters are exploited to infer missing values. Given the parameters $\Theta_{j,t}$, reconstruction of the missing variable $x_{ij,t}$ is obtained with a single forward pass through each network.

$$r(x_{ij,t} | \Theta_{j,t}) = f_j(\mathbf{x}_{i(\sim j)}, \Theta_{j,t}) \quad (2.23)$$

Instead of updating each missing variable with the reconstruction of that variable, we introduce a damping parameter to smooth out the learning, which was empirically observed as providing better stability. The final update equation at iteration t is given as

$$x_{ij,(t+1)} = x_{ij,t} + \rho(r(x_{ij,t} | \Theta_{j,t}) - x_{ij,t}), (i, j) \in \bar{\Omega} \quad (2.24)$$

At each iteration, re-training each network with the updated missing variables by using a random parameter initialization scheme would result in substantial increase in training time. We

Algorithm 1 FSNMC algorithm

```

1: procedure FSNMC( $\mathbf{X}, \Omega, \mathbf{X}_t$ )
2:   Initialize parameters  $\Theta_0 = \{\Theta_{j,0}\}_{j=1,\dots,D}$  randomly;
3:   Normalize  $\mathbf{X}$ ;
4:    $x_{ij,(0)} = 0, (i, j) \in \bar{\Omega}$ ; ▷ Initialize missing variables
5:    $t = 1$ ;
6:   while  $\text{conv}(\mathbf{X}_t - \mathbf{X}_{t-1})$  do ▷ Check for convergence
7:     for  $j = 1 : D$  do ▷ Parameter inferring loop
8:        $\Theta_{j,t}^0 = \Theta_{j,t-1}^F$ ; ▷ Transfer parameters
9:        $\Theta_{j,t}^F = \min_{\Theta_{j,t}} \sum_{(i,j) \in \Omega} (x_{ij,t} - f_j(\mathbf{x}_{i(\sim j),t}, \Theta_{j,t}))^2 + \lambda h(\Theta_{j,t})$  ▷ Train
10:      for  $j = 1 : D$  do ▷ Model exploiting loop
11:         $r(x_{ij,t} | \Theta_{j,t}^F) = f_j(\mathbf{x}_{i(\sim j)}, \Theta_{j,t}^F)$ ; ▷ Reconstruct missing variables
12:         $x_{ij,(t+1)} := x_{ij,t} + \rho(r(x_{ij,t} | \Theta_{j,t}^F) - x_{ij,t}), (i, j) \in \bar{\Omega}$ ; ▷ Update
13:       $t = t + 1$ ;
14:   return  $\mathbf{X}_t$ 

```

instead introduce a network parameter storage method between the iterations where each network's final weights are carried through. For example, at iteration $t = 0$, the model parameters are initialized with small random numbers as it is customary in neural network training. After the first iteration, optimized model parameters lead to a minima point over the surface of the cost function. Since the difference between two consecutive iterations of the network are the slowly changing in terms of missing values in the dataset, beginning from a random point on the cost function surface for the next optimization would be time consuming. Assuming that the point found in the previous optimization would be close to the minima of the cost function of the previous iteration, the number of epochs needed to reach optimal minima shall reduce dramatically with the application of this method. In the beginning of each iteration, the stored parameters are assigned as follows

$$\Theta_{j,t+1}^0 = \Theta_{j,t}^F, \quad (2.25)$$

where $\Theta_{j,t+1}^0$ corresponds to the initial parameters of network j at iteration $t + 1$ and $\Theta_{j,t}^F$ corresponds to the optimized parameter at iteration t .

The proposed algorithm differs from the competitive methods with the following aspects:

- Linear latent variable models assume common latent variables for all features. However, FSNMC defines separate set of latent variables for each feature which increases both model capacity and flexibility.
- PCA, Matrix Factorization and Auto-encoder methods try to minimize a common reconstruction error defined over all of the observed variables of incomplete matrix. However, FSNMC defines an independent optimization goal for each feature.
- TSR algorithm assumes a linear relationship between observed and unobserved variables of each instance and the regression framework is based on this linear relationship over the factor loading matrix. However, FSNMC uses MLPs to extract non-linear relationships between the associated feature and the remaining features, and the parameters of each network are optimized for better prediction instead of better representation.
- Although our approach is predictive model unlike the generative VBPCA and VAE, for matrix completion task, most of the time, the accuracy is more important than expressing domain knowledge. FSNMC is focused on accuracy by modeling a non-linear strong regression framework.

2.4 Comparative Study

2.4.1 Experimental Setup

Comparative experiments are performed using five baseline algorithms including TSRE, VBPCA, AutoREC, VAE and FSNMC. Four small-sized and two medium-sized real world datasets are chosen to analyze the performances whereas two medium-sized synthetic datasets are also used to see the performance variation when the process behind the data generation changes. In total, the six real world datasets we have chosen are labeled as concrete, nutrient, wine, olive oil, protein and abalone as described in detail below. The concrete dataset ($N = 103, D = 10$) describes three mechanical features of concrete according to the concentration of seven ingredients for different samples of concrete for a total number of 10 features. The dataset is complete which means no

missing value exists inherently and it is available on the University of California Irvine’s Machine Learning Repository (UCI-MLR) [25]. The nutrient dataset ($N = 104, D = 20$) describes the nutritional value of different vegetables. For some vegetables, the content in some nutrients is denoted as “traces”, in this case, the content of the nutrient was assumed to be negligible and zero. The dataset is maintained by Health Canada and already contains 1.9% of missing values [26]. The wine dataset ($N = 173, D = 13$) describes the chemical attributes of wine samples. The dataset is complete and is available on the UCI-MLR [27]. Olive oil dataset ($N = 572, D = 8$) describes the concentration of fatty acids in the wine products produced in different regions. The dataset is complete and was compiled by Forina et al. [28]. Unlike the previous datasets, the protein dataset ($N = 45730, D = 9$) includes a significantly higher number of samples and describes physicochemical properties of the protein tertiary structure. The dataset is complete and is available on the UCI-MLR [29]. The final dataset to round up the real world examples is Abalone ($N = 4177, D = 8$) which consists of attributes of the abalones, that are measured for each individual animal [30]. The dataset has gender information which is a categorical variable thus removed from the dataset to obtain a homogeneous real valued dataset. Future work will consider including the mixed datatype datasets with both numerical and categorical features. Within all real-life datasets, the features are correlated with each other to some extent (from weak to strong), that can be seen by computing covariance matrices.

2.4.2 Implementation Details

TSRE algorithm is an iterative method where SVD is performed at each iteration to find the factor loading matrix \mathbf{W}_t . After the regression is performed to update the missing values for each instance by using Eq. 2.7, other parameter of the model m_t is evaluated by applying column-wise mean operation on the matrix updated with new predictions. The model parameter set that is to be inferred is $\Theta_t = \{\mathbf{W}_t, m_t\}$ and the hyper-parameter set to be optimized by cross-validation is $\zeta = (\gamma, K)$ in which γ defines the fraction of the observed variables used for early stopping and K corresponds to the number of latent variables.

VBPCA algorithm uses the observed variables of each instance to update both the parameter and hyper-parameter sets. The model parameter set to be inferred is $\Theta = (W, Z, m)$ and hyper-parameter set is $\zeta = (v_y, v_m, v_{m,j}, K)$. Each parameter is inferred by the application of EM algorithm except hyper-parameter K , which is optimized via cross-validation.

For MLP based approaches like AutoREC, VAE and FSNMC the network structure such as the number of hidden layers l and the number of latent nodes K in a hidden layer are chosen as hyper-parameters optimized via cross-validation. For these algorithms, the parameters defining the network structure are chosen from the sets $l = \{1, 2\}$ and $M = \{2, 3, 4, 5\}$ to obtain fair performance comparison. The parameters of these approaches are inferred by using the stochastic gradient descent method. AutoREC has encoder and decoder MLP parameters $\Theta = \{\Theta_d, \Theta_e\}$ optimized with stochastic gradient descent by using the observed variables. Cross-validation is used to optimize the network structure and regularization strength λ such that $\zeta = (M, l, \lambda)$. VAE, on the other hand, does not have a regularization parameter, i.e, the hyper-parameter set is $\zeta = (K, l)$. Indeed, the regularization comes from the prior on latent variables [19]. The parameter set $\Theta = \{\Theta_\mu, \Theta_\Sigma, \Theta_f\}$ of the model is also inferred by using stochastic gradient descent.

Unlike VAE and AutoREC, FSNMC is an iterative algorithm such that the parameter optimization is performed at each iteration t given the updated missing values from the previous step. Another major difference is the number of networks associated with the number of features. In this case, the parameter set of the algorithm that needs to be inferred by using stochastic gradient descent method at iteration t is given as $\Theta_t = \{\Theta_{j,t}\}_{j=1,\dots,D}$. The hyper-parameter set of the model is $\zeta = (K, l, \lambda, \gamma, \rho)$. Two of these parameters, γ which corresponds to the fraction of observed variables separated to be used for early stopping and ρ , which is the damping parameter are fixed as %5 and 0.1 respectively. FSNMC algorithm is implemented by using the following Python libraries: TensorFlow, Numpy and Scipy. The complete script is accessible on GitHub repository at <https://github.com/maktukmak/NNMC>.

2.4.3 Cross-validation and Evaluation Procedure

Hyper-parameters of the algorithms are optimized using cross-validation. The procedure starts with separating some of the observed variables and keeping them in a separate validation set. The fraction of splitting is fixed at %5 of the observed variables such that %95 of the observed variables was used to infer parameter set Θ while the rest is used for optimizing hyper-parameter set ζ . Since the experiments are repeated with varying fractions of missing values, hyper-parameter optimization is repeated for each algorithm, dataset and fraction of missing value combinations for each experiment. Each optimization exploit grid search with combination of the elements of ζ , which means $|\zeta|!$ experiments are performed for each scenario. Mean square error is used as the performance metric for different hyper-parameter combinations.

$$MSE_{val} = \frac{1}{|\Omega_{val}|} \sum_{i,j \in \Omega_{val}} (x_{ij} - r(x_{ij}|\Theta, \zeta))^2 \quad (2.26)$$

Each experiment is repeated ten times to obtain a statistical average performance and the set that gives the lowest error is chosen.

Given the optimized hyper-parameter set for each test scenario, we run the algorithms ten times by merging validation and training sets to obtain the best performances for each algorithm. The experimental setup has differences for small and medium datasets. We use Missing Completely at Random (MCAR) mechanism to remove %5, %20, %40 and %70 of the variables of small datasets in a test set for performance comparison. For medium datasets, we setup an additional experiment with %90 fraction of missing values to observe the performances at a relatively sparse setting. Removing %90 of the variables in small datasets resulted in artificially low performance for each algorithms likely due to having very few variables to infer the model parameters. The reported performance metric is the mean square error evaluated on the test set.

$$MSE_{test} = \frac{1}{|\Omega|} \sum_{i,j \in \Omega} (x_{ij} - r(x_{ij}|\Theta, \zeta))^2 \quad (2.27)$$

To perform experiments, the high-performance computing cluster of University of South Florida is used to parallelize the experiments among several processors.

Table 2.1: MSE comparison on small datasets

	FSNMC	TSRE	VBPCA
Concrete			
%5	0.14 (\pm 0.02)	0.15 (\pm 0.03)	0.23 (\pm 0.03)
%20	0.33 (\pm 0.03)	0.39 (\pm 0.05)	0.37 (\pm 0.04)
%40	0.59 (\pm 0.07)	0.70 (\pm 0.08)	0.74 (\pm 0.07)
%70	0.94 (\pm 0.06)	0.95 (\pm 0.06)	0.96 (\pm 0.05)
Nutrient			
%5	0.36 (\pm 0.03)	0.44 (\pm 0.04)	0.43 (\pm 0.03)
%20	0.48 (\pm 0.03)	0.52 (\pm 0.05)	0.52 (\pm 0.03)
%40	0.63 (\pm 0.05)	0.66 (\pm 0.07)	0.67 (\pm 0.06)
%70	0.81 (\pm 0.07)	0.83 (\pm 0.07)	0.83 (\pm 0.06)
Wine			
%5	0.47 (\pm 0.03)	0.51 (\pm 0.03)	0.49 (\pm 0.02)
%20	0.51 (\pm 0.04)	0.56 (\pm 0.03)	0.53 (\pm 0.04)
%40	0.62 (\pm 0.06)	0.64 (\pm 0.04)	0.61 (\pm 0.05)
%70	0.87 (\pm 0.05)	0.82 (\pm 0.06)	0.79 (\pm 0.04)
Olive Oil			
%5	0.21 (\pm 0.02)	0.21 (\pm 0.04)	0.22 (\pm 0.03)
%20	0.28 (\pm 0.03)	0.33 (\pm 0.03)	0.31 (\pm 0.04)
%40	0.47 (\pm 0.07)	0.50 (\pm 0.06)	0.48 (\pm 0.06)
%70	0.74 (\pm 0.07)	0.74 (\pm 0.07)	0.71 (\pm 0.06)

2.4.4 Results and Discussion

The performance comparison for small scale datasets is given in Table 2.1. The table is compiled with all the results for FSNMC, TSRE and VBPCA algorithms. AutoREC and VAE algorithms are excluded due to significantly lower accuracy, most likely due to the small size of the dataset whereas these algorithms are proposed for medium to large sized datasets. Bold values correspond to the minimum errors corresponding to that specific test case. FSNMC clearly outperforms other algorithms in a majority of the scenarios including small-sized datasets, especially for low fractions of missing values. In order to demonstrate a holistic view of performance, Figure 2.2 shows the average MSE values over all the datasets for each of the varying fractions of missing

values. Except the fraction of %70, we can conclude that FSNMC is the superior choice to infer missing values regardless of the dataset.

Table 2.2: MSE comparison on medium datasets

	FSNMC	TSRE	VBPCA	AutoREC	VAE
Protein					
%5	0.21 (\pm 0.02)	0.24 (\pm 0.02)	0.24 (\pm 0.03)	0.32 (\pm 0.03)	0.32 (\pm 0.02)
%20	0.22 (\pm 0.02)	0.26 (\pm 0.03)	0.25 (\pm 0.02)	0.34 (\pm 0.04)	0.34 (\pm 0.05)
%40	0.30 (\pm 0.04)	0.30 (\pm 0.04)	0.30 (\pm 0.03)	0.37 (\pm 0.03)	0.39 (\pm 0.04)
%70	0.44 (\pm 0.04)	0.47 (\pm 0.03)	0.44 (\pm 0.03)	0.50 (\pm 0.05)	0.50 (\pm 0.07)
%90	0.81 (\pm 0.06)	0.72 (\pm 0.05)	0.73 (\pm 0.05)	0.75 (\pm 0.07)	0.76 (\pm 0.06)
Abalone					
%5	0.15 (\pm 0.03)	0.15 (\pm 0.03)	0.30 (\pm 0.08)	0.30 (\pm 0.04)	0.36 (\pm 0.05)
%20	0.16 (\pm 0.03)	0.21 (\pm 0.04)	0.23 (\pm 0.05)	0.34 (\pm 0.06)	0.34 (\pm 0.06)
%40	0.24 (\pm 0.04)	0.25 (\pm 0.04)	0.27 (\pm 0.05)	0.36 (\pm 0.05)	0.38 (\pm 0.06)
%70	0.48 (\pm 0.06)	0.39 (\pm 0.05)	0.40 (\pm 0.04)	0.51 (\pm 0.06)	0.49 (\pm 0.04)
%90	0.84 (\pm 0.05)	0.71 (\pm 0.05)	0.71 (\pm 0.04)	0.82 (\pm 0.07)	0.76 (\pm 0.07)

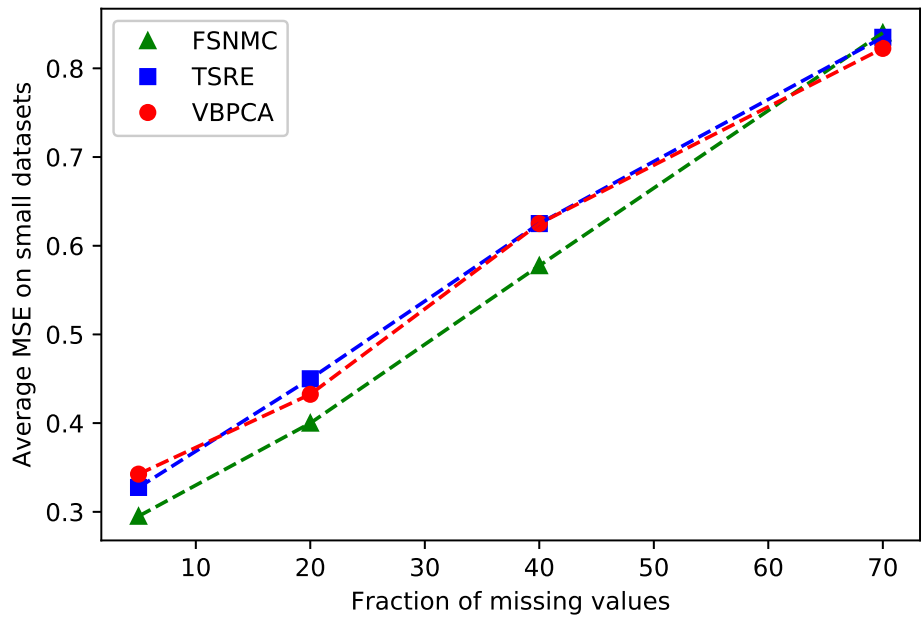
The MSE performances for medium scale datasets is compiled in Table 2.2. As expected, AutoREC and VAE methods now result in competitive performance. However, especially for low fractions of missing values, FSNMC still outperforms all other approaches regardless of the dataset whereas TSRE and especially VBPCA provide better accuracy at higher fractions. Figure 2.2.b shows the average MSE values over all the datasets for each of the varying fractions of missing values where the proposed FSNMC method and VBPCA outperform competing algorithms at low and high fractions of missing values respectively.

2.5 Conclusions and Future Work

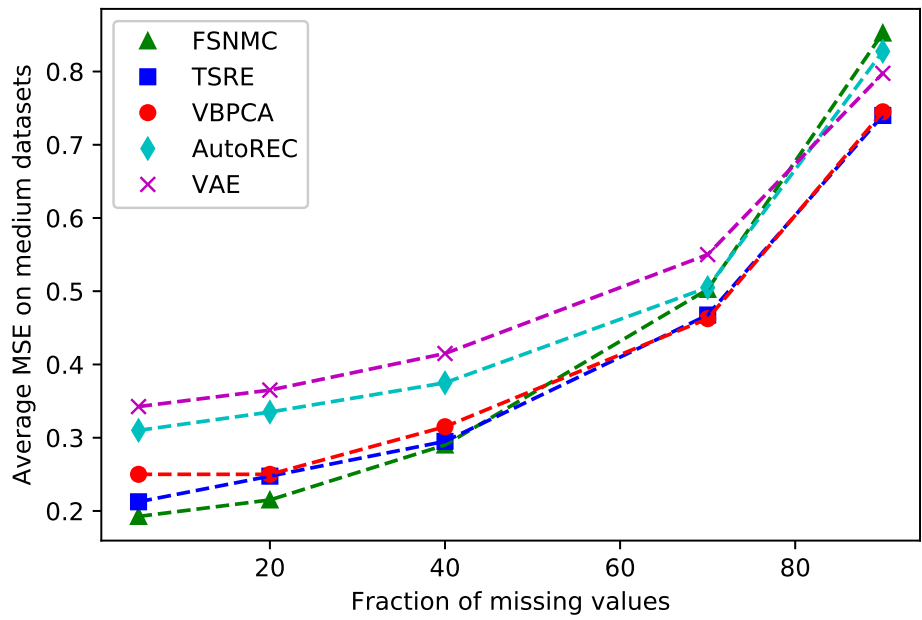
FSNMC is an MLP based matrix completion algorithm to deal with missing values specifically in the scientific experimental datasets, which are characterized as having low level sparsity and size of small to medium. The experimental study in this chapter includes a wide range of combinations on the dataset size and the fraction of missing values, and demonstrates the superior performance of the proposed model as compared to the baseline models. FSNMC can be an ideal solution if one have small-sized dataset and desire to impute the missing values with high accuracy.

If the dataset size gets larger, e.g., a medium sized dataset, the algorithm can still be an ideal solution if the fraction of missing values is comparatively low, i.e, under %50.

However, as the number of features increases, the number of parallel networks associated to each feature required for FSNMC algorithm will increase. Although the number of parameters linearly increases with respect to the number of features, the training time, i.e,computational complexity, will increase exponentially due to the alternating training scheme of the networks. As a result, the disadvantage of this model is that it can not efficiently scale in terms of computation time for the datasets with large number of features, i.e, the fat datasets. To overcome this issue, one should avoid the algorithms requiring alternating schemes for the training. The obvious solution is to refer to the generative models, which are efficient in terms of computation. The next chapter will introduce such a model that can further incorporate additional side information to improve the accuracy of the imputation, which is a very useful property for the recommender systems.



(a) On Small Datasets



(b) On Medium Datasets

Figure 2.2: Average Mean Square Error

Chapter 3: Heterogeneous Data Fusion

3.1 Introduction

The ultimate objective of a recommender system is to learn the user patterns explicitly or implicitly by using the available information. Depending on the platform, the information sources can include, i) the past interactions and feedbacks between the users and the items, ii) the demographic information of the users, iii) the features directly related to the items, iv) the social relationships/trust between the users, v) the network structures connecting the users/items based on specific criteria, vi) user-contributed information such as textual reviews and vii) cross-domain knowledge from external domains [31, 32]. Collaborative filtering methods use only the interactions between the users and the items to learn the user patterns. In contrast, content-based methods use only the side information which may be provided by the users (such as gender, age, occupation) or the content provider (such as movie genre, year). The hybrid recommender system combines both approaches for a demonstrably more successful overall recommendation accuracy for a variety of applications [2]. Specifically, the scenarios where the users and the items have very few observed interactions, called the cold-start scenario, represents a particular challenge which can be better resolved by the hybrid recommendation systems that can combine the more readily available and less sparse user and item side information sources with their significantly more sparse interactions [31].²

Incorporating side information is an intuitive solution for the cold start problem. In most cases, the users and items have available demographic information that can be used as relevant side information. In the case where a user has no interactions (examples include new users or existing users trying out a new category of products) but still some available side information like the age,

²Part of this chapter was published in [33]. Permission is included in Appendix A.

gender, occupation, etc. the system can still infer a recommendable item for this user based solely on the side information. On the contrary, a warm-start scenario is considered when a user has too many interactions where his/her pattern implies more about the user than their demographic information. A hybrid system should use the information sources efficiently and make an accurate inference for both the cold-start and warm-start settings. However, the process of incorporating side information is non-trivial due to the diversity and heterogeneity of the side information data format. For example, for a movie, the release date corresponds to a numerical feature, whereas its genre has a categorical value. Nonetheless, both should be incorporated appropriately according to the nature of these observations.

Probabilistic generative models are powerful tools that can be used for the datasets, including many missing values. They allow missing data to be handled in a principled way by marginalizing over the distribution of the unobserved variables [34]. Since these models account for the uncertainty of the latent variables, they also handle over-fitting problem, that occurs severely in the case of very sparse data, by regularizing the latent variables with proper priors [35]. Therefore, these models are particularly useful for recommender systems with the condition that one can overcome the challenge of incorporating heterogeneous multi-modal side information which represents the main objective of our study. In this chapter³, we propose a multi-modal generative model and a variational Expectation Maximization (EM) algorithm to infer the latent representations of both the users and the items to leverage the emerging variational inference methods [36]. Our method demonstrates the efficient incorporation of mixed data type side information in a scalable probabilistic generative framework. In summary, our contributions can be summarized as follows:

- A novel probabilistic generative model that can incorporate mixed data type side information. The natural parameters of the side information sources are regressed from the latent variables. This allows the incorporation of any data type that can be modeled with the exponential family distribution, although we mainly focus on categorical and numerical features. Since the model

³Part of this chapter was published in [33]. Permission is included in Appendix A.

is generative, it can make an inference not only in the presence of missing values in the rating matrix but also in the side information as well.

- A fundamental solution to solve the problem of intractable inference, which emerges due to the data type variety of the side information, by deriving a variational EM method that turns the inference into an optimization problem. By using the appropriate local quadratic approximations, the posterior distributions of the user and the item latent variables are approximated as Gaussian motivated by the Bernstein-Von Mises theorem [37].
- A reduced computational complexity which scales with the product of the number of items and the number of users (i.e., only linear in each dimension), which makes it suitable for large datasets.
- The state-of-the-art performance on both synthetic and real datasets when compared to a wide range of well-established baselines as well as some more recent contributions.

This chapter is organized as follows. The prior related work is given in Section 3.2. The model is defined in Section 3.3.1 and the corresponding inference method is derived accordingly in Section 3.3.2. Computational complexity analysis is given in Section 3.3.3. Detailed experimental studies including the simulation results and a wide range of performance comparisons on real-world datasets are presented in Section 3.4. Lastly, the chapter concludes with possible directions for future work in Section 5.

3.2 Related Work

In recent years, researchers have shown greater interest in incorporating side information specifically to solve the cold-start problem [31,32]. We group the related studies into four categories based on how the side information is being treated. The category of baseline Matrix Factorization (MF) models consists of algorithms that do not use side information; that is, the only source of information is the rating matrix. Prior based models use the side information to regularize the latent space of the users and the items. Regression-based models incorporate the side information into

a common latent space shared by both the interactions and the features. Disjoint models remove the latent space sharing property and assume that the rating and side information are generated independently.

3.2.1 Baseline MF Models

Linear latent variable models such as principal component analysis (PCA) [35] and matrix factorization (MF) [1] have originally led the way in matrix completion tasks such as recommender systems. In particular, MF has been a milestone in collaborative filtering. In this model, based on the assumption that the sparse rating matrix is low rank, the users and the items are mapped into a joint low dimensional space such that the ratings are modeled as the products of the representations in this space. The model optimizes the latent representations to explain the observed interactions by using Stochastic gradient descent or alternating least squares methods. However, sparse nature of the observed data makes the optimization highly prone to over-fitting. Probabilistic matrix factorization (PMF) [38] extends the MF models by introducing zero-mean Gaussian priors for the latent variables for more robust performance in terms of over-fitting. The priors result in L2 norm regularization for the latent variables if one performs MAP estimation for the model parameters. The regularization strength corresponds to the variances of the Gaussian priors which are optimized via cross-validation procedure. However, it is still prone to over-fitting unless the regularization parameters are chosen carefully. Bayesian probabilistic matrix factorization (BPMF) [39] further extends the PMF model by using Gaussian-Wishart priors for the means and the co-variances of the latent variables instead of the standard zero mean and identity covariance. That leads to computing posterior of the latent variables instead of the point estimates, which is useful for modeling uncertainty of the variables. Since the complexity is controlled automatically based on the training data, the model is more robust to the hyper-parameter selection. Some further recent extensions on top of the aforementioned models include, i) the local matrix factorization [40–42], which extends the PMF model by introducing local estimation emerged from the idea of mixture models [43], ii) the mixture rank approximation, which models the rank of the rating matrix in

the mixture model with a Laplacian prior to infer the latent space dimension automatically [44] and iii) neural network models [45–48] as alternative factorization methods in order to replace the linear models with their non-linear counterparts containing many free parameters, which may cause sensitivity to over-fitting due to the sparse nature of the observations.

3.2.2 Prior Based Models

Prior based models incorporate the user and item features by forming a prior for the user and item latent vectors. A stochastic process given by a polynomial function of features is used to regularize the latent variables of both users and items in [49]. Applying feature-based regression to the priors of the latent variables instead of zero-mean Gaussian priors (as in PMF) lead the way to incorporate side information where a Monte Carlo EM was used to fit the model. Similarly, in [50], the priors of the user and item latent variables are regressed from the features vectors. Factorized Gaussian priors are given as the regression coefficients with a mean field assumption for variational inference. Kernelized probabilistic matrix factorization (KMF) [51] model assigns Gaussian process priors to the latent factors. The covariance of the priors is derived from the similarity matrix evaluated from the side information of the users and the items. Recently, in [52], the similarity-like matrix, which is called the user-to-user topic inclusion degree based sparse network, is introduced for social-network link prediction. The network is fused with the observed interaction matrix through a probabilistic model where the side information is used as the mean prior.

3.2.3 Regression-Based Models

Regression-based models assume that the side information and the latent vectors of the users and the items are linearly dependent. In [53], the BPMF model is extended to incorporate the side information by performing a linear regression on the real-valued features of the users and the items. Dirichlet prior is added to the model for local estimation to improve the performance further where collapsed Gibbs sampling is used to fit the data. In another work [54], probabilistic

modeling is combined with matrix factorization where the side information consists of the observed words in the articles. A latent topic space is used for fitting by introducing the latent Dirichlet allocation model [55] with regression in the item latent space for joint estimation. Maximum a posterior (MAP) estimation of the latent variables is performed with the EM algorithm. In [56], the similarity matrices are used within a generative model, i.e. the latent factors are assumed to be the ancestors of the similarity matrices in the graphical model. The regression parameters and the latent factors are optimized jointly. An extended work in [57] introduces locality constraint into the latent space to learn local collective embeddings (LCE). [58] proposes an algorithm where the Gaussian process regression is used to incorporate the real-valued features to the matrix factorization model where the probit likelihood is used for preference ranking. For inference, the EM algorithm is used along with the expectation propagation approximation for non-Gaussian likelihoods. Variational auto-encoders are also used in probabilistic learning of feature latent representations [59]. By following [55], the article recommendation is performed by replacing the LDA model with a variational auto-encoder. Additionally, especially for contextual recommendation, factorization machines [60] and tensor factorization methods [61], that can use additional information beyond rating matrix, are proposed. Recently, several algorithms [62–64] are developed based on tensor factorization to transfer knowledge from other domains as side information to alleviate the cold-start problem.

3.2.4 Disjoint Models

Contrary to the previously discussed methods, there is a work in the literature that the rating matrix and side information are assumed to be generated independently, i.e. they don't share the same latent space. In [65], matrix factorization is augmented with regression against the real-valued side information by using a weighted scheme. The side information is assumed to be marginally independent. In [66], the normalized features are added to the latent vectors and stochastic gradient descent is performed as in MF. In [67], side information is used to compute multiple item similarity functions. These functions are weighted for each user with trained weights

to make personalized recommendations. In [68], a dense sub-matrix is extracted from the rating matrix by selecting the users and items with large numbers of interactions. Matrix factorization is then performed to find latent factors of the corresponding users and items. Afterwards, a linear regression model is employed to relate the latent factors and the similarity matrices of the users and the items where the regression weights are evaluated with the selected user/item latent factors. The resulting algorithm is called DecRec. A similar method proposed in [69] uses concatenated attributes instead of a similarity matrix. For warm-start users and items, latent vectors are evaluated by using the factor model to learn a mapping between the attributes and latent vectors. In [70], in-dependency assumption holds for social and geographical information for the task of a point of interest estimation to fuse the sources of information via matrix factorization. A neural network model proposed in [47], in which concatenated item and user features are fed to two auto-encoders to learn low dimensional representations. These representations are then added to the MF model whose networks and latent vectors are jointly optimized.

The proposed algorithm falls in the general category of regression-based models. The differences between our algorithm and the aforementioned models are several folds. First, all the models treat the side information as uni-modal, i.e. the side information is assumed to be of a single data type. Majority of the models [51, 56, 57, 67–69] handle the mixed data problem by pre-processing the similarity measures to form a real-valued similarity matrix as the side information. However, that creates a significant burden on computational costs and memory requirements to the extent where these approaches become unscalable for very large datasets as discussed in Section 3.4. Another problem is their performance relies heavily on the selection criteria of similarity metric which is not straightforward to compute for mixed-data type features. In contrast, the proposed model does not require any pre-processing, i.e the feature dimension is preserved which allows the selection of a lower dimensional latent space for a demonstrably better trade-off between performance and scalability. The mixed data problem is solved by a principle probabilistic generative approach where the real-valued, categorical and binary features are modeled by using Gaussian, categorical and Bernoulli distributions respectively. To demonstrate the improvements

in performance, we pick and compare the representative recent algorithms from all four categories. The comparison details are presented in Section 3.4.

3.3 Proposed Model

3.3.1 Model Definition

In this section, we describe the details of the proposed model. The model is developed with multi-modal side information including one multivariate real-valued and one categorical observation for both the users and the items, and with the sparse rating matrix formed by the interactions. The ultimate goal is to infer the posterior distributions of the user and the item latent variables to explain both the observed ratings and associated side information. The graphical representation of the proposed model is shown in Figure 4.1. In the probabilistic model, $\mathbf{u}_i \in R^K$ corresponds to the latent variable associated with user i and $\mathbf{v}_j \in R^K$ corresponds to the latent variable associated with item j . Zero-mean spherical Gaussian priors are assumed for these multivariate latent variables as follows:

$$p(\mathbf{u}_i) = \mathcal{N}(\mathbf{u}_i | \mathbf{0}_K, \lambda_u^{-1} \mathbf{I}_K), \quad (3.1)$$

$$p(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j | \mathbf{0}_K, \lambda_v^{-1} \mathbf{I}_K), \quad (3.2)$$

where λ_u and λ_v are the precision hyper-parameters for the distributions. K is the latent space dimension. Instead of the zero-mean prior, it is trivial to use Gaussian-Wishart priors as in [39] for fully Bayesian treatment to prevent over-fitting that can easily occur in the case that the regularization precision parameters are not tuned correctly within a validation set. However, for simplicity of derivations, we stick to the zero-mean spherical priors. The generative process assumes that both the real-valued and the categorical side information represented by $\mathbf{x}_i \in R^{D_u}$ and $\mathbf{y}_i \in R^{M_u}$, respectively, are generated from \mathbf{u}_i through the model parameters via regression. We hold on to this assumption since many regression-based models in the literature [53, 54, 57, 71] have proved that modeling the generation process for the features linearly through the natural parameters of

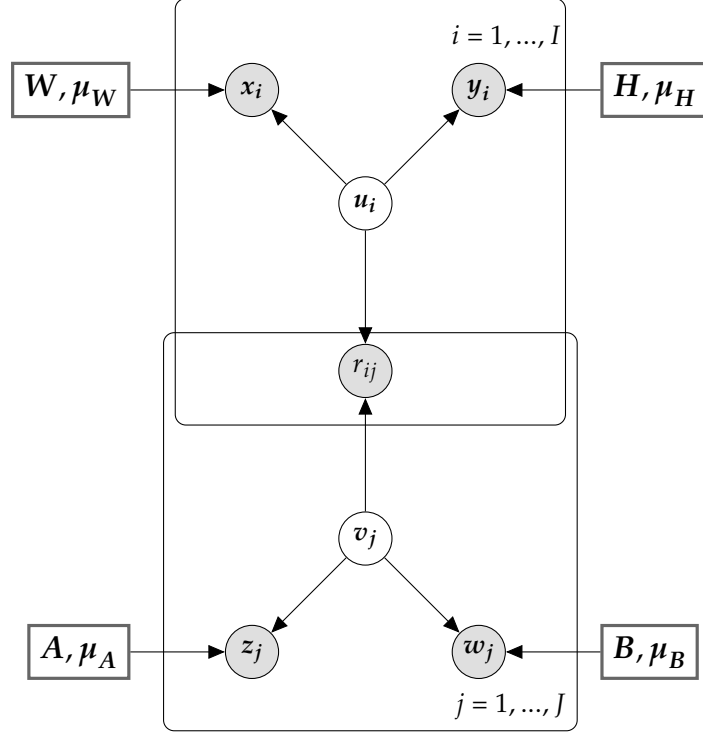


Figure 3.1: Graphical model representation of the proposed MF-MSI method. The upper plate is for the users, the lower plate is for the items, and the intersection is for the ratings. x_i and z_j denote real-valued side information while y_i and w_j denote categorical side information.

their distributions is a valid assumption and results in reasonable performance. Accordingly, the conditional probability of x_i is given as a Gaussian distribution:

$$p(x_i|u_i) = \mathcal{N}(x_i|Wu_i + \mu_W, \Sigma_x), \quad (3.3)$$

where $W \in R^{D_u \times K}$, $\mu_W \in R^{D_u}$ and $\Sigma_x \in R^{D_u \times D_u}$ are the model parameters associated with the real-valued user side information. For y_i , the categorical conditional distribution is assigned as follows:

$$p(y_i|u_i) = \text{Cat}(y_i|\mathcal{S}(Hu_i + \mu_H)), \quad (3.4)$$

where $H \in R^{M_u \times K}$ and $\mu_H \in R^{M_u}$ are the model parameters that are associated with the categorical side information. \mathcal{S} is the Soft-max function that maps the natural parameters of the categorical distribution to the probability of each class. The natural parameters of both distributions

are linearly modeled, i.e., $\eta_{G,i} = \mathbf{W}\mathbf{u}_i + \boldsymbol{\mu}_W$ for the Gaussian distribution, and $\eta_{C,i} = \mathbf{H}\mathbf{u}_i + \boldsymbol{\mu}_H$ for the categorical distribution. $\eta_{G,i}$ corresponds to the mean of the Gaussian distribution, and $\eta_{C,i} = [\log \frac{p_1}{p_{M_u+1}}, \dots, \log \frac{p_{M_u}}{p_{M_u+1}}]^T$ where $\{p_1, \dots, p_{M_u+1}\}$ are the probabilities in the categorical distribution. A symmetric configuration is used for the item side with the following distributions:

$$p(\mathbf{z}_j | \mathbf{v}_j) = \mathcal{N}(\mathbf{z}_j | \mathbf{A}\mathbf{v}_j + \boldsymbol{\mu}_A, \boldsymbol{\Sigma}_z), \quad (3.5)$$

$$p(\mathbf{w}_j | \mathbf{v}_j) = \text{Cat}(\mathbf{w}_j | \mathcal{S}(\mathbf{B}\mathbf{v}_j + \boldsymbol{\mu}_B)), \quad (3.6)$$

where $\mathbf{z}_j \in R^{D_v}$ and $\mathbf{w}_j \in R^{M_v}$ represent the real valued and the categorical side information for item j , respectively. The corresponding model parameters are $\mathbf{A} \in R^{D_v \times K}$, $\boldsymbol{\mu}_A \in R^{D_v}$, $\boldsymbol{\Sigma}_z \in R^{D_v \times D_v}$, $\mathbf{B} \in R^{M_v \times K}$ and $\boldsymbol{\mu}_B \in R^{M_v}$. Finally, the rating matrix is assumed to be generated with the interactions between the user and the item latent variables. The conditional probability for each rating is modeled with the precision parameter c as follows:

$$p(r_{ij} | \mathbf{u}_i, \mathbf{v}_j) = \mathcal{N}(r_{ij} | \mathbf{u}_i^T \mathbf{v}_j, c^{-1}). \quad (3.7)$$

3.3.2 Inference

Next, we infer the posterior distributions of the user and the item latent variables \mathbf{u}_i and \mathbf{v}_j given the observed ratings and the multi-modal side information. We also find the Maximum Likelihood estimations of the global model parameters that are collected in the set Θ as

$$\Theta = \{\mathbf{W}, \boldsymbol{\mu}_W, \mathbf{H}, \boldsymbol{\mu}_H, \mathbf{A}, \boldsymbol{\mu}_A, \mathbf{B}, \boldsymbol{\mu}_B, \boldsymbol{\Sigma}_x, \boldsymbol{\Sigma}_z, c\}.$$

The model has two hyper-parameters that is included in the set $\zeta = \{\lambda_u, \lambda_v\}$. In order to fit the latent variable models, the EM algorithm, which maximizes a lower bound for the marginal likelihood, provides a powerful solution [34]. However, an exact EM algorithm cannot be used to infer the model parameters due to the intractable posteriors of the latent variables for the categorical

likelihoods. Specifically, the complete data likelihood is given for user i and item j by following generative process as follows:

$$L_{ij} = p(\mathbf{u}_i)p(x_i|\mathbf{u}_i)p(\mathbf{y}_i|\mathbf{u}_i)p(\mathbf{v}_j)p(\mathbf{z}_j|\mathbf{v}_j)p(\mathbf{w}_j|\mathbf{v}_j)p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j). \quad (3.8)$$

The likelihood consists of the categorical likelihoods of $p(\mathbf{y}_i|\mathbf{u}_i)$ and $p(\mathbf{w}_j|\mathbf{v}_j)$ which make an exact inference intractable. Instead, motivated by the Bernstein-von Mises theorem [37], we use variational inference by restricting the posterior distributions only to Gaussian to make the lower bound tractable. The variational EM approach is used by defining the local quadratic bounds for categorical likelihoods where Bohning bound has been shown to provide a useful lower bound [72–74]. This bound is obtained by locally approximating the log-sum-exp (lse) function for the log-likelihood of multinomial and categorical distributions [75]. The approximation is performed around a point called the free variational parameter. The log likelihood of the categorical distribution of user side information after applying Bohning bound can be written as follows:

$$\begin{aligned} \log p(\mathbf{y}_i|\mathbf{u}_i) &= \log \frac{e^{\mathbf{y}_i^T \boldsymbol{\eta}_{C,i}}}{1 + \sum_{k=1}^{M_u} e^{\boldsymbol{\eta}_{C,i,k}}} \\ &= \mathbf{y}_i^T \boldsymbol{\eta}_{C,i} - \text{lse}(\boldsymbol{\eta}_{C,i}) \\ &\geq \mathbf{y}_i^T \boldsymbol{\eta}_{C,i} - \frac{1}{2} \boldsymbol{\eta}_{C,i}^T \mathbf{F}_u \boldsymbol{\eta}_{C,i} + \mathbf{g}_i^T \boldsymbol{\eta}_{C,i} - e_i \\ &\geq \mathbf{y}_i^T (\mathbf{H}\mathbf{u}_i + \boldsymbol{\mu}_H) - \frac{1}{2} (\mathbf{H}\mathbf{u}_i + \boldsymbol{\mu}_H)^T \mathbf{F}_u (\mathbf{H}\mathbf{u}_i + \boldsymbol{\mu}_H) \\ &\quad + \mathbf{g}_i^T (\mathbf{H}\mathbf{u}_i + \boldsymbol{\mu}_H) - e_i, \end{aligned} \quad (3.9)$$

where $\boldsymbol{\eta}_{C,i,k}$ are the elements of the vector $\boldsymbol{\eta}_{C,i}$, and the lse function is given by $\text{lse}(\boldsymbol{\eta}_{C,i}) = \log(1 + \sum_{k=1}^{M_u} e^{\boldsymbol{\eta}_{C,i,k}})$. When the quadratic bound approximation is used, the lower bound to the complete data log-likelihood also becomes quadratic which lets the posteriors be approximated as Gaussian distributions. This is a reasonable approximation for large number of features ($D_u + M_u + I$) since the conditions of Bernstein-Von Mises theorem are satisfied under the exponential family models

with a Gaussian prior [37]. The intermediate parameters that are used within the bound are given as follows [72]:

$$\mathbf{F}_u = \frac{1}{2}(\mathbf{I}_{M_u} - \frac{1}{M_u + 1} \mathbf{1}_{M_u} \mathbf{1}_{M_u}^T), \quad (3.10)$$

$$\mathbf{g}_i = \mathbf{F}_u \boldsymbol{\psi}_i - \mathcal{S}(\boldsymbol{\psi}_i), \quad (3.11)$$

$$\mathbf{e}_i = \frac{1}{2} \boldsymbol{\psi}_i^T \mathbf{F}_u \boldsymbol{\psi}_i - \mathcal{S}(\boldsymbol{\psi}_i)^T \boldsymbol{\psi}_i + \text{lse}(\boldsymbol{\psi}_i), \quad (3.12)$$

where $\boldsymbol{\psi}_i$ is the free variational parameter around which the lse function is approximated. At each iteration, this parameter is updated as well to change the local approximation point. Gaussian log-likelihoods for user i that appears in the complete data log-likelihood are given as follows:

$$\log p(\mathbf{u}_i) = -\frac{K}{2} \log(2\pi) - \frac{1}{2} \log |\lambda_u^{-1} \mathbf{I}_K| - \frac{\lambda_u}{2} \mathbf{u}_i^T \mathbf{u}_i, \quad (3.13)$$

$$\log p(r_{ij} | \mathbf{u}_i, \mathbf{v}_j) = -\frac{1}{2} \log(2\pi) - \frac{1}{2} \log c - \frac{c}{2} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j), \quad (3.14)$$

$$\log p(\mathbf{x}_i | \mathbf{u}_i) = -\frac{D_u}{2} \log(2\pi) - \frac{1}{2} \log |\boldsymbol{\Sigma}_x| - \frac{1}{2} (\mathbf{x}_i - \mathbf{W} \mathbf{u}_i - \boldsymbol{\mu}_W) \boldsymbol{\Sigma}_x^{-1} (\mathbf{x}_i - \mathbf{W} \mathbf{u}_i - \boldsymbol{\mu}_W). \quad (3.15)$$

The log-likelihoods for the items are similar due to the symmetry of the model and will not be replicated to avoid clutter. The lower bound for the complete data log-likelihood is found by the summation of the log-likelihoods of each factor in Eq.3.8. In the EM algorithm, taking the expectation of this bound with respect to the posterior distributions of the latent variables \mathbf{u}_i and \mathbf{v}_j first by using the old model parameter values and later by maximizing this expectation with respect to these parameters will yield a new parameter set [34].

In E-step, specifically, we first obtain the means and the variances of the Gaussian approximation for the posteriors of \mathbf{u}_i and \mathbf{v}_j . These can be derived by completing the square by collecting quadratic and linear terms in the log-likelihood to form Gaussian likelihoods [76]. It is important

to note that since a lower bound is used, the variational posterior distributions are obtained as $q(\mathbf{u}_i) = \mathcal{N}(\mathbf{u}_i | \mathbf{m}_{ui}, \boldsymbol{\Sigma}_{ui})$ for \mathbf{u}_i and $q(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j | \mathbf{m}_{vj}, \boldsymbol{\Sigma}_{vj})$ for \mathbf{v}_j instead of the exact posteriors. The E-step equations for the variational parameters \mathbf{m}_{ui} and $\boldsymbol{\Sigma}_{ui}$ of $q(\mathbf{u}_i)$ are given as follows:

$$\boldsymbol{\Sigma}_{ui} = (\lambda_u \mathbf{I}_K + \mathbf{H}^T \mathbf{F}_u \mathbf{H} + \mathbf{W}^T \boldsymbol{\Sigma}_x^{-1} \mathbf{W} + c(E[\mathbf{V} \mathbf{O}_i \mathbf{V}^T]))^{-1}, \quad (3.16)$$

$$\mathbf{m}_{ui} = E[\mathbf{u}_i] = \boldsymbol{\Sigma}_{ui}(c(E[\mathbf{V}] \mathbf{O}_i \mathbf{r}_i) + \mathbf{H}^T(\mathbf{y}_i + \mathbf{g}_i - \mathbf{F}_u \boldsymbol{\mu}_H) + \mathbf{W}^T \boldsymbol{\Sigma}_x^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_W)), \quad (3.17)$$

$$E[\mathbf{u}_i \mathbf{u}_i^T] = \boldsymbol{\Sigma}_{ui} + E[\mathbf{u}_i]E[\mathbf{u}_i^T], \quad (3.18)$$

where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_J]$ and $\mathbf{r}_i = [r_{i1}, \dots, r_{iJ}]^T$. \mathbf{O}_i is a $J \times J$ diagonal matrix whose entries are the binary indicators of the observed ratings of each item for user i to calculate the sufficient statistic by summing only the second moments of the items rated by user i . c is a global parameter that weighs this statistic to maintain a balance between the ratings and the side information. The first term in Eq.3.16 is the prior precision given for \mathbf{u}_i that prevents over-fitting. The second and the third terms correspond to the contributions of the categorical and real-valued side information respectively. Note the fact that the second term depends only on the global parameters \mathbf{H} and \mathbf{F}_u instead of the instances. The last term couples the user posterior covariance with the second moment of the items through a coefficient c . The optimal parameter c is estimated during the M-step by taking the sparsity of the dataset into account. After calculating the posterior covariance, the posterior mean is calculated by using Eq.3.17. The first term in this equation couples the mean of the item latent variables with the observed ratings. \mathbf{O}_i term effectively includes only those item variables that are rated by the user i . c is used to weigh this coupling term with respect to the second and third terms corresponding to the observations with the categorical and the Gaussian side information respectively. The sum of these terms is multiplied with the posterior covariance to calculate the posterior mean of the latent variable of the user i . Next, the second moment of each user is calculated since the coupling between the items and the users in the posterior covariance calculation for item latent variable \mathbf{v}_j appears through this statistic, as shown in Eq.3.16 for the item second moment. Finally, the variational free parameters for each user and item are updated in

the E-step for appropriate local approximation. Optimal update for a user which has been shown in [73, 75] is given as follows

$$\boldsymbol{\psi}_i = \mathbf{H}E[\mathbf{u}_i] + \boldsymbol{\mu}_H. \quad (3.19)$$

The equations for the item vector \boldsymbol{v}_j are similar (due to the symmetrical graphical model) with the same form but different parameters. The sum of second moments over all items and users is used as a sufficient statistic to evaluate the M-step.

In M-step, by using the predicted posterior distributions of \mathbf{u}_i and \boldsymbol{v}_j after each E-step, the model parameters are estimated point-wise to maximize the lower bound of the expected complete-data log-likelihood, which corresponds to the M-step of the EM algorithm. To find the update equations for the model parameters, the derivative of the expectation of complete-data log-likelihood with respect to each model parameter is evaluated. For the Gaussian modality of the user side, the update equations for the global parameters \mathbf{W} , $\boldsymbol{\mu}_W$ and $\boldsymbol{\Sigma}_x$ are obtained as follows.

$$\mathbf{W} = \left[\sum_i (\mathbf{x}_i - \boldsymbol{\mu}_W) E[\mathbf{u}_i]^T \right] \left[\sum_i E[\mathbf{u}_i \mathbf{u}_i^T] \right]^{-1}, \quad (3.20)$$

$$\boldsymbol{\mu}_W = \frac{1}{I} \sum_i \mathbf{x}_i, \quad (3.21)$$

$$\boldsymbol{\Sigma}_x = \text{diag} \left\{ \frac{1}{I} \sum_i (\mathbf{x}_i - \boldsymbol{\mu}_W)(\mathbf{x}_i - \boldsymbol{\mu}_W)^T - (\mathbf{x}_i - \boldsymbol{\mu}_W) E[\mathbf{u}_i]^T \mathbf{W}^T \right\}. \quad (3.22)$$

These are exactly the same update equations as in the factor analysis models [43]. For the categorical modality, following the same approach, the update equations are obtained for the global parameters \mathbf{H} and $\boldsymbol{\mu}_H$ as follows:

$$\mathbf{H} = \left[\sum_i (F_u^{-1}(\mathbf{y}_i + \mathbf{g}_i) - \boldsymbol{\mu}_H) E[\mathbf{u}_i]^T \right] \left[\sum_i E[\mathbf{u}_i \mathbf{u}_i^T] \right]^{-1}, \quad (3.23)$$

$$\boldsymbol{\mu}_H = \frac{1}{I} \sum_i \left\{ F_u^{-1}(\mathbf{y}_i + \mathbf{g}_i) - \mathbf{H}E[\mathbf{u}_i] \right\}. \quad (3.24)$$

Lastly, the precision parameter c is updated in the same way as follows,

$$c = \frac{1}{|\Omega|} \sum_{i,j \in \Omega} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2, \quad (3.25)$$

where Ω is the set of index pairs $\{i, j\}$ of the observed ratings and $|\Omega|$ is the cardinality of the set. The successive E and M-steps are performed until all the parameters converge to steady-state values.

Once the model is trained and the posterior distributions for all users and items are obtained, a user's score on an item is predicted for the purpose of making a personalized recommendation. A straightforward approach is to compute the inner products of the user and the item posterior means for the unobserved ratings as follows:

$$\hat{r}_{ij} = E[\mathbf{u}_i]^T E[\mathbf{v}_j] = \mathbf{m}_{ui}^T \mathbf{m}_{vj}. \quad (3.26)$$

3.3.3 Computational Complexity

In this section, we analyze each step of the algorithm in terms of their computational complexities to develop a general understanding of the scalability of the proposed approach. We start with the covariance computation in the E-step, given by Eq.3.16. The Gaussian modality term inside the summation requires multiplication of a $K \times D_u$ matrix with its transpose which results in $O(K^2 D_u)$. Note that multiplication with the diagonal term Σ_x^{-1} in Eq.3.16 has no additional cost. Similarly, the categorical modality term has a complexity of $O(K^2 M_u)$, where multiplication with F_u does not increase the complexity due to its special form given in Eq.3.10. The last term which couples the item second moments with the user posterior covariance requires multiplication of a $K \times J$ matrix with its transpose, resulting in $O(JK^2)$ computations. Hence, for each user, the overall computation of the posterior covariance matrix in Eq.3.16 requires $O(K^2(M_u + D_u + J + K))$. We can assume that for a typical recommender system J , (the number of items), is very large compared to D_u and M_u . Moreover, the number of latent dimensions K is also typically chosen much smaller

than I and J , hence the complexity reduces to $O(JK^2)$ for Eq.3.16. The computation of the posterior mean in Eq.3.17 includes only matrix-vector multiplications that require $O(KJ)$ computations. The computation of second moment in Eq.3.18 requires $O(K^2)$ computations. Finally, for all users, we repeat the computations in Eq.3.16-Eq.3.18, resulting in $O(IJK^2)$ asymptotic complexity. Due to the symmetry of the proposed model for categorical posterior, we similarly have a complexity of $O(IJK^2)$, thus the total cost of the E-step is $O(IJK^2)$.

The computation of \mathbf{W} in the M-step requires multiplication of two terms. The first one requires a summation over multiplication of $D_u \times 1$ and $1 \times K$ vectors which result in $O(IKD_u)$. The inversion in the second term requires $O(K^3)$ and the multiplication of the two terms requires $O(D_u K^2)$. Subsequently, the total cost is $O(IKD_u)$. Similarly, the computation of Σ_x has $O(IKD_u)$ complexity, and for \mathbf{H} the complexity is $O(IKM_u)$. Their joint complexity is $O(IK(D_u + M_u))$. Due to the symmetry, the item side parameters need $O(JK(D_v + M_v))$ computations, which makes the total cost of a single M step as $O(IK(D_u + M_u) + JK(D_v + M_v))$.

In terms of the overall computational complexity of a single EM iteration, the focus is on the number of users I and the number of items J as they will significantly outweigh any other system parameter in a typical large scale implementation. We see that while the complexity of the E-step scales by IJ , the M-step scales by $(I + J)$. Conclusively, one can say that for large I and J , the E-step will dominate the computational load with an overall model scaling factor of IJ . Since the complexity linearly scales with both the number of users and the number of items, the proposed algorithm is competitively scalable for big data applications in terms of time computational complexity.

3.4 Experimental Results

3.4.1 Evaluation Models

In this section, we briefly describe the models which are included for the purpose of comprehensive performance evaluation. For labeling purposes, the proposed model will be called

the MF-MSI method ⁴. The other models can be categorized into three groups. The first group includes some baseline MF algorithms that are related to the proposed algorithm ⁵. The second group consists of extensively used standard benchmark algorithms. The last group consists of the recent algorithms that can incorporate side information, which are selected from the categories defined in Section 3.2.

1. Baseline MF Algorithms

- **BPMF**: In this model, the side information is not incorporated. The user and item latent variables are inferred by using only the sparse rating matrix. In this baseline model, the second and third terms in both the summations of both Eq.3.16 for posterior covariance and Eq.3.17 for mean estimation are simply removed. Since there are no categorical likelihoods in this altered complete data likelihood, the exact EM solution is applied instead of the variational EM. This particular baseline model resembles the well-known approaches in the literature such as Bayesian Probabilistic Matrix Factorization (BPMF) [39] and Factor Analysis [43]. In fact, the model definition here is the same as BPMF but the inference is performed via EM instead of Markov Chain Monte Carlo (MCMC), hence, this algorithm will be labeled as BPMF in the comparative studies.
- **PMF**: A gradient-based optimization is used to find the point estimates of the parameters and the MAP estimates of the latent variables which maximize the complete data log-likelihood [38]. This method does not use the posterior covariances of latent variables which makes it more sensitive to the hyper-parameters [72]. PMF method is implemented by removing the side information terms in Eq.3.16.

2. Standard Algorithms

⁴The code is available at <https://github.com/maktukmak/MF-MSI>.

⁵Surprise package is used for the implementation of some algorithms in this category.

- NormalPredictor: The unknown ratings are predicted by sampling from a normal distribution whose mean and variances are estimated from the training data using the Maximum Likelihood Estimation.
- BaselineOnly: The global, the user and the item-specific means are evaluated by using the ratings in the training dataset. Predictions are performed by summing up the mean values [77].
- KNN Models: In the user-based neighborhood models, the similarity between users is computed by using the cosine distance and the predictions are evaluated by linearly weighting the predictions of the k-neighbors with the pre-computed similarity values. We considered four different types of KNN models. KNNBasic is performed with raw ratings. KNNWithMeans takes into account the mean of the user and the item ratings. KNNWithZScore incorporates the z-score normalization for each user. KNNBaseline uses the baseline ratings computed by summing up the global, the user and the item mean rating [78].
- SVD: This model factorizes the rating matrix into two low dimensional user and item matrices. The global mean, the user bias and the item bias are incorporated in the model. Stochastic gradient descent is used to optimize low dimensional matrices and biases. It is similar to the PMF model but also includes biases [1].
- NMF: This model is similar to SVD but the latent factors are forced to be positive. It is recommended for use particularly in datasets with only positive interactions [79, 80].
- SVDpp: This is an extension of SVD taking into account implicit feedback. A new set of item factors is introduced to capture implicit ratings. All the factors are optimized along with biases by using a gradient-based approach [78, 81].
- SlopeOne: Average differences between the ratings of the target item and the items rated by the other users are evaluated in a pairwise manner [82].

- CoClustering: K-means algorithm is used to form the clusters for the users and the items and the co-cluster for the ratings. The means of these three clusters are summed up to find the prediction of an unknown rating [83].

3. Recent State-of-the-art Algorithms

- LCE: A matrix factorization model which can incorporate side information via collective factorization. This model is similar to the proposed MF-MSI in terms of the common latent space utilization. However, the model can only incorporate real-valued side information. For mixed data types, the side information is pre-processed to form real-valued similarity matrices by using RBF kernels [57].
- DecRec: This model extracts a submatrix from the rating matrix, which is dense enough to be completed with low error rates. The completion is performed via classical Matrix Factorization by evaluating the latent factors. The cosine similarity matrices for the users and the items are computed by using the side information. Linear regression is performed to find the latent factors of the users and the items which are excluded by the sub-matrix by using similarity matrices and in-sub-matrix latent factors [68].
- KMF: Kernelized matrix factorization uses Gaussian process priors to regulate the columns of the latent matrices as opposed to the rows as in classical PMF models. The covariance matrices of the priors are simply assigned as similarity matrices formed by using the side information of the users and the items [51].
- LightFM: This model incorporates mixed data type side information by adding the features from the metadata of the users and the items to the classical matrix factorization model. This model is not probabilistic and gradient-based optimization is used to find the point estimates of the latent vectors [66].

3.4.2 Datasets

Four different datasets are used in the experimental study. The first one is a synthetic dataset generated, i) to analyze the model behavior with respect to varying levels of sparsity, ii) to assess the convergence property and iii) to observe the computational load and scalability with respect to the growing dataset. The comparative study includes three different variations of the MovieLens dataset with the different number of interactions (100K, 1M, and 10M) commonly used as the official benchmark datasets in recommender system applications to assess small to large scale performance. A specific advantage of MovieLens dataset is the availability of multi-modal side information for both the users and the items. The statistics of each dataset are summarized in Table 3.1.

- **Synthetic:** The synthetic dataset is generated by following the generative process described in section 3.3.1. The rating matrix is generated by randomly removing a fraction of the generated values. In addition to the sparse rating matrix, the fully observed multi-modal side information is generated for both the users and the items. The Gaussian modality dimensions are chosen as $D_u = 3$ and $D_v = 3$. Two categorical modalities are incorporated for each side such that $M_u = [5 \ 3]$ and $M_v = [5 \ 3]$ which means that the first and the second categorical features have 6 and 4 classes (where the last class used as the pivot) respectively. The dimension of the latent space is fixed as $K = 3$. λ_u , λ_v and c are fixed as 1. The number of the users/items (I , J) and the missing value fractions are varied according to the specifications of the experiments.
- **MovieLens 100K:** MovieLens 100K is one of the most popular small-scale recommender system datasets used for bench-marking. The rating matrix is generated by the interactions of 943 users with 1682 items. The number of interactions is 100K which makes the fraction of missing values 0.937. The dataset has fully observed user side information such as age, gender, occupation and zip code. We model the age information as a uni-variate Gaussian modality and gender and occupation information as categorical modalities with 2 and 21

Table 3.1: MovieLens dataset statistics

Statistics	MovieLens-100K	MovieLens-1M	MovieLens-10M
# Users	943	6040	71567
# Items	1682	3883	10681
# Ratings	100000	1000209	10000054
Rating Sparsity	0.937	0.957	0.987
Real-valued info for users	Age	Age	-
Categorical info for users	Gender, occupation	Gender, occupation	-
Real-valued info for items	Release	Release	Release
Categorical info for items	Genre	Genre	Genre

classes, respectively. For the item side, the dataset has movie title, release date, and genre. We model the release date as a uni-variate Gaussian. Genre is a 21-dimensional non-1-of-K binary indicator. Hence, the genre information is modeled as 21 different categorical modalities where each one can have two values (Bernoulli distribution).

- **MovieLens 1M:** Similarly, MovieLens 1M is generated by approximately one million interactions of 6040 users on 3883 items which makes the sparsity around 0.957. It has the same metadata as MovieLens 100K dataset for both the users and the items.
- **MovieLens 10M:** Finally, MovieLens 10M is a large-scale dataset that provides approximately 10M interactions of 71567 users on 10681 items. The fraction of missing values, in this case, is 0.987. Unlike the previous two datasets, the user side information is not officially provided. For the item side information, we similarly have the genre and release date like the previous two datasets.

3.4.3 Evaluation Metrics

To compare the performance of the algorithms described in Section 3.4.1, two performance evaluation metrics are used throughout the study. Mean Square Error(MSE) is used to assess the performance when explicit rating prediction is performed, which is evaluated as follows:

$$MSE = \frac{1}{|\Omega_{test}|} \sum_{i,j \in \Omega_{test}} (\hat{r}_{ij} - r_{ij})^2$$

where Ω_{test} is the set in which the indices of the test interactions are stored.

The second metric is “recall” that measures the ranking performance of the model. Generally, recall is a more practical assessment of a recommender system as it directly measures the recommendation performance. In the special case of movie datasets, the evaluation is given as follows:

$$Recall@L = \frac{\text{number of movies the user liked in the top } L \text{ recommendations}}{\text{total number of movies the user liked}}$$

where L is the number of recommended movies that are selected from the test set of the corresponding user. For the remainder of the section, recall is reported as the average of all the user’s individual recalls.

3.4.4 Splitting the Dataset for Training and Testing

We apply two different test scenarios called the warm-start and the cold-start. The warm-start scenario corresponds to the case where at least one interaction for all the items and the users appear in the training set such that at least some information in the rating matrix is present for all the test users and items. We use the following recipe to create this condition. If the number of interactions for an item is smaller than 5, then all of its interactions are included only in the training set and no interaction for that item exists in the test set. If the number of interactions for an item is larger than 5, a randomly selected 60% – 20% – 20% of these interactions are separated for training, validation, and testing respectively.

On the contrary, the cold-start scenario corresponds to the case where some items have all of their associated interactions appeared in the testing set with no associated interactions in the training set. In order to create this case, %20 of the items are randomly chosen as test items and all of their interactions are separated exclusively for the testing set. Similarly, another %20 of randomly chosen items is dedicated as the validation set where all of their interactions are also removed from the training set.

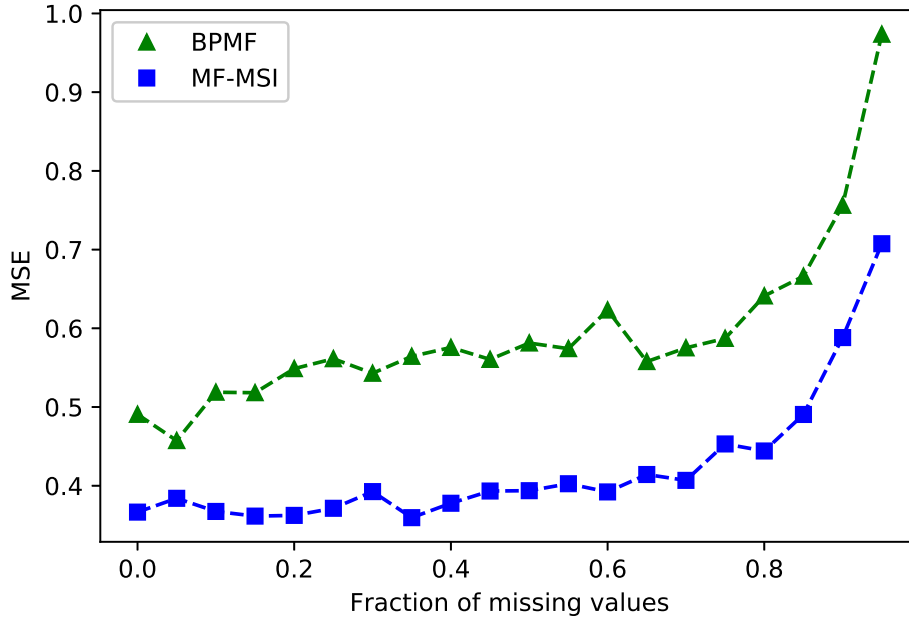


Figure 3.2: MSE vs missing value fraction on synthetic dataset

3.4.5 Simulation Study

First, we conduct a simulation in which a synthetic dataset is generated according to the configuration described in Section 3.4.2 with 300 users and 500 items. The purposes of this study are: i) to confirm the hypothesis that incorporating side information into the prediction process improves the performance of the recommender system, ii) to assess the performance sensitivity with respect to the sparsity level and iii) to observe the convergence behavior and iv) to test scalability of the proposed model. In this synthetic dataset, five modalities of the information exist including the Gaussian and categorical modalities for the user/item side information as well as the rating matrix. We induce the missing values and calculate the MSE performance on the fully observed rating matrix with respect to the varying fractions of missing values from %0 to %95 with increments of %5. In this section, the proposed model is only tested against the BPMF model which does not utilize any side information but is otherwise the same predictor. The results are presented in Figure 3.2.

We see a significant improvement in the performance of the MF-MSI model over the BPMF across all different fractions of missing values which serves as a strong empirical validation of

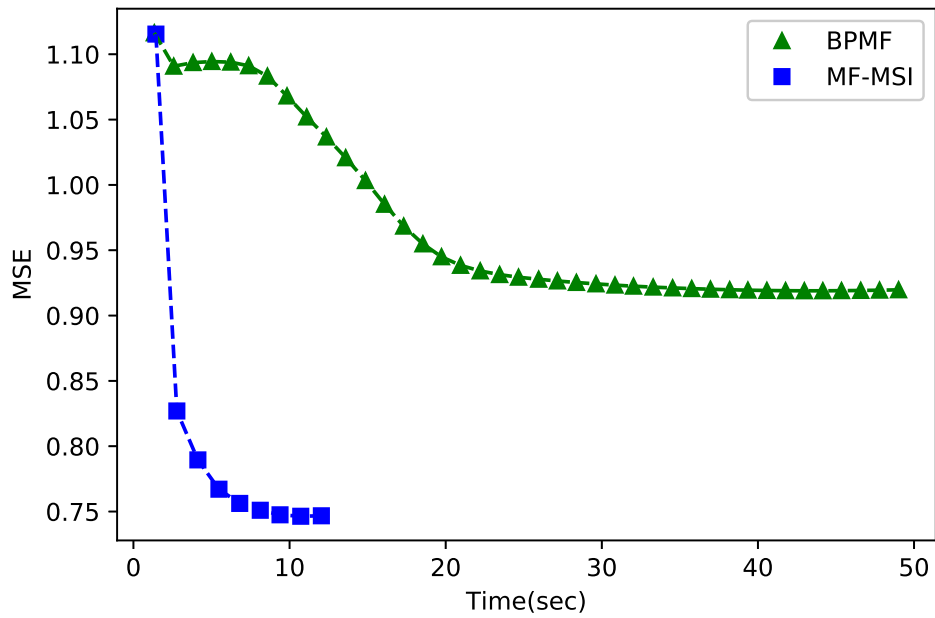
the model and its behavior. These results are averaged across 10 experiments for each fraction. Although the difference is around 0.13 when the missing value fraction is 0, i.e., all ratings are observed, the difference more than doubles to 0.27 when the missing value fraction is %0.95. This indicates that the MF-MSI model fits better to the dataset that follows a generative model assumption even when the fraction of missing values is very high with the help of observed side information. The high fraction of missing values case in this experiment is important and relevant since in most of the real world datasets, the rating matrices have high sparsity ($> \%95$) levels.

Next, we observe the convergence rate of the MF-MSI model compared to the BPMF model for the same hyper-parameter configuration. Figure 3.3.(a) shows the “MSE with respect to time in seconds” when the fraction of missing values is set as %95. The MF-MSI model tends to converge faster to a lower MSE. The BPMF, on the other hand, displays an elbow during the first few iterations which slows down the convergence. The addition of side information seems to remove this elbow and lead to a smoother and faster (as much as 4x times) convergence.

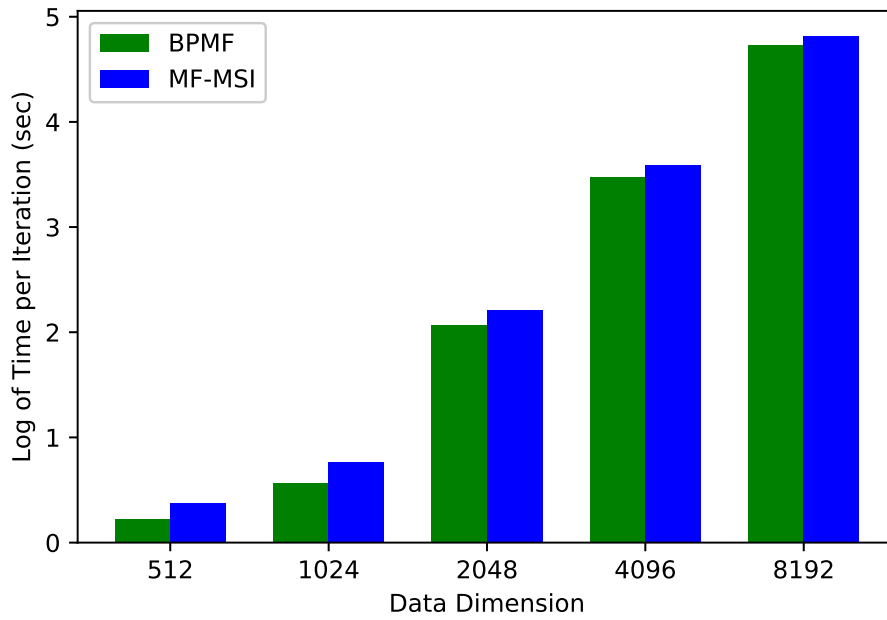
As discussed in Section 3.3.3, when the number of users I and the number of items J are large compared to the dimensions of the side information, time complexity reduces to a factor of $I \times J$. Figure 3.3.b shows the log of time per iteration when I and J increase from 512 to 8192. Although the total time increases exponentially for both models, the time difference between them vanishes as the numbers of users and items increase. As a result, one can conclude that adding the side information by using the proposed approach does not necessarily increase the time complexity per iteration for large scale applications while leading to an almost order of magnitude faster convergence to a more accurate point in a smaller number of iterations.

3.4.6 Movie Recommendation Study

In this section, we conduct a comprehensive experimental study to assess the performance of the proposed algorithm by using the MovieLens datasets. Firstly, we show the qualitative results by examining the latent space learned by the algorithm. Next, we compare the prediction and ranking performance of the algorithm with the baseline MF, the standard benchmark and the recent



(a) MSE convergence



(b) Log of time per iteration vs data dimension (number of users and items)

Figure 3.3: Convergence analysis on synthetic dataset

state-of-the-art algorithms, respectively. As a pre-processing, the real-valued side information for both the users and the items is normalized to have zero mean and unit variance. The categorical information is converted to 1-of-K binary representations. The last classes are designated as pivots such that M_u and M_v are equal to the number of the corresponding user and item classes minus 1. Additionally, the model hyper-parameters of each algorithm in the experimental study are optimized by using the validation set for each dataset. A grid search is performed and the parameter set that corresponds to the best performance in the validation set is used in the test set to report the evaluation results.

- Examining the Latent Space:

In this section, we analyze how well the proposed generative model fits the data by illustrating how the users and the items are grouped together in the latent space with respect to their statistical similarities. Since the posterior distributions of the user and the item latent variables are obtained after inference, the latent space can be discovered and explored by using KL divergence, which is a more convenient distance metric than the Euclidean distance due to the availability of posterior covariance. The closeness of two users can be assessed via KL divergence between the two multivariate Gaussian by using the following closed form expression:

$$D_{KL}(\mathbf{u}_i || \mathbf{u}_k) = \frac{1}{2} \left(\text{tr}(\boldsymbol{\Sigma}_{uk}^{-1} \boldsymbol{\Sigma}_{ui}) + (\mathbf{m}_{uk} - \mathbf{m}_{ui})^T \boldsymbol{\Sigma}_{uk}^{-1} (\mathbf{m}_{uk} - \mathbf{m}_{ui}) - K + \ln \left(\frac{|\boldsymbol{\Sigma}_{uk}|}{|\boldsymbol{\Sigma}_{ui}|} \right) \right),$$

where “tr” denotes the trace operator. Table 3.2 shows three sample users neighboring in the latent space. First, User1 is selected randomly among all users and then the two closest users are found using the KL divergence. The metadata of the corresponding users is provided in the table along with the names of the movies the users liked and their corresponding metadata. One can observe the similarity of the users’ demographic information such as gender (all female) and age (all middle age) and the fact that the occupations are not contrary. Similarly on the item side, one can see that some movies such as ‘Ice Storm’ and ‘The Postman’ appear

Table 3.2: Sample users from MovieLens 100K

User 1 (Female, 40, Librarian)		
Boogie Nights	1997	Drama
In and Out	1997	Comedy
Postman, The	1997	Drama
Mad City	1997	Action, Drama
Ice Storm	1997	Drama
User 2 (Female, 50, Other)		
Ice Storm	1997	Drama
As Good As It Gets	1997	Comedy, Drama
Wings of the Dove, The	1997	Drama, Romance, Thriller
Good Will Hunting	1997	Drama
Wag the Dog	1997	Comedy, Drama
User 3 (Female, 35, Administrator)		
Cold Comfort Farm	1995	Comedy
Postman, The	1997	Drama
Emma	1996	Drama, Romance
Sense and Sensibility	1995	Drama, Romance
George of the Jungle	1997	Children, Comedy

in each user’s list. The movie release dates are also close and the genres (such as comedy and romance) are overlapping. It is obvious that the model fits all the preferences from the rating matrix as well as the side information which includes the metadata of the users and the items in the latent space. Furthermore, we can use the similar approach to find movies that are close in the latent space. A sample of two different movie groups is listed in Table 3.3. The first movie in each list is chosen randomly and the KL divergence is used to find the 4 closest movies in the latent space for each group. Much like the user case, one can see how the side information including the genre and the movie release dates can help improve the clustering performance of mixed data type observations.

- Comparison with Baseline MF Algorithms:

For all the models in this category, the latent space dimension is chosen the same for a fair comparison, $K = 10$. Prior hyper-parameters λ_u and λ_v are chosen as 1. The rating precision c is initialized as 1. Figure 3.4.(a) shows the recall performances for both the warm and the cold settings. In the warm setting, as shown in the figure on the left, all models

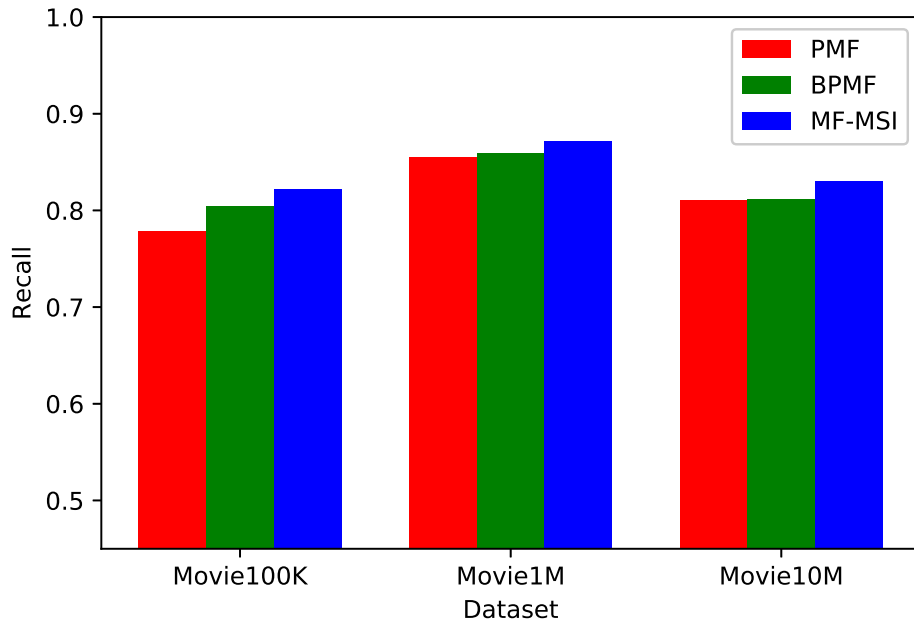
Table 3.3: Sample movie groups from MovieLens 100K

Group 1		
Shall We Dance?	1937	Comedy, Musical, Romance
Gay Divorcee, The	1934	Comedy, Musical, Romance
Top Hat	1935	Comedy, Musical, Romance
Women, The	1939	Comedy
Band Wagon, The	1953	Comedy, Musical
Group 2		
Ghost	1990	Comedy, Romance, Thriller
Pretty Woman	1990	Comedy, Romance
While You Were Sleeping	1995	Comedy, Romance
In the Line of Fire	1993	Action, Thriller
American President	1995	Comedy, Drama, Romance

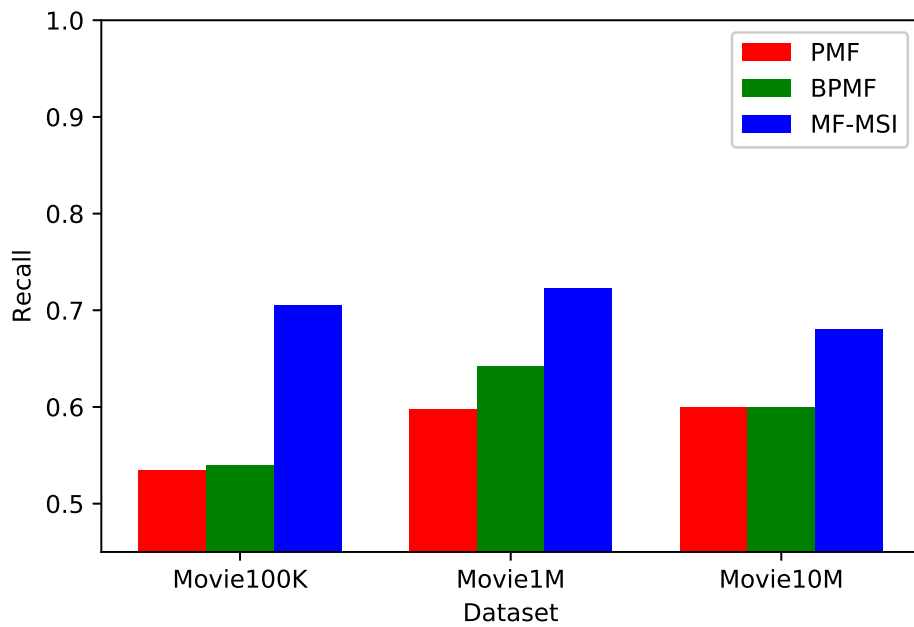
perform similarly (with slightly higher performance for the proposed algorithm) due to the cross-information being included in the rating matrix for both the training and the testing sets. As expected, the PMF model performs the worst in every scenario, while the proposed MF-MSI model performs slightly better than all the other models for all the three datasets.

The advantages of the proposed method become much clearer as shown in Figure 3.4.(b) when the more challenging and realistic cold setting environment is used. As expected, all three algorithms perform worse than the warm setting startup, however, the proposed model is significantly less affected. Naturally, the PMF and the BPMF models cannot generalize properly due to the initial lack of information in the rating matrix for the cold setting, which makes their performances depend highly on proper initialization.

Next, we analyze the effect of the number of recommended movies, L , on the recall performance. Figure 3.5 shows that the differences between the recall performances of the models are getting larger when the number of recommended movies (L) decreases. This behavior is even more apparent in the cold setting case suggesting that the side information allows the proposed model to have a significantly better recommendation performance specifically for its top recommendations. This represents the ideal case for a commercial recommender system as the majority of the users focus on the top 2-3 items in their recommended list where accuracy becomes more important.

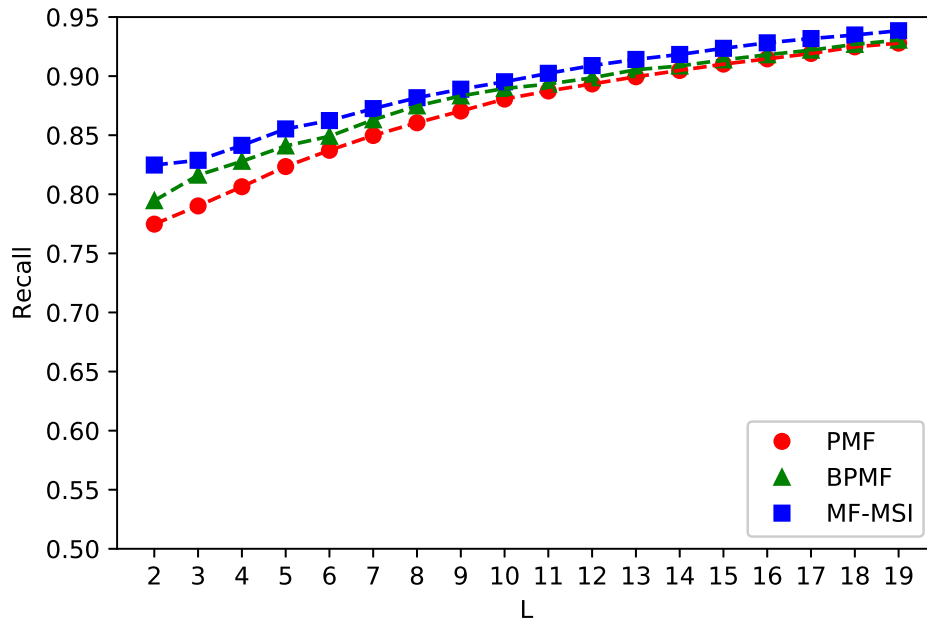


(a) Warm setting

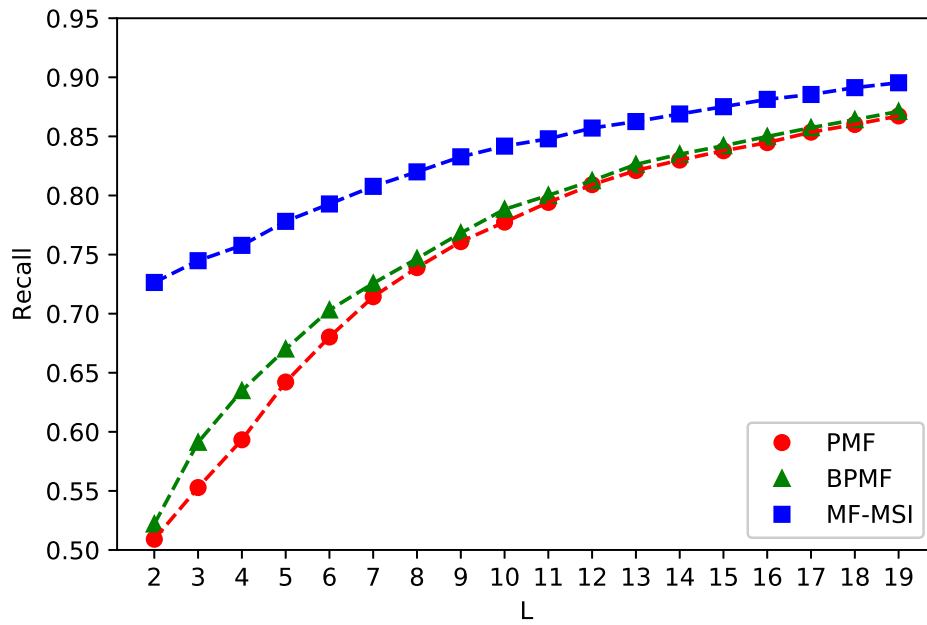


(b) Cold setting

Figure 3.4: Recall performances on MovieLens datasets when $L = 2$



(a) Warm setting



(b) Cold setting

Figure 3.5: Recall performances for the varying number of recommended movies

Table 3.4: MSE comparison with standard benchmark algorithms on the warm setting

Algorithms	MovieLens-100K	MovieLens-1M	MovieLens-10M
NormalPredictor	2.304	2.268	1.329
BaselineOnly	0.893	0.828	0.750
KNNBasic	0.964	0.872	*
KNNWithMeans	0.910	0.872	*
KNNWithZScore	0.910	0.874	*
KNNBaseline	0.934	0.901	*
NMF	0.872	0.848	0.766
SVD	0.895	0.794	0.664
SVDpp	0.863	0.767	0.659
SlopeOne	0.893	0.824	0.743
CoClustering	0.949	0.841	0.781
PMF	0.901	0.802	0.691
BPMF	0.850	0.776	0.671
MF-MSI	0.812	0.728	0.632

- Comparison with the Standard Benchmark Algorithms:

The standard algorithms are not capable of incorporating side information. To this end, the comparison is performed only for warm-start condition via MSE metric. Each experiment is repeated over 10 times to obtain a statistically meaningful result. At each trial, the training/testing split is performed randomly. As indicated in table 3.4, the proposed model outperforms the standard algorithms by exploiting the side information. The closest performance is achieved by SVDpp and BPMF. KNN models cannot produce results for the MovieLens 10M since the dataset is too large for these models to process*. Specifically, these models require similarity matrices to be computed and stored in the memory. In the case of MovieLens 10M dataset, the sizes of matrices are 71567×71567 and 10681×10681 for the user-based and item-based models respectively which requires a significantly large memory size for storage and processing. Our 32GB memory could not able to store the matrices. This scalability issue is a well-known bottleneck of neighborhood-based models [2, 76].

- Comparison with Recent State-of-the-art Algorithms:

The models in this category are able to incorporate the side information. We compare their performances with the proposed model under both the warm and the cold start conditions.

For recall performance, the number of recommended movies L is selected as 10. Table 3.5 presents the averaged results over 10 experiments with different random split initialization for each experiment. It is important to note that, as in the case of the KNN models, the LCE, DecRec and KMF algorithms cannot scale to the Movielens-10M dataset due to the space complexity of the models and their imposed memory limitations*. The LCE and DecRec algorithms require storing the similarity matrices calculated by using the side information of the users and the items, which results in $O(I^2 + J^2)$ space complexity. Additionally, the KMF algorithm requires inverting these matrices to obtain kernels for the Gaussian Process priors which results in $O(I^3 + J^3)$ computational complexity. LightFM and MF-MSI do not have these types of bottlenecks since they work on the raw features instead of the similarity matrices. LCE also needs a relatively large latent space dimension compared to MF-MSI. The reason is that the side information, which is the similarity matrix, is high dimensional. In order to project this matrix to the latent space, the dimension should be extended significantly. In [57], the authors suggest a latent space dimension of 500 while MF-MSI can achieve similar results with only a 20 dimensional space. That makes the proposed algorithm faster and more efficient compared to LCE. For instance, for the MovieLens 100K dataset, MF-MSI converges in 1.9s which is several orders of magnitude faster than LCE which requires 61.2s for reasonable performance.

In the warm-start scenario, MF-MSI performs better than all the other algorithms in terms of both RMSE and Recall. For the MovieLens 10M dataset, the proposed algorithm outperforms LightFM which is the only other state-of-the-art algorithm in this list scalable to the size of this dataset. In the cold-start scenario, MF-MSI outperforms the other algorithms in terms of ranking performance. However, KMF has better MSE performance for MovieLens 1M. Considering the scalability issue of KMF, one can conclude by the results reported in Table 3.5 that the overall performance of the proposed model is higher than the competitive algorithms in a significant majority of the test scenarios using the three differently sized datasets.

Table 3.5: Performance comparison with recent state-of-the-art algorithms

Warm-Start	MovieLens100K		MovieLens-1M		MovieLens-10M	
	<i>MSE</i>	<i>Recall</i>	<i>MSE</i>	<i>Recall</i>	<i>MSE</i>	<i>Recall</i>
LCE	0.854	0.890	0.729	0.901	*	*
DecRec	1.187	0.842	1.196	0.814	*	*
KMF	0.863	0.892	0.769	0.900	*	*
LightFM	1.021	0.889	1.034	0.893	1.827	0.886
MF-MSI	0.812	0.900	0.728	0.904	0.632	0.902
Cold-Start						
LCE	1.253	0.849	1.363	0.819	*	*
DecRec	1.234	0.854	1.208	0.823	*	*
KMF	1.208	0.821	1.140	0.791	*	*
LightFM	1.274	0.857	1.343	0.822	2.047	0.824
MF-MSI	1.192	0.861	1.261	0.827	1.490	0.886

3.4.7 A Discussion on Deep Learning Methods

In the experimental setup of MF-MSI algorithm, we excluded deep learning methods. The main reason is that there is a recent debate going on about suspicious performances of neural network models when applied to recommender systems. This claim has been validated with an elaborate analysis in a recent study [84]. We now follow the results presented in this study to show the ambiguity about neural network approaches. Some additional studies that point out similar problems can be found in [85–90].

Indeed, most of the recent proposed deep-learning based recommender system models have been found non-reproducible even the ones which have been published in good publication venues. [84] considered 18 algorithm published at top conferences that claim state of the art recommendation performance. Only 7 of them have been identified reproducible, i.e., there were publicly available codes, or authors could provide them upon request, and the codes could at least can be run without major modifications. Here, we briefly explain these reproducible deep learning algorithms.

Collaborative Memory Networks (CMN) algorithm [91] combines memory networks, attention mechanisms, neighborhood approaches, and latent factor models. Meta-path based Context for Recommendation (MCRRec) method [92] uses sampling to select higher quality paths, and

co-attention mechanism to represent path-based users, items, and contexts. The method also incorporates auxiliary information of items. Collaborative Deep Learning (CDL) [93] uses stacked denoising auto-encoders to learn a joint representation for collaborative information, i.e., latent representations of users and items as real-valued vectors. Neural Collaborative Filtering (NeuMF) [94] replaces inner products of vectors used in matrix factorization with non-linear neural architectures. Lastly, Spectral Collaborative Filtering (SpectralCF) [95] constructs bipartite graph from user-item relationship, then use a convolutional operation on spectral domain.

The authors compared these algorithms with some computationally and conceptually simple baseline algorithms. Here, we include three of them. The algorithm, Popular, recommends the most popular items in order to the users, so there is no personalization or learning. KNNUser algorithm uses the ratings of the users to compute a similarity measure between users, then recommends items to a target user by exploiting the preferences of her neighbor users. KNNItem algorithm, on the other hand, uses the ratings of the items to compute similarity measure between items. The results of the study are compiled in Table 3.6. Note that, we also used KNNUser algorithm to evaluate the performance of our proposed algorithm in 3.4.6.

It is clear from the results that these deep learning algorithms are not successful in top-k item recommendation tasks as much as they claim, as compared to the simple baseline algorithms. The study found out that authors of the papers mostly setup unfair test environments, and perform unfair tuning of baseline algorithms to promote their algorithms. So, there exists very huge ambiguity about these algorithms, which prevent researchers to include them as benchmarks.

Our intuition about why complex models fail or can not perform better proportional to their complexity, is that the recommender system datasets are highly sparse ($> 99.9\%$). The deep learning models contain high number of parameters to be optimized given the data. However, the number of observations are significantly low due to the sparsity. It is well know that deep learning models consume very large number of examples to be properly optimized. The recommender system tasks do not provide such environment. We suggest that well-defined statistical linear models are much more suitable in such conditions. Especially, the probabilistic models, into which

Table 3.6: The performance analysis of various deep learning algorithms

<i>Pinterest Dataset</i>			<i>MovieLens-1M Dataset</i>		
	HR@5	NDCG@5		REC@20	MAP@20
Popular	0.1668	0.1066	Popular	0.1853	0.0576
KNNUser	0.6886	0.4936	KNNUser	0.2881	0.1106
KNNItem	0.6966	0.4994	KNNItem	0.2819	0.1059
CMN	0.6872	0.4883	SpectralCF	0.1843	0.0539

<i>Pinterest Dataset</i>			<i>CiteULike-a Dataset</i>		
	HR@5	NDCG@5		REC@50	REC@100
Popular	0.1663	0.1065	Popular	0.0038	0.0073
KNNUser	0.7001	0.5033	KNNUser	0.0685	0.1028
KNNItem	0.7100	0.5092	KNNItem	0.0846	0.1213
NeuMF	0.7024	0.4983	CDL	0.0543	0.1035

<i>MovieLens-100K Dataset</i>		
	PREC@10	REC@10
Popular	0.1907	0.1180
KNNUser	0.2313	0.1802
KNNItem	0.3327	0.2199
MCRc	0.3077	0.2061

our proposed model falls, can well-define the problem, model the missing values using probability distributions by encoding uncertainty, give well performance with low computational complexity. Additionally, we also showed in Chapter 2, that our neural network-based matrix completion method performed worse as compared to probabilistic linear model such as VBPCA when sparsity level reaches high values, that also supports our intuition here.

3.5 Conclusion

In this chapter, we introduce a fundamentally different approach to incorporate multi-modal side information using a novel probabilistic generative framework for recommender systems. A scalable and computationally efficient statistical inference method based on variational EM is derived for datasets with very sparse interactions between the users and the items to exploit their associated multi-modal side information. The Bayesian structure of the model naturally enables the discovery of full multi-variate distributions over the latent space to provide a better prediction

performance in both the mean-square-error and recall metrics as more side information becomes available. The improvement in both the accuracy and the ranking performances of the proposed model is clearly demonstrated over a wide range of popular benchmark models for both warm and cold start test scenarios across both synthetic and real datasets. In fact, the state-of-the-art performance is achieved for the majority of the cases when compared to other recently introduced recommender systems which also incorporate the side information in different ways.

Our findings have several important implications when it comes to the next generation recommender systems. First of all, the fact that side information is utilized to improve the performance suggests that any additional knowledge acquisition by the companies both for the items they are promoting and their users would be beneficial regardless of the platform. Furthermore, the reduction in computational complexity to the level of scaling linearly with both the number of users and the number of items, would allow competitively scalable big data applications even when additional side information is rich and complex in nature.

3.6 Limitations and Future Research Directions

Ultimately, the main purpose of this study in this chapter is to serve as the proof-of-concept for the MF-MSI algorithm supported by both the fundamental derivations and empirical observations with a comprehensive experimental setup. Nonetheless, there are three separate promising pathways for further research. For instance, we demonstrate the scalability of the proposed model in terms of time complexity. While the memory complexity of the MF-MSI algorithm is better, especially when compared to the more recently proposed approaches, it still represents a challenge especially for extremely large datasets. Recent work on variational inference [96] provide promising solutions to such memory problems with stochastic optimization. As future work, stochastic variational EM can be introduced to the model to deal with the memory complexity issues for very large datasets to ultimately support online learning which is necessary for e-commerce applications. As additional future work, the model can incorporate recent techniques in the literature as discussed in Section 3.2 primarily to increase the prediction performance even further. For

instance, local factorization can be performed with a mixture model, or the dimensions of the latent space can be automatically inferred within a mixture rank model. Finally, the current structure of the multi-modal side information used in the experimental setup (such as age, occupation, title, year) is comparatively simple compared to the richer and more heterogeneous representations such as textual user reviews, movie synopsis. While the underlying model of inferring the multi-modal latent space is capable of expanding to more complex data representations, a significant portion of the future work can nonetheless focus on exploiting more diverse combinations of side information with the help of recent breakthrough in data fusion and language modeling such as contextual word embeddings.

Chapter 4: Sequential Attack Detection

4.1 Introduction

In most of the commercial platforms, the recommender systems play a key role in improving the user satisfaction by providing personalized experiences to ultimately increase sales and revenue. The recommender systems are mostly data-driven algorithms where the main source of information is the user feedback. To provide accurate recommendations for each user, the algorithms, even the modern ones currently used by popular services such as Netflix and Amazon, generally exploit collaborative filtering approaches [1] which are prone to manipulation [97]. Creating fake profiles and designing rating entries intelligently can adversely impact the recommendations for the genuine users. Reasons for such activities include promoting or nuking a specific item for which the attacker wants to either increase or decrease sales or popularity. Some attackers may only aim to disturb the system operation and reduce its efficiency for the genuine users [3].⁶

The collaborative filtering algorithms exploit the rating history of the users to extract the information of “closeness” between the users to recommend items by using the preferences of the neighboring users [1,99]. In a system, consisting of a large number of users and genuine ratings, one can try to infer the characteristics of genuine user behavior by processing the ratings and forming distinctive features to detect anomalies. Many supervised approaches try to classify the test users based on their rating profiles [100]. In this case, the assumption is the availability of the rating profiles of both the fake and the genuine users which is practically impossible in most applications. As a result, in the case of different types of attacks the system is not trained for, the performances of such algorithms degrade very quickly. To remedy this, we only consider the unsupervised attack detection in this study which assumes no a priori information about the attack type.

⁶Part of this chapter was published in [98]. Permission is included in Appendix A.

The main source of information for collaborative filtering is the rating history of the user profiles. Recently, researchers have begun looking into incorporating the side information, including the user attributes, item features, network structure, social friend/trust network, etc. to improve the recommendation quality [31,32]. Significant improvements in recommendation accuracy have been achieved by using this additional information in a statistical framework specifically for cold start scenarios [101]. The cold start scenario defines the setting when the users or items in the system do not have enough registered ratings or past interactions. Naturally, the cold start users and items have the potential to benefit dis-proportionally from side information sources. Surprisingly, up to now, the proposed attack detection algorithms considered solely the ratings and did not focus on the side information. In theory, the evidence of anomaly coming from the ratings can and should be aggregated with the evidence of anomaly coming from the side information entries. In this study, we consider for the first time in literature, the user attributes as an additional source of evidence for anomaly detection. This case is highly practical since accessing to the attributes of the genuine users registered in the system requires a much higher in-depth knowledge or sophisticated attack capabilities than accessing the ratings (which anyone with a computer and sufficient time could accomplish). In this study, we assume the attacker creates profiles by overlooking the compatibility between the ratings and the attributes of the genuine users. To this end, we assume that the attributes are randomly selected while the ratings are intelligently filled for each fake user. As a result, similar to the case of improving the recommendation quality for the cold start/user items, we now consider exploiting the side information to improve attack detection performance by detecting any mismatch.

The well-known anomaly detection algorithms require the exact knowledge of the probability density functions (pdf) of both the genuine and anomalous data [102, 103]. In recommender systems, it can be assumed that the genuine data is available to the operator to try and fit a probability distribution. However, the vectors associated with the rating profiles of the users are very high-dimensional and sparse. They also exhibit complex interactions with the vectors of the other users which makes it very hard or otherwise intractable to model such data by using high dimensional multivariate pdfs. The case is even worse for the fake profiles. There are different attack types

which means it is practically very difficult to model all kinds of anomalies with some parametric multivariate pdfs. In fact, one can safely assume that the anomalous pdf is totally unknown. To overcome these problems, we propose to extract uni-variate statistics given the high dimensional sparse rating vectors and the side information of both the users and the items. By using single dimensional statistics, we aim to lower the computational complexity of the detection algorithm for efficient implementation to time-sensitive online settings.

For efficient attack detection in recommender systems, the main challenges to obtaining useful uni-variate statistics are three fold: (i) the statistics should inherently include the compatibility between the ratings and the attributes, (ii) the statistics should be sufficiently informative to distinguish between the genuine and attack users, and (iii) the statistics should be easy to compute for quick evaluation in online settings. To meet these challenges, we first propose a linear latent variable model to embed the observed data of the genuine users including the sparse rating vectors and mixed data type user and item attributes. For flexibility, we consider two data types for the attributes: the real and categorical valued, since these data types are the most common to represent the attributes a user or an item can exhibit (such as gender and age for the users and year and genre for the items). The embeddings of the users and items lie in a low dimensional sub-manifold as a good representative of the genuine user behavior. Since the model is trained using both the attributes and the ratings, it exhibits a unified latent representation of the compatibility between the attributes and the ratings. The parameters of the model can then be used to measure such compatibility for the test users by producing uni-variate statistics which would deviate from the nominal values if the test user is an attacker. It is important to note that our proposed statistic is likelihood-based with a closed form expression for the trained model parameters which can be computed very efficiently in one-shot for the online settings.

Most of the attack detection frameworks developed for recommender systems focus on the sample-by-sample decision by ignoring the temporal relationship between the attack users. However, it is common to observe users who have different preferences which should be considered as non-persistent random outliers (trying a new movie genre) instead of actual anomalies. Single

outlier detection methods are vulnerable to false alarms in such scenarios [104]. On the other hand, if the system does observe persistent outliers in a short time interval, it may indicate a real attack. In this study⁷, we aim to detect persistent attacks by taking temporal relations of the test users into account and accumulating their anomaly statistics over time. This method resembles the well-known CUSUM sequential change detection algorithm that accumulates log-likelihood ratios (LLR) [102, 103] albeit that we instead use proposed uni-variate statistics. The sequential attack detection provides two advantages over the single attack profile detection: (i) lowering the false alarm rates by not labeling the non-persistent outlier users as attackers and (ii) increasing the true positive rates by accumulating the small evidences when the attack is hard to detect but still persistent.

In summary, we have the following contributions to the literature of attack detection in recommender systems:

- We design a probabilistic linear latent variable model that can represent the mixed data-type attributes of both the users and items, and the observed ratings. The parameters of the model define a latent space which inherently relates the attributes and the ratings of the genuine users.
- We propose uni-variate statistics to distinguish between the attackers and the genuine users. The statistics are easily computed in one-shot through the trained latent variable model parameters.
- We introduce the sequential attack detection problem in recommender systems and propose a CUSUM-like sequential detection algorithm by exploiting the uni-variate statistics generated under the genuine latent space.
- We demonstrate the performance improvements over both the single attack profile detection algorithms and the sequential detection algorithms that can only use the observed ratings by using a thorough experimental setup on three popular real world datasets.

⁷Part of this chapter was published in [98]. Permission is included in Appendix A.

4.2 Related Work

The literature includes many examples for attack types that can be performed on recommender systems [3, 99]. Subsequently, a range of attack detection algorithms have been introduced in the last two decades to mitigate such attacks [100]. We first discuss the popular attack types before briefly explaining the detection algorithms in the literature by categorizing them into static and temporal models. Note that we assume that the attackers could pass the authentication methods [105–107].

4.2.1 Attack Types

Researchers have developed a range of attack prototypes to demonstrate and study the effects of the attack profile injection on recommender systems and subsequently proposed the algorithms to detect and mitigate them. To form a realistic profile, the attackers do not only give ratings for the target items, but they select some filler items and fill them as well. The attack types basically differ from each other by the selection criteria of the filler items and the given ratings. An early simple attack type, called random attack, considered random selection and using the global mean of all the ratings for the chosen filler items in the system [108]. Average attack used item specific mean values instead of the global mean by exploiting more in-depth knowledge [109]. Bandwagon [108] and popular item [110] attacks were proposed to decrease the amount of knowledge required for the average attack but having similar efficiency by using the most popular (liked or rated) items as filler instead. Some attack types [111–113] require specific knowledge of the ratings of the users. While they can be used as upper performance bound (due to their complexity) they are also impractical to implement. The attack types specialized for nuking items were proposed in [114]. In this case, the filler items were selected among the most disliked items and were given low ratings along with the target item. To prevent the attack being detected, an obfuscation methodology was proposed in [115]. This method similarly uses the most popular items as filler by modifying the average attack to hide the malicious activity. [116] proposed a methodology to mislead the detection algorithms that use clustering. The attack efficiency can be traded off for better obfuscation if the attacker is

aware of the detection strategy [97]. A recent paper introduced adversarial algorithms to actually learn the detection strategy which increases the effectiveness of attack types across the board [117].

4.2.2 Static Detection Algorithms

In the attack detection algorithms developed for recommender systems, it is common to ignore the temporal relations between both the users and the ratings. In this scenario, the observed data consists of fully observed genuine and attack user ratings without any temporal dependency. The algorithms generally form distinctive features by using the ratings to distinguish the attack profiles from the genuine ones. By assuming similarity between the influential users [118] and the attack users, a set of features was proposed early in the field [119]. These features were extended to include the unusual number of ratings as an anomaly indicator in [120] and the sum of squared deviations of the rating from the mean values in [121]. In general, the features were then used in either unsupervised clustering (mostly k-means) [118–121] or supervised classification (kNN, SVM) [122, 123] for final decision. PCA algorithm was used in [124, 125] to classify highly correlated user profiles as potential attack users by exploiting the PCA of the user covariance matrix. A few components associated with the small eigenvalues were used as a feature for detection. Neyman-Pearson (NP) statistical test based on LLRs was proposed to identify attack types by exploiting the item selection preferences of the users [116]. A similar algorithm was based on the latent variable model by examining the entropy of the rating distributions in [126]. A probabilistic model based on Beta distribution was proposed in [127]. Graph-based algorithms [128–130] were also used to identify the most highly correlated groups as attackers. It is important to note that all the algorithms discussed in this subsection are examples of batch anomaly detection class of algorithms which require the entire data for evaluation with no direct trivial extension for online settings.

4.2.3 Temporal Detection Algorithms

To take into account the temporal relations between the ratings, [131] used the features associated with span, frequency and mount properties. Instead of hand-crafted features, [132] developed a hidden Markov model to learn the temporal rating behavior and identify the attack users with hierarchical clustering. In [133], a statistical method was proposed to detect significant changes in the mean and standard deviation of the ratings associated with each item. In the anomaly detection literature, many online algorithms have been proposed specifically for sequential detection [104]. SVM-based algorithms use kernel mapping to determine the decision region within the nominal data [134–136]. However, it is not always straightforward to choose and compute kernel functions for the defined problem. On the other hand, sliding window based Nearest Neighbor (NN) methods [137] generate a graph after each observation to compute a NN statistic, which makes them computationally costly. The quickest detection framework [102, 103] requires exact knowledge of the data distribution before and after the change. It uses LLRs to detect the change time as quickly as possible by controlling the false alarm rates, which makes it more suitable to the problem at hand if one can overcome the issue of inferring parametric distributions for the anomalous and genuine data. For a closely related application, the reputation systems, [138] proposed to use a two-sided CUSUM algorithm to detect the change in the rating distribution of each item by assuming Gaussian distributed ratings and observing only the mean parameter and not the variance. [139] improved this approach by assuming categorically distributed ratings and proposing a GLR algorithm by replacing the unknown attack distribution with the one estimated from the online data by using MLE.

4.3 Proposed Framework

In this section, we provide the details of our framework to successfully and quickly detect the sequential attacks in recommender systems. First, we define the latent variable model and present an optimization scheme based on variational EM algorithm. Then, we explain how to generate and exploit the uni-variate statistics from the trained model to detect persistent attacks.

4.3.1 Latent Variable Model

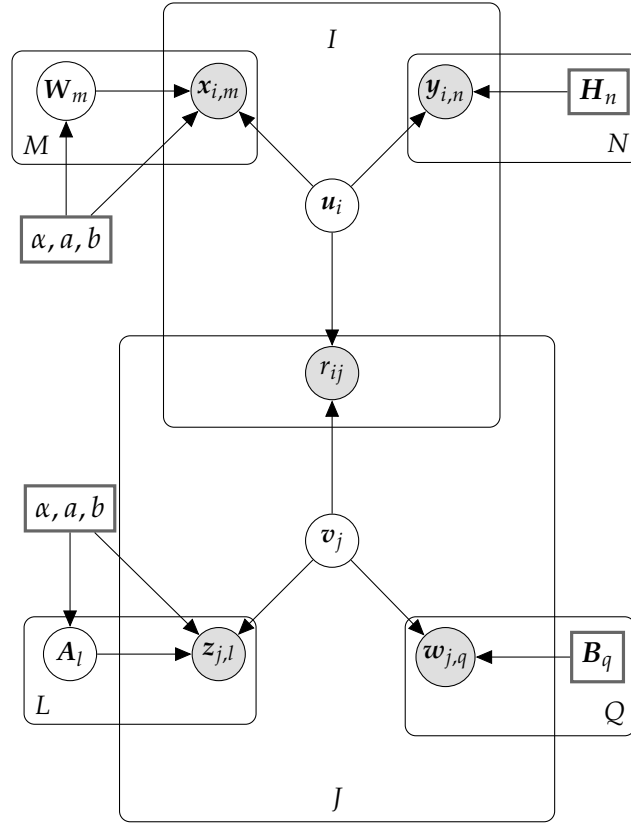


Figure 4.1: Graphical model representation of the linear latent variable model. The upper part is for the users, the lower part is for the items, and the intersection is for the ratings. $x_{i,m}$ and $z_{j,l}$ denote real-valued attributes while $y_{i,n}$ and $w_{j,q}$ denote categorical attributes.

We assume that the observed data consists of mixed data-type attributes for each user, mixed data-type attributes for each item, and real valued ratings. The users are indexed by i , where $i \in \{1, \dots, I\}$ and I is the total number of users. The items are indexed by j , where $j \in \{1, \dots, J\}$ and J is the total number of items. In order to represent this multi-modal data, we design a probabilistic linear latent variable model. The graphical demonstration of the model is shown in Fig.4.1. For each user and item, a latent variable is assigned to explain both the observed ratings and the associated attributes. The proposed model assumes Gaussian distribution for the latent variables [101]. Subsequently, zero mean spherical Gaussian priors are assigned as

$$p(\mathbf{u}_i) = \mathcal{N}(\mathbf{u}_i | \mathbf{0}_K, \lambda_u^{-1} \mathbf{I}_K), \quad (4.1)$$

where $\mathbf{u}_i \in \mathbb{R}^K$ represents the latent variable for user i and λ_u is the precision constant. The latent space is assumed to be K dimensional. We model the real-valued user attributes by using the Linear Gaussian Model:

$$p(\mathbf{x}_{i,m} | \mathbf{u}_i, \mathbf{W}_m, \boldsymbol{\Sigma}_{xm}) = \mathcal{N}(\mathbf{x}_{i,m} | \mathbf{W}_m \mathbf{u}_i, \boldsymbol{\Sigma}_{xm}). \quad (4.2)$$

Here, the mean of the m th real-valued observation $\mathbf{x}_{i,m} \in \mathbb{R}^{D_m}$ is linear function of the latent variable \mathbf{u}_i as in Factor Analysis models [43], where $m \in \{1, \dots, M\}$, M is the total number of real-valued user attributes, and D_m is the dimension of the observation. $\boldsymbol{\Sigma}_{xm} \in \mathbb{R}^{D_m \times D_m}$ is the observation noise covariance matrix. $\mathbf{W}_m \in \mathbb{R}^{D_m \times K}$ is the factor loading matrix. The classical FA models treat the latter two matrices as fixed and computes the MLE solution. However, MLE is prone to over-fitting, especially when the number of observations is small and/or there are missing observations [76]. We treat them as unknown and define random variables to regularize the problem by assigning conjugate priors. Since \mathbf{W}_m and $\boldsymbol{\Sigma}_{xm}$ are coupled in a non-factorized way in the likelihood, a natural full conjugate prior is in the form of Normal-inverse-gamma (NIG) distribution [76]:

$$p(\mathbf{W}_{m,d}, \boldsymbol{\Sigma}_{xm,dd}) = \mathcal{N}(\mathbf{W}_{m,d} | 0, \boldsymbol{\Sigma}_{xm,dd} \alpha^{-1} \mathbf{I}_{D_m}) \quad (4.3)$$

$$IG(\boldsymbol{\Sigma}_{xm,dd} | a, b),$$

where a and b corresponds to the shape and scale parameters of the distribution of each diagonal element d of $\boldsymbol{\Sigma}_{xm}$, respectively. We denote the dimension of the observation as D_m , where $d \in \{1, \dots, D_m\}$. Zero mean prior is assigned for the factor loading matrix \mathbf{W}_m . where α denotes how strong the belief for this prior is. We model the categorical-valued user attributes $\mathbf{y}_{i,n} \in \{1, \dots, M_n + 1\}$ by using a Linear Softmax Model:

$$p(\mathbf{y}_{i,n} | \mathbf{u}_i, \mathbf{H}_n) = \text{Cat}(\mathbf{y}_{i,n} | \mathcal{S}(\mathbf{H}_n \mathbf{u}_i)), \quad (4.4)$$

where $n \in \{1, \dots, N\}$ and N is the total number of categorical attributes. $\mathbf{H}_n \in \mathbb{R}^{M_n \times K}$ is the factor loading matrix for the categorical attribute n and M_n is the number of different categories with the pivot class excluded. Unlike W_m , this parameter is treated as fixed by following [72] since the softmax model is shown to be more prone to over-fitting when variational learning is performed. Note that the bias terms are removed in the aforementioned linear models to avoid cluttering. If the attribute variables are fully observed, the mean value is subtracted as a preprocessing to dropout the bias term. Otherwise, the optimization algorithm should take the bias terms into account by evaluating them at each step separately [140] or by absorbing them into the factor loading matrices [72]. In this framework, we assume that all the attributes are fully observed.

The last observation for the users is the rating history. Although many different models and distributions have been used to represent the ordinal ratings [38, 141, 142], we use dot product Gaussian conditional distribution for each observed rating since it is simple, reasonable and extensively used:

$$p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j) = \mathcal{N}(r_{ij}|\mathbf{u}_i^T \mathbf{v}_j, c^{-1}), \quad (4.5)$$

where c corresponds to the confidence parameter for the ratings. It can be treated as fixed [140] and optimized by using MLE or as hyper-parameter [38] and can be optimized via cross-validation. We use the former and optimize it during the training. The proposed model is symmetric for both the user and item sides which means the same type of models described until now are used for the item side. The prior for each item latent variable \mathbf{v}_j is given as:

$$p(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j|\mathbf{0}_K, \lambda_v^{-1} \mathbf{I}_K). \quad (4.6)$$

Same latent space dimension K is used to ensure coherence for dot-product rating modeling. Each real valued item attribute $z_{j,l} \in \mathbb{R}^{D_l}$, where $l \in \{1, \dots, L\}$ and L being the total number of real-valued attributes, is modeled by using a Gaussian conditional distribution as follows:

$$p(z_{j,l}|\mathbf{v}_j, \mathbf{A}_l, \boldsymbol{\Sigma}_{z_l}) = \mathcal{N}(z_{j,l}|\mathbf{A}_l \mathbf{v}_j, \boldsymbol{\Sigma}_{z_l}) \quad (4.7)$$

where $\Sigma_{zl} \in \mathbb{R}^{D_l \times D_l}$ is the noise covariance matrix and $A_l \in \mathbb{R}^{D_l \times K}$ is the factor loading matrix. Aforementioned NIG full conjugate prior is also used for these matrices:

$$p(A_{l,d}, \Sigma_{zl,dd}) = \mathcal{N}(A_{l,d} | 0, \Sigma_{zl,dd} \alpha^{-1} \mathbf{I}_K) \quad (4.8)$$

$$IG(\Sigma_{zl,dd} | a, b),$$

where we used same hyper-parameter set $\{\alpha, a, b\}$ as the user side. We denote the dimension of the observation as D_l , where $d \in \{1, \dots, D_l\}$. Lastly, the softmax model for the item categorical attribute $w_{j,q} \in \{1, \dots, M_q + 1\}$, where $q \in \{1, \dots, Q\}$, is defined as:

$$p(w_{j,q} | v_j, \mathbf{B}_q) = \text{Cat}(w_{j,q} | \mathcal{S}(\mathbf{B}_q v_j)), \quad (4.9)$$

with the corresponding factor loading matrix $\mathbf{B}_q \in \mathbb{R}^{M_q \times K}$, where $q \in \{1, \dots, Q\}$, Q is the total number of categorical-valued item attributes, and M_q is the number of categories without pivot class.

Combining the latent variables $\{\mathbf{u}_i, \mathbf{v}_j, \mathbf{W}_m, \mathbf{A}_l\}$, the fixed model parameters $\{\mathbf{H}_n, \mathbf{B}_q, c\}$, the hyper-parameters $\{\lambda, \alpha, a, b\}$, and the observed variables $\{r_{ij}, \mathbf{x}_{i,m}, \mathbf{y}_{i,n}, \mathbf{z}_{j,l}, \mathbf{w}_{j,q}\}$, we can write down the complete data log-likelihood for a user-item pair by following the chain rule as follows:

$$L_{ij} = p(\mathbf{u}_i) p(\mathbf{v}_j) p(r_{ij} | \mathbf{u}_i, \mathbf{v}_j)$$

$$\prod_{m=1}^M p(\mathbf{x}_{i,m} | \mathbf{u}_i, \mathbf{W}_m, \Sigma_{xm}) \prod_{d=1}^{D_m} p(\mathbf{W}_{m,d}, \Sigma_{xm,dd})$$

$$\prod_{l=1}^L p(\mathbf{z}_{j,l} | \mathbf{v}_j, \mathbf{A}_l, \Sigma_{zl}) \prod_{d=1}^{D_l} p(\mathbf{A}_{l,d}, \Sigma_{zl,dd}) \quad (4.10)$$

$$\prod_{n=1}^N p(\mathbf{y}_{i,n} | \mathbf{u}_i, \mathbf{H}_n) \prod_{q=1}^Q p(\mathbf{w}_{j,q} | \mathbf{v}_j, \mathbf{B}_q).$$

The optimization goal is to infer the latent variables and the model parameters to maximize the likelihood given the observed variables for all the user-item pairs in the data.

4.3.2 Model Optimization

During the model optimization we are interested in retaining the uncertainties for the user/item latent variables since they are both the roots of the graph and local variables. For this purpose, among the latent variables, the posterior inference is performed only for $\{\mathbf{u}_i, \mathbf{v}_j\}$. We perform MAP estimation for the other latent variables $\{\mathbf{W}_m, \mathbf{A}_l\}$. We seek MLE solution for the fixed model parameters $\{\mathbf{H}_n, \mathbf{B}_q, c\}$ and cross-validation is performed for the hyper-parameters $\{\lambda, \alpha, a, b\}$. EM algorithm is powerful tool for optimizing models like these with posterior inference and fixed point estimation (MLE/MAP), [76]. In the E-step, the posterior distributions of the latent variables are inferred and the sufficient statistics are accumulated. In the M-step, MLE/MAP estimations are computed from the sufficient statistics obtained in the E-step. Note that, in this section, we only show the equations for the user part to avoid cluttering. Due to the symmetry of the model, one can easily obtain the equations for the item part by changing the user variables in the equations with the appropriate item variables.

The joint probability of the model includes two different distribution factors, Gaussian and Categorical. Since exact inference is not possible, it is common to use approximations to perform variational inference. The user/item latent variables have Gaussian priors and the most of the features, ratings (J) and real-valued attributes (M), are modeled with Gaussian conditional distributions. Thus, it is reasonable to seek Gaussian posteriors due to the conditions of Bernstein-von Mises theorem [37] since the number of Gaussian variables is much larger than the number of categorical variables ($J + M \gg N$). To perform Gaussian approximation, the categorical likelihoods are approximated with a quadratic lower bound. In this study, we employ the Bohning bound [143], which has been used in [72] to provide a lower bound for the LogSumExp (LSE) function over the categorical likelihoods. After the bound is applied, the log likelihood of the n th categorical valued user attribute takes the following form:

$$\begin{aligned} \log p(\mathbf{y}_{i,n}|\mathbf{u}_i) &\geq \mathbf{y}_{i,n}^T \mathbf{H}_n \mathbf{u}_i - \frac{1}{2} \mathbf{u}_i^T \mathbf{H}_n^T \mathbf{F}_{u,n} \mathbf{H}_n \mathbf{u}_i \\ &\quad + \mathbf{g}_{i,n}^T \mathbf{H}_n \mathbf{u}_i - \mathbf{e}_{i,n}. \end{aligned} \quad (4.11)$$

There are three intermediate parameters $\{\mathbf{F}_{u,n}, \mathbf{g}_{i,n}, \mathbf{e}_{i,n}\}$ emerging due to the bound:

$$\mathbf{F}_{u,n} = \frac{1}{2} (\mathbf{I}_{M_n} - \frac{1}{M_n + 1} \mathbf{1}_{M_n} \mathbf{1}_{M_n}^T), \quad (4.12)$$

$$\mathbf{g}_{i,n} = \mathbf{F}_{u,n} \boldsymbol{\psi}_{i,n} - \mathcal{S}(\boldsymbol{\psi}_{i,n}), \quad (4.13)$$

$$\mathbf{e}_{i,n} = \frac{1}{2} \boldsymbol{\psi}_{i,n}^T \mathbf{F}_{u,n} \boldsymbol{\psi}_{i,n} - \mathcal{S}(\boldsymbol{\psi}_{i,n})^T \boldsymbol{\psi}_{i,n} + \text{lse}(\boldsymbol{\psi}_{i,n}), \quad (4.14)$$

where $\mathbf{F}_{u,n}$ depends only on the number of classes. This specific form allows the bound to have a fixed curvature [72]. $\mathbf{g}_{i,n}$ and $\mathbf{e}_{i,n}$ depend on the free variational parameter $\boldsymbol{\psi}_{i,n}$. This parameter is subsequently optimized during training for each data point to form a tight bound.

In the E-step of the variational EM algorithm, we seek posteriors of the user and item latent variables $\{\mathbf{u}_i, \mathbf{v}_j\}$ and compute the sufficient statistics for the M-step. The variational distributions are Gaussian requiring the computation of the mean vector and the covariance matrix for each user given the rating history and the associated attributes. We use an alternating scheme to update the user and item posteriors since they are coupled in the complete data likelihood. In particular, to compute the user latent variables, the item latent variables are held fixed alongside the observed variables, i.e:

$$q(\mathbf{u}_i | \mathbf{x}_i, \mathbf{y}_i, \mathbf{r}_i, \mathbf{V}) = \mathcal{N}(\mathbf{u}_i | \mathbf{m}_{ui}, \boldsymbol{\Sigma}_{ui}) \quad (4.15)$$

where \mathbf{V} is the matrix of item latent variables ordered as $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_J]$. After updating the user latent variables, the item latent variables are updated in similar fashion by fixing the user latent variables. To sum up, two alternate E-steps are performed consecutively for the users and the items. Update equations are derived by following the procedure for linear Gaussian systems [76] (See Appendix C for details of the inference). For covariance matrix of each user, the quadratic terms

in the complete data log-likelihood that depend on \mathbf{u}_i are collected as sum terms and the matrix inversion is applied:

$$\begin{aligned} \Sigma_{ui} = & (\lambda_u \mathbf{I}_K + \sum_{n=1}^N \mathbf{H}_n^T \mathbf{F}_{u,n} \mathbf{H}_n + \sum_{m=1}^M \mathbf{W}_m^T \Sigma_{xm}^{-1} \mathbf{W}_m \\ & + c(E[\mathbf{V}\mathbf{O}_i\mathbf{V}^T]))^{-1} \end{aligned} \quad (4.16)$$

For the mean vectors, the linear terms in the complete data log-likelihood which depend on \mathbf{u}_i are collected as sum terms and multiplied by the covariance matrix:

$$\begin{aligned} \mathbf{m}_{ui} = & \Sigma_{ui} (c(E[\mathbf{V}]\mathbf{O}_i\mathbf{r}_i) + \sum_{n=1}^N \mathbf{H}_n^T (\mathbf{y}_{i,n} + \mathbf{g}_{i,n}) \\ & + \sum_{m=1}^M \mathbf{W}_m^T \Sigma_{xm}^{-1} \mathbf{x}_{i,m}) \end{aligned} \quad (4.17)$$

As indicated above, the update equations for the item latent variables have similar forms except the parameter, latent variable and indices names. Lastly, free variational parameter $\boldsymbol{\psi}_{i,n}$ is updated until convergence for each user and item as:

$$\boldsymbol{\psi}_{i,n} = \mathbf{H}_n \mathbf{m}_{ui} \quad (4.18)$$

In the M-step, we compute MAP estimation of the latent variables $\{\mathbf{W}_m, \mathbf{A}_l\}$ and MLE of the model parameters $\{\mathbf{H}_n, \mathbf{B}_q, c\}$. By setting the gradient of the complete data log-likelihood with respect to \mathbf{W}_m to zero, we obtain the MAP estimation:

$$\mathbf{W}_m = \left[\sum_i \mathbf{x}_{i,m} E[\mathbf{u}_i]^T \right] \left[\alpha \mathbf{I}_K + \sum_i E[\mathbf{u}_i \mathbf{u}_i^T] \right]^{-1} \quad (4.19)$$

Here, the sum over the second moments of \mathbf{u}_i and the sum over the dot product of the real valued user attribute and the mean vector are the expected sufficient statistics calculated in advance in the E-step. α is the regularization constant coming from the NIG prior. For the factor loading matrix

\mathbf{H}_n of the categorical attributes, the form is slightly different due to the MLE and applied quadratic bound:

$$\mathbf{H}_n = \left[\sum_i F_{u,n}^{-1}(\mathbf{y}_{i,n} + \mathbf{g}_{i,n}) E[\mathbf{u}_i]^T \right] \left[\sum_i E[\mathbf{u}_i \mathbf{u}_i^T] \right]^{-1} \quad (4.20)$$

Although the sum over the second moment statistic is the same with the real-valued side, the other statistic changes due to the lower bound. The categorical observations $\mathbf{y}_{i,n}$ are linearly transformed to the real-valued pseudo observations through the intermediate parameters of the bound. MAP estimation of the noise covariance matrix of the real-valued user attributes is given as:

$$\begin{aligned} \Sigma_{xm} = \text{diag} \left\{ \frac{1}{I + 2(a + 1)} [2b + \sum_k \mathbf{W}_{m,k}^2 \alpha + \sum_i \mathbf{x}_{i,m} \mathbf{x}_{i,m}^T \right. \\ \left. - 2\mathbf{x}_{i,m} E[\mathbf{u}_i]^T \mathbf{W}_m^T + \mathbf{W}_m E[\mathbf{u}_i \mathbf{u}_i^T] \mathbf{W}_m^T] \right\} \end{aligned} \quad (4.21)$$

where \mathbf{W}_m is updated in advance as in FA models [43]. Lastly, the confidence parameter for the ratings is updated as follows:

$$c = \frac{1}{|\Omega|} \sum_{i,j \in \Omega} E[(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2] \quad (4.22)$$

The updates are iterated in the specific order (E-step for the users \rightarrow E-step for the items \rightarrow M-step) until the model likelihood or the model variables converge. Collecting the model variables and the latent variables excluding the local latent variables $\{\mathbf{u}_i, \mathbf{v}_j\}$ in a global latent space parameter set, i.e. $\Theta = \{\mathbf{W}_m, \mathbf{A}_l, \mathbf{H}_n, \mathbf{B}_q, c\}$, we obtain a genuine latent space that can be used in the proposed attack detection framework. The test users are projected to this space through the set Θ to produce the statistics to be used in detecting genuine versus attack users.

4.3.3 Attack Detection

Aforementioned latent variable model provides a unified representation for the users and the items in the training set. Resulting latent space consists of real-valued latent variables and

model parameters that represent both the observed ratings and the attributes. In other words, we implicitly obtain the compatibility between the ratings and the attributes. Assuming the users in the training set are completely genuine, we can exploit this compatibility through the latent space parameter set Θ to detect anomalies for the test users registered to the system.

Regarding to the problem of the online sequential attack detection in recommender systems, the most related framework is the quickest attack detection framework [102, 103], which is based on log-likelihood ratio (LLR):

$$s_t = \ln \frac{p_{\Theta_1}(y_t)}{p_{\Theta_0}(y_t)} \quad (4.23)$$

where p_{Θ_0} is the distribution of the data before the change (attack) and p_{Θ_1} is the distribution of the data after the change. Here, the change in scalar parameter Θ_0 is considered by the observing data point y_t at each time step t . The main motivation is to detect the change with minimum possible delay and minimum false alarm rate, which is formulated as a mini-max problem by Lorden [144]. To this end, since LLR shows a negative drift before the change and a positive drift after the change, a CUSUM algorithm is formulated as follows:

$$g_t = \max(0, g_{t-1} + s_t), \quad (4.24)$$

and the stopping rule and the alarm time are given as:

$$t_a = \min(t : g_t \geq h). \quad (4.25)$$

CUSUM algorithm is optimal for sequential change detection if the parameters of the distributions both before and after the change are known, and the alarm time is unknown [102]. However, in the problem that we consider, there are three unknown independent parameters which require modifications to the CUSUM algorithm for successful operation:

- *The parameter of the distribution before the attack:* The latent variable model defined in Section 4.3.1 provides one multivariate Gaussian distribution for each genuine user registered

in the system. However, it is difficult to estimate a single parametric distribution for these moderate dimensional multivariate data and seek parameter change by plugging the parameter into the LLR.

- *The parameter of the distribution after the attack:* Although it is possible to infer a distribution for the genuine users, it is very hard for the attack users since their distribution can change with the attack type. In Generalized Likelihood Ratio (GLR) test, this distribution is replaced by its MLE. However, similar to the case of the parameter before the change, it is hard to estimate a single parametric distribution after the change.
- *The attack time:* It is almost impossible to have information about the time when the attack begins in recommender systems. Since there is no a priori information about the distribution of the attack time, we cannot resort to Bayesian approaches. To this end, attack time is considered as a nonrandom unknown value as in the CUSUM algorithm.

Hence, we modify the CUSUM algorithm by replacing the LLR with a proposed univariate statistic. This results in a “CUSUM-like” algorithm as the framework for our sequential attack detection algorithm.

We propose to extract the univariate statistics for the test users by exploiting the trained model. This statistic should characterize the compatibility between the ratings and the attributes. It should be informative to distinguish between the genuine user profiles and the attack user profiles and easily computed during the online phase. We assume the registration process starts with requiring the users enter their attributes. The users can then assign ratings for different items in the system. Subsequently, we evaluate the rating likelihoods given the user attributes. To this end, we first need to infer the posterior distributions of the user latent variables given the attributes and the trained model parameters. Specifically, for user t , it corresponds to:

$$\mathbf{u}_t \sim p(\mathbf{u}_t | \mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,M}, \mathbf{y}_{t,1}, \dots, \mathbf{y}_{t,N}, \Theta, q(\mathbf{V})). \quad (4.26)$$

Note that, in the off-line phase of the framework, the user t corresponds to one of the genuine users on which the model trained. In the online phase, it corresponds to one of the test users. The posterior distribution is conditioned on the observed user attributes, the latent space global parameter set Θ , and the variational distribution of the item latent variables $q(\mathbf{V})$. The item latent variables are inferred during EM training and kept fixed afterwards. We need to integrate over $q(\mathbf{V})$ to obtain the posteriors of u_t . Fortunately, it is easy to compute since as a closed form expression, which corresponds to one E-step for the user t as formulated in Eq. 4.16 and 4.17. After the inference, the rating likelihood is computed through the dot-product modeling as follows:

$$p(r_{tj}|\mathbf{u}_t, \mathbf{v}_j) = \mathcal{N}(r_{tj}|\mathbf{u}_t^T \mathbf{v}_j, c^{-1}). \quad (4.27)$$

Note that \mathbf{u}_t and \mathbf{v}_j are random variables with their inferred second moments as uncertainties. One should take the uncertainties into account instead of using only the dot product of the mean values for this evaluation to obtain formal and more informative likelihoods. One alternative is to use Monte Carlo sampling to generate a large number of samples for \mathbf{u}_t and \mathbf{v}_j , to then compute the expected squared error. However, this process is relatively slow. Fortunately, since the latent variables are Gaussian, there is a closed form expression for this expectation, which is given in the Appendix B to avoid clutter.

In this framework, we propose two different approaches for the attack detection based on the users and the items.

- **User-based Approach:** In this approach, a uni-variate statistic is produced for each test user and the sequential attack detection is performed based on the test user indices. Specifically, d_t denote the anomaly score of user t . If we assume the distribution of the score as $p_{\Theta_0}^d$ for the genuine user and $p_{\Theta_1}^d$ for the attack user, we expect the following change in the distribution when the attack begins at time step τ :

$$d_t \sim \begin{cases} p_{\Theta_0}^d, & \text{if } t < \tau \\ p_{\Theta_1}^d, & \text{if } t \geq \tau \end{cases} \quad (4.28)$$

where $p_{\Theta_0}^d$ and $p_{\Theta_1}^d$ are unknown. To this end, we propose to use the precomputed likelihoods for the score d_t as follows:

$$d_t = \frac{1}{|\Omega_t|} \sum_{j \in \Omega_t} -\log p(r_{tj} | \mathbf{u}_t, \mathbf{v}_j). \quad (4.29)$$

where Ω_t is the set of items the user t has rated. We use average negative log-likelihood over the observed ratings of user t . This score is expected to produce high values for the attack users and low values for the genuine users. Since the user latent variables are conditioned on the user attributes, the rating likelihoods will have high magnitudes whenever the attributes of a test user are not compatible with his ratings profile based on the genuine latent space parameters. This property of the score satisfies the necessary conditions as a formal and informative metric to distinguish between the attack and the genuine users by exploiting the compatibility between the attributes and the ratings. The score can also be easily computed as it simply requires averaging over the log-likelihoods. The log-likelihoods themselves have low computational cost due to simple algebraic operations as provided in the Appendix ??.

In the offline phase of the algorithm, we first evaluate d_t for each genuine user in the system and store their anomaly scores. Then, we evaluate the tail probability (p-value) of the test users since the anomalies in fact correspond to right tail events based on the distribution of the nominal scores. The p-value is evaluated as follows:

$$p_t = \int_{d_t}^{\infty} p_{\Theta_0}^d(z) dz = 1 - F_{\Theta_0}^d(d_t) \quad (4.30)$$

where $F_{\Theta_0}^d(d_t)$ is the cumulative distribution function (CDF) of d_t . Since we don't know CDF of d_t , we estimate its empirical distribution function (EDF) by using the scores we stored in the offline phase. The EDF of F_0^d is estimated as follows:

$$F_{\Theta_0}^d(z) = \frac{1}{I} \sum_{i=1}^I 1_{\{d_i \leq z\}} \quad (4.31)$$

In the online phase, for each test user t , the corresponding p-value is given as:

$$p_t = \frac{1}{I} \sum_{i=1}^I 1_{\{d_i > d_t\}} \quad (4.32)$$

which is simply the fraction of the genuine users whose scores are higher than d_t . Ideally, the uni-variate statistic should generate negative values for the ratings belonging to a genuine user and positive values for the ratings belonging to an attack user to be compatible with LLR. By exploiting the p-value approach, we evaluate the corresponding statistic for the test user t as follows:

$$s_t = \log \frac{\alpha}{p_t}, \quad (4.33)$$

where α is the significance level.

- **Item-based Approach:** In this approach, the ratings of the test users entered for a specific item are observed. The detection task is performed independently for each item in the system. If an anomaly is detected for any of the items, a system-wide alarm is set. In this case, instead of averaging over the items a user has rated, the anomaly score is computed for each entry as follows:

$$d_{ij} = -\log p(r_{ij} | \mathbf{u}_t, \mathbf{v}_j), \quad (4.34)$$

where d_{tj} is the simply negative log-likelihood of the rating user t entered for item j . Contrary to the user-based approach, we only consider the ratings of the users who rated item j to evaluate the p-value as:

$$p_{tj} = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} 1_{\{d_{ij} \geq d_{tj}\}} \quad (4.35)$$

where Ω_j is the set of the genuine users who rated item j assuming the test user t rated the item j . Next, the uni-variate statistic is evaluated with α significance level as follows:

$$s_{tj} = \log \frac{\alpha}{p_{tj}}, \quad (4.36)$$

Please note that there is a statistic for each item. The time index t is based on the test user index who has only rated the item j . In the user-based approach, the time index is for all the test users regardless of the item indices.

One can decide if an entry is anomalous or not by observing only the uni-variate statistics s_t or s_{tj} , which is representative of the single-instance outlier detection technique. However, this technique is prone to false alarms due to the existence of non-persistent anomalies including some genuine users trying and rating new items in unexpected ways. To avoid the false alarms for the non-persistent anomalies, we propose a CUSUM-like sequential attack detection framework to use the uni-variate statistics to detect only the persistent attacks. The aim is to detect the attacks quickly by limiting the false alarm rates while accumulating the statistical evidence of the anomaly. In this algorithm, we plug the computed uni-variate statistics instead of LLR in the CUSUM algorithm as follows:

$$g_t = \max\{g_{t-1} + s_t, 0\}, \quad g_0 = 0, \quad (4.37)$$

where g_t is the cumulative statistic. If this statistic exceed a predefined threshold, we raise a system-wide alarm:

$$t_a = \min\{t : g_t \geq h\} \quad (4.38)$$

where t_a is the alarm time and h is the threshold, which is chosen to minimize the false alarm rate and the detection delay. In the item-based approach, a separate detector is set up for each item in the system:

$$g_{t,j} = \max\{g_{t-1,j} + s_{tj}, 0\}, \quad g_{0,j} = 0 \quad (4.39)$$

If any of the cumulative statistics exceed the threshold, the system-wide alarm is set as follows:

$$t_a = \min\{t : \bigvee_{j=1}^J g_{t,j} \geq h\} \quad (4.40)$$

After the alarm is set, the beginning/ending time of the attack and the suspected profiles are determined. The beginning time of the attack corresponds to the last time the cumulative statistic is zero before the alarm is set, i.e. $t_b = \max\{t < t_a : g_t = 0\}$. The ending time of the attack corresponds to the first time when the cumulative statistic is zero after the alarm is set, i.e. $t_e = \min\{t > t_a : g_t = 0\}$. The order in the normal operation is $t_b < t_a < t_e$. The test users falling into the time interval $[t_b, t_e]$ are flagged as suspicious. The individual classification is performed based on the uni-variate statistics of these users such that if $s_t > 0$, the test user t is flagged as an attacker.

4.4 Experimental Study

In this section, we first provide the details of the test environment including the attack generation procedure, the dataset descriptions and the brief explanation of the baseline algorithms. We then perform extensive experiments to assess the robustness of the proposed framework to the attack and dataset characteristics. We demonstrate the results on both single profile and sequential detection tasks. The experimental results indicate that the proposed framework provides both

Algorithm 2 Sequential attack detection in recommender systems (User-based)

```
1: Offline Phase ;
2: Inputs:  $\mathbf{x}_{i,m}, \mathbf{y}_{i,n}, r_{ij}, \mathbf{z}_{j,l}, \mathbf{w}_{j,q}$  ;
3: for  $iter_{train} := 1$  to ( $iter_{max}$  or conv) do ▷ Train loop
4:   for  $iter_{var} := 1$  to (conv) do ▷ Variational loop
5:     Infer user covariances  $\Sigma_{ui}$  and means  $\mathbf{m}_{ui}$  by (4.16) and (4.17) ;
6:     Update variational parameters  $\psi_{i,n}, \mathbf{e}_{i,n}, \mathbf{g}_{i,n}$  respectively by (4.18), (4.14), (4.13);
7:   for  $iter_{var} := 1$  to (conv) do ▷ Variational loop
8:     Infer item covariances  $\Sigma_{vj}$  and means  $\mathbf{m}_{vj}$  ;
9:     Update variational parameters  $\psi_{j,q}, \mathbf{e}_{j,q}, \mathbf{g}_{j,q}$  respectively;
10:  MAP global variables  $\mathbf{W}_m, \mathbf{A}_l$  and noise covariances  $\Sigma_{xm}, \Sigma_{zl}$  by (4.19) and (4.21) ;
11:  MLE global variables  $\mathbf{H}_n, \mathbf{B}_q$  by (4.20) ;
12:  Update rating confidence  $c$  by (4.22)
13: for  $i := 1$  to  $I$  do ▷ Genuine EDF loop
14:   Infer user posterior covariance  $\Sigma_{ui}$  and mean  $\mathbf{m}_{ui}$  by (4.16) and (4.17) ;
15:   Evaluate rating likelihood  $p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j)$  by (B.1);
16:   Compute score  $d_i$  by (4.29) ;
17: Outputs:  $d_i$  ;
18: Online Phase ;
19: Inputs:  $\mathbf{x}_{t,m}, \mathbf{y}_{t,n}, r_{tj}$  ;
20: for  $t := 1$  to  $T$  do ▷ Test loop
21:   Infer user posterior covariance  $\Sigma_{ut}$  and mean  $\mathbf{m}_{ut}$  by (4.16) and (4.17) ;
22:   Evaluate rating likelihood  $p(r_{tj}|\mathbf{u}_t, \mathbf{v}_j)$  by (B.1);
23:   Compute score  $d_t$  by (4.29) ;
24:   Compute tail probability  $p_t$  by (4.32) ;
25:   Compute the statistic  $s_t$  by (4.33) ;
26:   Accumulate  $g_t$  by (4.37) ;
27:   if  $g_t \geq h$  then Raise alarm at  $t_a = t$  ; and break ;
```

consistent and superior performance over a wide range of test conditions compared to a range of baseline algorithms.

4.4.1 Test Environment

- **Attack Generation:** In the literature of attack detection in recommender systems, the researchers have resorted to generating artificial attack profiles to show the vulnerability of the existing recommender systems, and to design attack detection frameworks for mitigation. Generating an artificial attack profile requires filling the rating vector intelligently. Since

Table 4.1: Dataset statistics

	MovieLens	Book-crossing	LastFM
# of users	3883	5288	26360
# of items	6040	8902	9940
# of ratings	1000000	18797	1419212
Density(%)	4.3	0.03	0.54
Global rating mean	3.75	3.89	1.84
Side info of the users	Age, occupation, gender	Age, location	Age, country, gender
Side info of the items	Release year, genre	-	-

the main motivation is to affect a target item, a very low or high rating is usually assigned to this item. However, since all the genuine profiles have sparse rating vectors, the attacker generally chooses at least some filler items and rate them more realistically to hide their malicious activity and increase the attack efficiency. The fraction of the number of filler items to the total number of items in the system, is called the filler size. On the other hand, the attacker has to create multiple profiles as a single profile is not sufficient to disturb the overall system. The fraction of the amount of fake profiles injected to the system to the total number of users already registered, is called the attack size.

In terms of generating a single attack profile, we utilize several popular attack types developed for the recommendation systems in the literature. Although they are not different from each other in filler size, they differ by the filler item selection criteria, and the ratings given for the filler items and the target item. We can subsequently group them into three categories as push attacks, nuke attacks and obfuscated attacks. Push attacks are mainly designed to create a positive prediction shift for a target item by entering high ratings. Nuke attacks are conversely designed to create a negative prediction shift. The main motivation of the obfuscated attacks is to hide the attack from the detection algorithms by changing the filler selection criteria and the ratings given for them.

In terms of push attacks, we consider random, average, and bandwagon attacks. Generating the profile of a random attack requires accessing the global mean and the standard deviation of the ratings in the system. The filler items are selected randomly and given ratings from a

Gaussian distribution with the statistical parameters of the ratings. The filler item selection for the average attack is similarly random but it uses the means and the standard deviations of the filler items instead of the entire set of ratings in the system. The filler ratings are sampled from the specific distributions of the associated filler item instead. While the average attack is more sophisticated and successful, this property also makes it a higher knowledge attack than the random attack. The bandwagon on the other hand, uses some additional popular items as filler (5 items are used in the experiments of this study) to increase its efficiency further. These items are rated high to make the profiles more visible to the collaborative filtering algorithms.

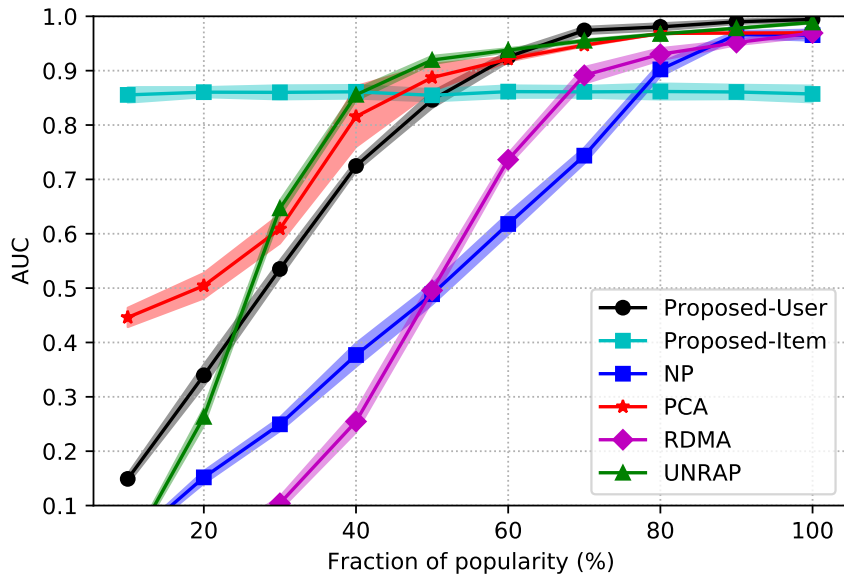
In terms of nuke attacks, we consider the reverse bandwagon and love-hate attacks. Similar and contrary to the bandwagon, the reverse bandwagon attack uses widely disliked items as filler. The filler items and the target item are rated low to nuke the target item. Love-hate attack selects random filler items and rate them as high, but the target item is rated low. In terms of obfuscated attacks, we consider popular and average over popular (AOP) attacks. To generate a popular attack, one needs to know the average ratings of the popular items. The selected popular items as filler are rated with respect to the global average and the item average. AOP is the modification of the average attack, in which the filler items are selected among the most popular items. This obfuscates the attack from the algorithms that use filler selection criteria to distinguish the attack profiles. The attributes of the attack profiles are chosen randomly from the pool of genuine user attributes. For each attribute of a profile, we first pick up a random genuine user then copy her corresponding attribute. This procedure is applied separately for each attribute and each profile resulting in a random but realistic attribute assignment for the attack profiles.

As previously discussed, a successful attacker should inject many fake profiles to affect a recommendation system. Another requirement to get quick and efficient results is that the attack has to be performed in a short time period. We name this type of attack as the sequential attack. We try to detect this type of attack by using the temporal relations of the profiles,

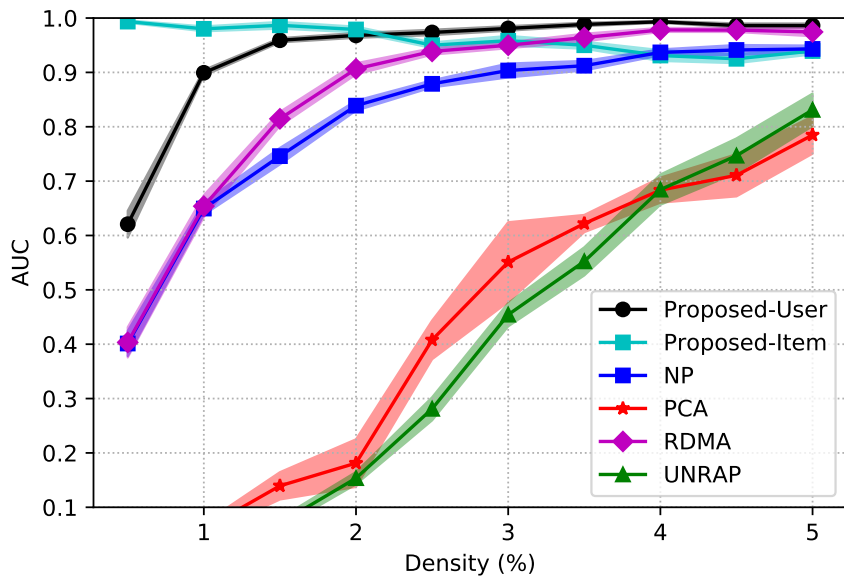
considering it as a persistent outlier. To generate a sequential attack, we generate a number of single attack profiles which represents the attack size. These profiles can be generated from either one of the single attack types as described or by mixing several of them randomly. In a real-world scenario, some genuine users may also interleave the attack sequence. To simulate this realistic scenario, we hold 10% of randomly selected genuine users from the dataset and train the model with the rest of the genuine users. We form a sequence of randomly selected 90% of the genuine held out users and mix into the remaining 10% the previously generated single attack profiles. The test set is formed by the concatenation of these two sequences.

- **Datasets:** In the experimental study, we consider three popular benchmark datasets: MovieLens (1M), Book-crossing and LastFM, to include both dataset size and conceptual diversity such as movies, books and music, to perform a robust assessment of the proposed framework. The MovieLens dataset provides 1M ratings of approximately 4K users on 6K movies. All the users and the movies have mixed data-type attributes. For instance, each user has age, gender and occupation information. We model the age as a one dimensional real-valued attribute, the gender as a binary categorical attribute, and the occupation as a 21-class categorical attribute. On the other hand, each item has the release year and genre information. We consider the release year as a one-dimensional real-valued attribute. The genre information is provided as a 19-dimensional binary vector where each genre is represented by one bit. Since it is not one-hot encoded, we model each bit as a 2-class categorical attribute.

Book-crossing dataset provides 1.1M ratings of 278K users on 270K books. Not all of the users have fully observed attributes thus we filter out the users with missing attributes for this study. The remaining users have the age and location information. We model the age similarly as a one-dimensional real-valued attribute, the location as an 8-class categorical attribute as we only consider English speaking countries. We filter the dataset to eliminate the imbalance of the countries, and to decrease the sparsity of the rating vectors of the users. For balance, we sample the users evenly based on their respective countries. For sparsity, we



(a) Obfuscation sensitivity when the AOP attack is injected



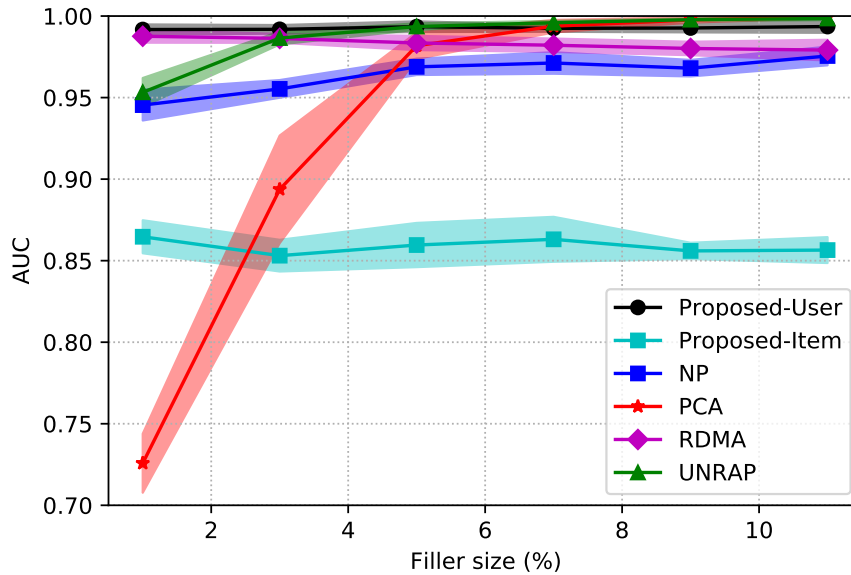
(b) Density sensitivity when the push attack is injected

Figure 4.2: Different characteristic of the two variants of proposed algorithm with respect to varying obfuscation of the attack and the sparsity of the dataset. The item based approach performs better in the case of high obfuscation and high sparsity although the user-based approach performs better in the remainder of the scenarios.

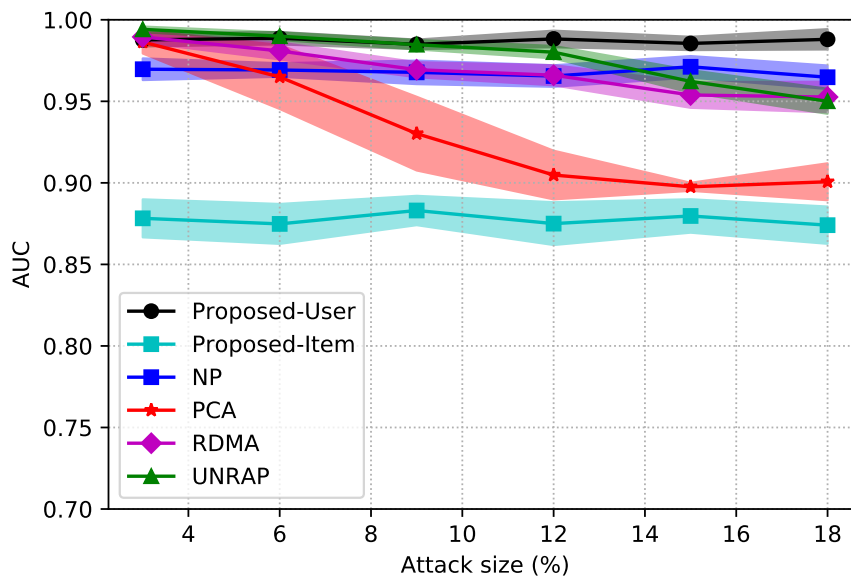
filter out the cold users and books which have small number of ratings. The resulting dataset consists of 19K ratings of 5K users on 9K books.

LastFM dataset provides 17M play counts of 360K users on 186K artists. We similarly filter out the users with missing attributes. The remaining users have age, gender and country information. We model the age as a one-dimensional real-valued attribute, the gender as a 2-class categorical attribute, and the country as a 10-class categorical attribute because we only consider the countries with large number of ratings. We further filter the dataset to decrease the sparsity of the rating vectors of the users to eliminate the cold users and play counts based on the low number of ratings. The resulting dataset consists of 1.4M ratings of 26K users on 10K books. We convert the play counts to integer pseudo ratings between 1 and 5. We first normalize the play counts for each user by dividing the individual play counts by the total play counts for that user. Then, each entry is log-transformed by using $\log(p_{i,j} + 1)$, where $p_{i,j}$ is the normalized play count. Finally, the entries of each user are scaled and quantized by ensuring the max entry of each user is 5. The statistics of all three datasets are shown in Table 4.1.

- **Baseline algorithms:** For performance comparison on single profile attack detection task, we consider four baseline algorithms. The first algorithm is a statistical test proposed in [115] called the NP-test, which is based on LLR. The distribution of the genuine profiles and attack profiles are obtained based on the item selection behavior, which is modeled through item popularity, and the integer ratings given for these items, which is modeled through Gaussian Q functions. The LLR value of each test user determines the classification, i.e, if positive, the test user is an attack profile. The second algorithm uses PCA to find out a metric associated with user correlations [125]. By assuming the attack users are correlated, PCA of the normalized design matrix is computed to select the least independent users by choosing the principal components associated with the smallest eigenvalues. The users are then sorted with a metric computed through the sum of squared values of a few principal components, and the top users are flagged as attack profiles. The third approach RDMA [119]



(a) Push attack with varying filler items

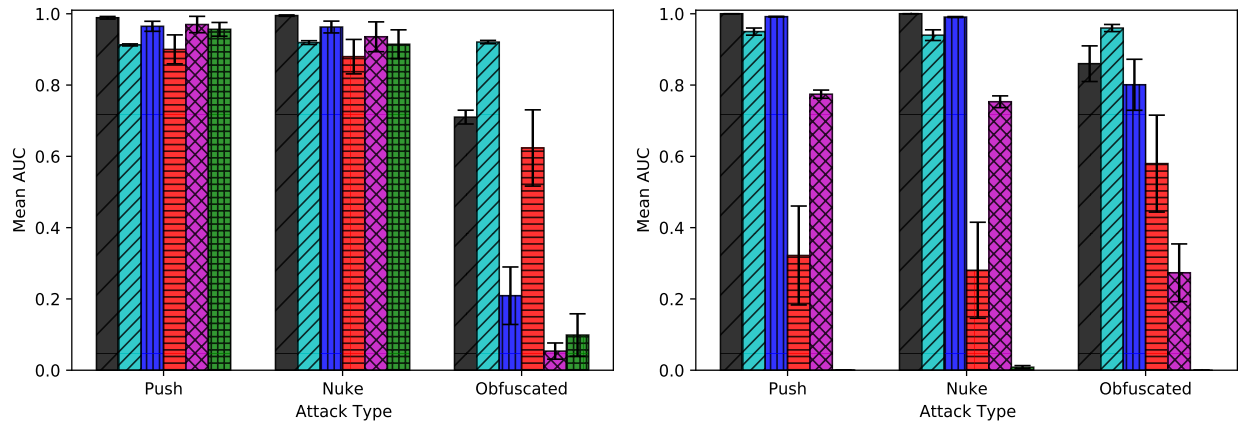


(b) Push attack with varying attack profiles

Figure 4.3: Average AUC scores when mixed push/nuke/obfuscated attacks are performed with different attack configurations, i.e, filler and attack size, to the MovieLens dataset. While the performances of most algorithms vary depending on the configuration, the proposed algorithm provides stable and accurate results among different conditions. More specifically, in the case of obfuscated attacks, the only reasonable performance is obtained by the proposed algorithm.

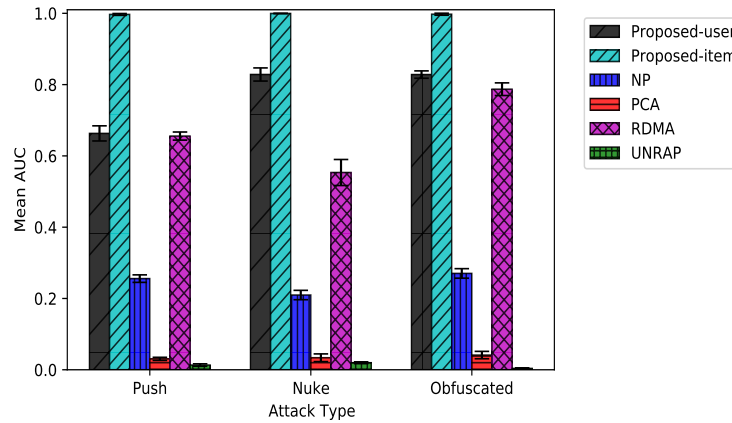
is a feature-based algorithm that forms distinctive features for the attack detection based on the ratings. The deviation of the ratings of each user from the mean values of the items the user rated is used as an indicator feature of an attack, i.e, if it is high, the user is classified as attacker. The last algorithm called UnRAP [121] is another feature based algorithm that uses H_v -score metric to distinguish genuine and attack profiles. This metric is computed through the squared deviation of the mean values of the user, the mean values of the items that the user rated, and the global mean of the overall rating matrix. A sliding window is then applied to examine the users having the top scores in descending order to filter out the users that didn't rate the target item. For performance comparison on sequential attack detection task, we consider two baseline algorithms based on the CUSUM framework and originally proposed for the reputation systems. We call these algorithms as mean detector [138] and GLR [139]. Both estimate unknown distributions of the ratings by using maximum likelihood estimation. The former assumes a Gaussian density for pre-post distributions and proposes a two-sided CUSUM to detect the mean shift of the ratings. The latter instead assumes a discrete probability model and estimate the model parameters of the rating samples given categorically distributed observations.

- **Metrics:** We use two metrics to assess the performance of the proposed framework and compare it with the baseline algorithms. For single profile detection, the problem can be regarded as a binary classification problem, and one can use the Area Under the Curve (AUC) metric as a single value indicator to measure the classification error of the profiles in the test set. It is evaluated as the area of the curve given the true positive rate against the false positive rate, which comprises the trade-off while selecting the threshold of the algorithms. For the optimal sequential detection scheme, the overall criterion is to minimize the delay for detection for a fixed time interval between false alarms [102]. To this end, to assess the sequential detection performance, we consider the plot of mean detection delay (MDD) against the log of a given false alarm period. The mean detection delay (MDD) is given as



(a) MovieLens

(b) LastFM



(c) BookCrossing

Figure 4.4: Average AUC scores when mixed push/nuke/obfuscated attacks are performed with different attack configurations, i.e, filler and attack size, to the three different datasets. While the performances of most algorithms vary depending on the configuration, the proposed algorithm provides stable and accurate results among different conditions. More specifically, in the case of obfuscated attacks, only the proposed algorithm provides acceptable scores for each of the three datasets whereas NP and RDMA are successful only in one out of three cases.

the average running length of the detection after the attack occurs. False alarm period is computed as the reciprocal of the average false alarm rate at a fixed detector threshold.

4.4.2 Robustness Assessment

In this section, we first consider the performance characteristics of the two variants of the proposed framework with respect to the obfuscation level and dataset sparsity. Then we change the

attack configuration, i.e, the filler size and the attack size and do performance comparison with the baseline algorithms.

- **Obfuscation and sparsity level:** We begin by evaluating the two variants of the framework with respect to the obfuscation level and dataset sparsity to show which variant is more suitable in particular cases. The experiments are conducted on the MovieLens dataset. To show the sensitivity of the variants with respect to the obfuscation level, we pick the AOP attack and change the fraction of popularity while fixing the attack size as %5 and filler size as the dataset average. At each fraction, a large number of experiments are repeated to obtain average performances where 100 different realizations of the attack profiles are generated for statistical significance. In each realization, the target item and the hold out users are randomly selected among the pool of the items and the users in the dataset. Fig. 4.2a indicates the comparison of the performances. The observation is that the user-based method fails when the attack is obfuscated by selecting the most popular items as filler although the item-based method provides a robust performance in this test condition. The reason is that while the user-based method aggregates the rating likelihoods of the attack profile, the item-based method uses only the rating likelihood of the target item. If the filler items are chosen and rated among the popular items, the likelihoods become nominal for the filler items. Hence, as the number of filler items suppresses the single target item, the user-based method tends to produce nominal anomaly scores due to effect of the popular items which reduces the detection performance. On the other hand, to show the sensitivity of the variants with respect to the dataset sparsity, we conduct experiments by varying the sparsity (density) of the MovieLens dataset by holding out the randomly selected rating entries with the same filler and attack size configurations. The observation from Fig. 4.2b is that although both of the variants provide reasonable performances among different sparsity conditions, the user-based method slightly degrades at high sparsity cases and the item-based method slightly degrades at low sparsity cases. The key reason is that the rating prediction performance of the latent variable model defined in Section 4.3.1 relies on the number of observed ratings

Table 4.2: Test setup

Dataset	Attack Type	Pre-distribution	Pre-mean	Post-distribution	Post-mean
MovieLens	Push	[0.06 0.16 0.39 0.31 0.07]	3.16	[0.05 0.10 0.10 0.35 0.45]	3.94
	Nuke	[0.01 0.02 0.19 0.41 0.37]	4.15	[0.45 0.35 0.10 0.10 0.05]	1.93
	Obfuscated	[0.32 0.19 0.3 0.16 0.04]	2.34	[0.05 0.10 0.10 0.35 0.45]	3.94
LastFM	Push	[0.85 0.13 0.02 0.01 0.01]	1.17	[0.05 0.05 0.30 0.30 0.30]	3.71
	Nuke	[0.18 0.45 0.09 0.18 0.09]	2.16	[0.45 0.35 0.10 0.10 0.05]	1.84
	Obfuscated	[0.54 0.32 0.07 0.05 0.02]	1.52	[0.05 0.05 0.30 0.30 0.30]	3.67

provided for the training. As the sparsity increases, the mean square error of the model increases, which results in noisy rating likelihoods. Summing up the negative likelihoods thus results in more noisy anomaly scores than using only a single likelihood as in the case of the item-based method.

- **Attack configuration:** We now pick the user-based method and compare it with the baseline algorithms to assess the robustness with respect to the attack configuration, i.e, filler size and attack size. Choosing a small filler size results in a low efficiency attack due to the nature of the collaborative filtering algorithms. However, a larger filler size can make the attack more visible creating an important trade-off. One could use the average filler size of the genuine users registered in the system, but it would require in-depth knowledge of the database, which may not be accessible to the attacker. It is also possible to design each attack profile with a random number of filler items. To this end, the filler size is a parameter for the attacker that is adjusted based on the purpose and the knowledge at hand. On the other hand, the attack detection performance should be reliable to this possibility of varying fractions of filler items. We design an experiment to show the performances of the single attack profile detection algorithms with respect to varying fractions of filler items. We repeat 100 experiments by generating mixed push attacks of size 5% with a 10% hold-out set and running the algorithms on MovieLens to classify the test users and obtain statistically significant results. We used a uniformly distributed average filler size in the range of 3% to 9% to place the average filler size of the dataset ($\sim 6\%$) in the middle of the distribution. Fig. 4.3a shows the average precision values of the algorithms in this setup. The results indicate

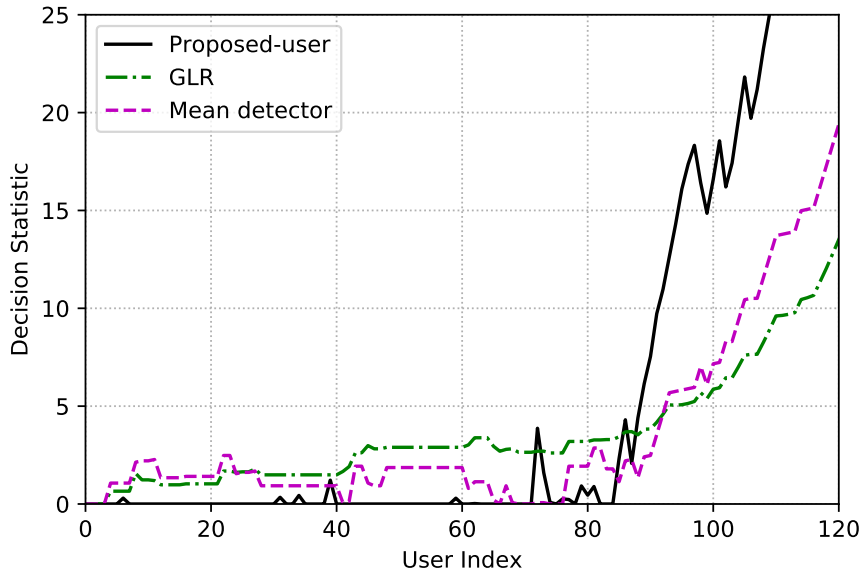
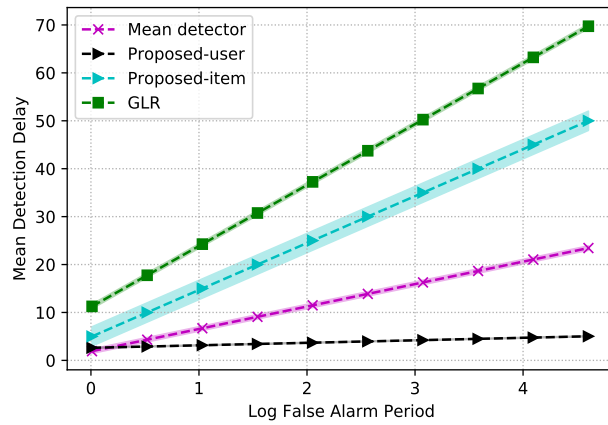
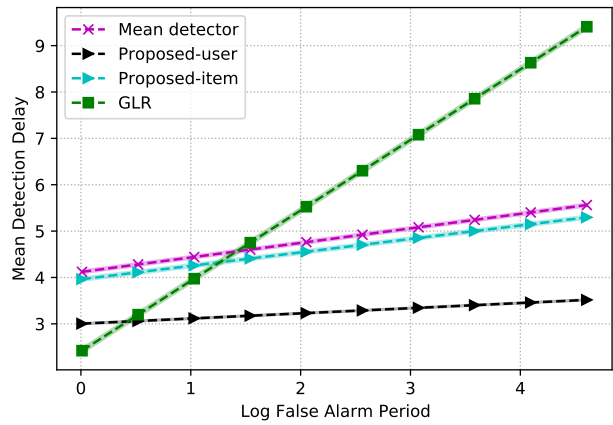


Figure 4.5: Sample decision statistics when push attack is injected for item #294 of Movielens dataset beginning from the user index 85.

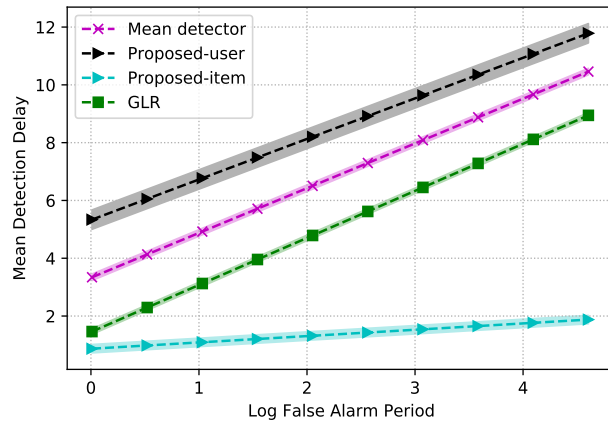
that the proposed algorithm is robust to the variations in the filler size while the baseline algorithms have varying performances. The attack size is the next parameter the attacker should consider for an efficient attack. The most efficient attack is the one with a small attack size and a high prediction shift which makes it unlikely on a well-secured platform since it generally requires an in-depth knowledge of the system. Increasing the number of attack profiles also increases the prediction shift but makes the attack more visible to the anomaly detection algorithm. Similar to the filler size, the attack detection algorithm should be robust to varying attack sizes. To assess their performance, we perform an experiment where we change the attack size fraction from %3 to %18 while fixing the hold-out set at %10 and we evaluate the mean AUC in the case of a mixed push attack with a filler size of %6. Fig. 4.3b shows the average results of 100 experiments for statistical significance. Similar to the experiments for the filler size, the results demonstrate the robustness of the proposed algorithm for different attack sizes.



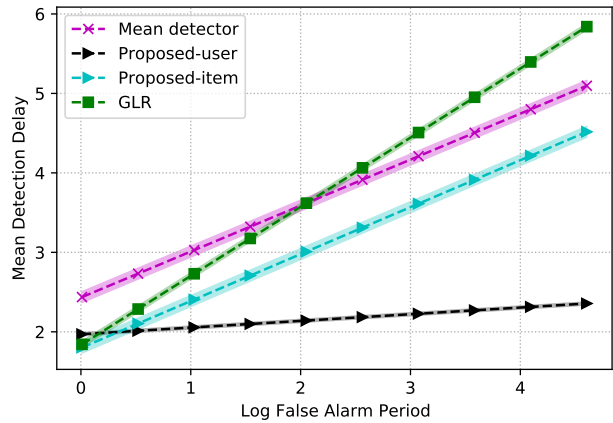
(a) Movielens-Push Attack



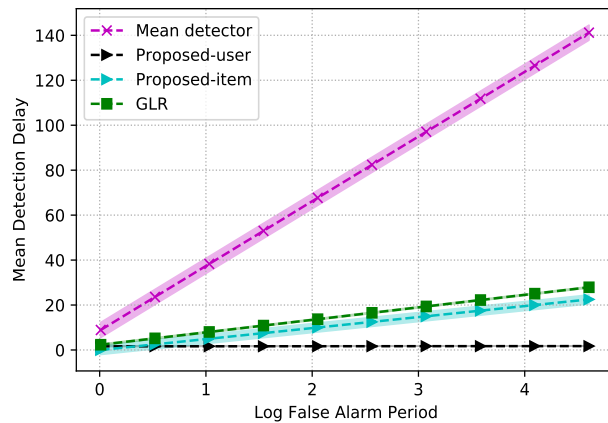
(b) Movielens-Nuke Attack



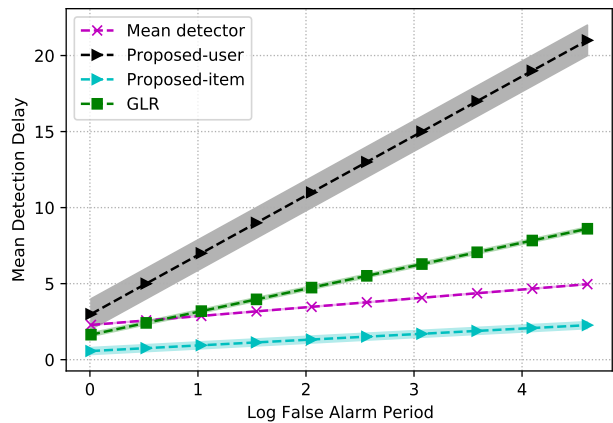
(c) Movielens-Obfuscated attack



(d) LastFM-Push Attack



(e) LastFM-Nuke Attack



(f) LastFM-Obfuscated Attack

Figure 4.6: Sequential detection performance when mixed push/nuke/obfuscated attacks are performed to the three different datasets. User-based method outperforms in the case of push/nuke attacks. Item-based method outperforms in the case of obfuscated attacks.

4.4.3 Profile Detection Performance

This section presents the overall performance evaluation of the competing algorithms in classification of the test user profiles as attacker or genuine on the three data sets described in Section 4.4.1. To get the average performances, we integrate the attack and filler size parameters by forming a 2d-grid. For the attack size, 10 equally spaced grid points are determined between the range of %3 to %10 of the total amount of users in the respective datasets. For the filler size variations, the average filler size of the dataset is first computed as \bar{f}_s , and later the respective range for each dataset is defined as % 0.5 \bar{f}_s to % 1.5 \bar{f}_s with 10 equally spaced grid points. 100 experiments are repeated with a randomly selected %10 hold-out set for statistical significance. The resulting mean AUCs are presented in Fig. 4.4 for each dataset and each mixed attack type separately. As listed in Table 4.1, the Book-crossing dataset is the one having the most sparse rating matrix and MovieLens is the one having the least. To this end, in the experiments, we used the item-based method for the Book-crossing dataset and the user-based method for the other two, based on the performance analysis performed in section 4.4.2. In terms of the attack type, we only consider the dataset sparsity for the selection between the item-based and user-based variants. While the item-based method is superior for some attack types, we have to assume no a priori information about the attack type. The results indicate that the two variants of the proposed framework both provide consistent detection performance over the rest of the algorithms for all the test conditions. The overall results are summarized as follows:

- The competing algorithms fail especially when the sparsity is very high. PCA fails because the filler size is very low in the case of sparse datasets, which increases the correlation between all the possible user profiles. NP fails as the distribution of the total amount of ratings the items get broader, the distribution lacks the sharp modes, which is the information NP relies upon. The performances of RDMA and UnRAP degrade due to the fact that the item means are noisy as the number of ratings per item is very low.

- Furthermore, the obfuscation attack drastically degrades the performances of the baseline algorithms. For both the MovieLens and LastFM datasets, the user-based method provides the most acceptable performance. PCA fails because the correlation between the genuine and attack user rating vectors becomes higher when the obfuscated attack is injected. RDMA and UnRAP fail since the attack type has knowledge of both the item means and the filler selection propensity of the genuine users. Among them, only the NP algorithm performs reasonably well on the LastFM dataset.

4.4.4 Sequential Detection Performance

We now focus on the performance of the sequential attack detection. In this case, the baseline algorithms are GLR and Mean-detector. Note that these algorithms only observe the rating sequence of a single test item and try to detect the changes in the distribution parameters. Since they need to infer the distribution parameters by using MLE, a long rating sequence for each item is necessary to get reasonable performance. Unfortunately, due to the high sparsity of its rating matrix, the Book-crossing dataset does not provide a sufficient number of ratings for almost any of the items it contains. To this end, we exclude the Book-crossing dataset from the second set of our experiments and use the MovieLens and LastFM datasets to achieve a fair comparison between all the algorithms. The test conditions are designed to provide distribution changes before and after the attack occurs. We consider three attack types as before including the push, nuke and obfuscated attacks. Although these attacks are designed to provide high (5) ratings for the push and obfuscated attacks, and low (1) ratings for the nuke attacks to the target item in consideration, we modify the attacks to provide distributions for the ratings of the target items instead of fixed values. We set the probabilities of the categorical distributions of the ratings after the attack occurs and generate independent samples from these distributions. Table 4.2 shows the parameters before and after the attack, used in the experiments of this section for each attack type and dataset. We fix the filler size to the average of each respective dataset \bar{f}_s , the hold-out size as %10, and the attack size as %5. Fig. 4.6 shows the sequential detection performances in these test conditions as MDD versus the

false alarm period plots. 500 repetitions are made for statistical significance. A sample realization when a push attack is injected to the MovieLens dataset, is shown as an example in Fig. 4.5. The results show that the two variants of the proposed framework (the user and item based) provide superior performance over the range of different distribution changes and datasets in comparison to the baseline algorithms. We have summarized the observations from the results as follows:

- As shown by the sample in Fig. 4.5 the decision statistic of the user-based method remains stable under genuine user entries. This implies that the model characterizes the genuine data correctly to prevent false alarms. After the attack occurs, the user-based method provides an abrupt change in the decision statistic, clearly indicating the beginning of the attack. On the contrary, the other two detectors need to slow their detection speed to avoid false alarms.
- For both datasets, when the push and nuke attacks are applied, the user-based method performs quicker detection than the other two algorithms to ensure fixed false alarm rates. We also observe that the mean detector performs slightly better than GLR for the conditions of this test setup.
- As we mentioned in 4.4.2, the performance of the user-based method degrades under heavy obfuscation. The results in Fig. 4.6(c) and Fig. 4.6(f) clearly indicate this behavior for the sequential detection task. However, the item-based method is robust to this condition and provides better performance than all the competing algorithms on both datasets.

The experimental results demonstrate that both variants of the proposed framework can effectively exploit the user attributes as an informative additional anomaly data source and clearly outperform the competing algorithm for a wide variety of test scenarios.

4.5 Conclusion

In this chapter, we address the problem of sequential attack detection in recommender systems. The proposed framework is completely unsupervised and require no a priori knowledge of the attacking strategies for ultimate flexibility and robustness compared to alternative approaches.

The latent variable model in the proposed framework provides a rich latent space, in which the users are represented by real valued latent variables given their sparse rating vectors and the mixed-data type attributes. Producing uni-variate statistics from this space to be accumulated in a CUSUM-like algorithm, our sequential detector provides a robust detection performance by aggregating the anomalies coming from the mismatch between the ratings and the attributes for a wide range of attack types, sizes and datasets. The experimental study shows the advantages of the two variants of our detector (user and item-based) on varying attacking strategies and configurations on three popular real-word datasets with completely different characteristics and statistics. In summary, the proposed framework provides higher accuracy and quicker detection over the competing methods in detecting anomalous activity in recommender systems.

As future work, the individual contributions of each side attribute to the detection performance can be analyzed both statistically and empirically. Furthermore, more sophisticated attack types on a greater range of datasets with different sparsity levels can be considered. Our hypothesis is that as the attack types get more sophisticated and the datasets more diverse, the true advantage of using side attributes will become even more apparent.

Chapter 5: Concluding Remarks

5.1 Summary

Recommender systems are key components of today's commercial platforms. These systems are data-driven, thus require high quality data to infer useful predictions. Today, it is possible to obtain unprecedented amounts of data due to the global connectivity of the information systems. The platforms can easily access useful information sources for their applications. However, these sources are potentially diverse, heterogeneous, sparse, noisy, fake, and imbalanced. For example, the rating matrix, which is the core information source of recommender systems for personalization, is very sparse and imbalanced. It can also contain fake entries of malicious profiles. The user demographics are heterogeneous including real valued, integer and categorical data types. Hence, the algorithm should consider all the aforementioned problems, and still produce high quality recommendations by fusing and exploiting available useful information. In this thesis, our main focus has been to fuse the available information to improve the recommendation quality, and the detection of malicious profiles by using diverse information sources.

The key point for recommender systems is imputing the missing values in the rating matrix with accurate and computationally efficient methods as in matrix completion. Under the umbrella of matrix completion, many algorithms exist, however the applicability of these algorithms changes from application to application. The performances of the algorithms mostly depend on the dataset specs, i.e., the dataset size such as the width and height of the matrix, the dataset sparsity, and the data-types of the entries. In Chapter 2, we briefly explained existing algorithms before introducing our proposed algorithm, FSNMC, as a self-supervised neural network based matrix completion algorithm with an alternating learning scheme. We demonstrated its superior performance over baseline algorithms on specific dataset specs. FSNMC outperforms others when: i) the dataset

size is small and sparsity level can change from 0% - 90%, and ii) the dataset size is medium and sparsity level can change from 0% - 50%.

However, the datasets in recommender systems are usually very big in terms of both height and width, highly sparse (>99%), and imbalanced, i.e, the observations do not homogeneously appear in terms of spatial position. Most of the matrix completion algorithms fail in these conditions including our FSNMC. Another concern is the availability of side information, which is hard to incorporate for a typical matrix completion algorithm. In Chapter 3, we proposed a latent variable model that can incorporate the side information of both users and items in a typical recommender system. The advantages of this algorithm is mainly the computational efficiency, i.e. it scales linearly in both the number of users and the number of items, and its superior performance over a dozen baseline algorithms. It also allows inference in the case of missing values in any information source with a principled probabilistic way.

Besides providing accurate recommendations to the users, one important property a recommender system has to have is trustworthiness. As we discussed in Chapter 4, the recommender systems are highly prone to malicious activities. One can create fake profiles, and inject fake ratings to effect the recommendations of other users. To prevent such activities, the fake profiles can be detected through authentication. However, it is still possible to trick this anterior system, and inject fake ratings. In this case, an algorithm to analyze the ratings and to detect the anomalies is necessary to maintain the quality of the platform. In Chapter 4, we proposed a sequential framework that analyzes not only the rating matrix but also the side information of the users to detect fake profiles quickly and accurately. Our proposed framework outperforms all the baseline algorithms that use only the rating matrix and provides more accurate detection rate. We also performed experiments on sequential detection performance and demonstrated the proposed framework's superior performance over two recent baseline sequential detectors.

5.2 Insights and Possible Future Work

Although the proposed fusion model used in this dissertation has been applied to incorporate the user demographics and item features, the framework can also support many other side information sources. For instance, the model can be expanded to incorporate text reviews of the users, cross domain information from different platforms, user trust/friendship network, and contextual information. Text reviews can be incorporated by following the correlated topic models [55, 145], where the observations are modeled as multinomial. When the latent variables are still modeled as Gaussian, it becomes straightforward to extend our model. Cross-domain information is especially useful for the cold users. One can perform predictions by analyzing the rating behavior of a cold user in different domain such as music by first computing the similarities with the other users within that domain, and then exploiting the similarities in the original target domain such as movie. These similarities are usually in the form of kernels [51]. These kernels can be incorporated to our framework as additional informative priors for the user latent variables instead of zero mean spherical priors, which creates a connection to Gaussian processes [76]. User trust/friendship network is similar to the cross-domain information as the former provides binary information. We can incorporate this information by using the same method as informative priors for user latent variables. The contextual information include the time/location of the users and the item popularity with respect to time/location slices. The location can be incorporated by modeling the observation as a Von Mises distribution [74]. To model the time behavior, the latent variables should be converted to stochastic processes similar to Hidden Markov Models [76].

Our proposed latent variable model relies on a lower bound to the categorical likelihood, called Bohning when incorporating categorical data. Although this bound leads to efficient variational inference in terms of computation, the performance can be further improved by using the recent findings on the lower bounds to lse function [146]. Another extension can be to exploit stochastic inference. Although the proposed framework in this study scales linearly in terms of computational complexity, the space complexity will be a problem for very large datasets since the algorithm performs full batch training. To alleviate this problem, one should modify the update

equations to enable online EM [147] or change the inference method slightly to enable stochastic variational inference [96]. The last possible extension can be to increase the number of parameters to increase the model capacity. The proposed framework is linear thus having a small number of global parameters to learn the connections between the instances. To increase the capacity, one proper extension is to increase the level of the model. Multilevel models [148] provide more parameters thus enabling the discovery of deeper connections between the instances. The caveat is to make efficient inference in this family of tiered models.

The main component of the attack detection framework is the proposed latent variable model, thus its aforementioned extensions will definitely improve the detection performance as well. Another extension can be to use more relevant information sources. Although we use user attributes in addition to the rating matrix, one direct extension can be to use the contextual information, and textual reviews to improve the performance further. Contextual information can help because the attackers can try to manipulate the system from irrelevant locations at irrelevant times. The timestamps of the rating entries may also provide useful information about an attack. When the model is trained with genuine data, it can infer the relationship between the user preferences and the contextual information. The deviation from this pattern can improve the anomaly detection. On some platforms, textual reviews are one of the main sources of explicit feedback. There is significant research to analyze textual data to infer the sentiments, the intention, and the authenticity [149]. It can be a useful extension of our framework if one can fuse the textual information with other relevant sources such as rating matrix, user attributes, contextual information, and increase the attack detection performance together with recommendation accuracy. However, these extensions and insights will need immense validation both theoretically and empirically by using many different datasets and attack types well suited for future work.

References

- [1] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, no. 8, pp. 30–37, 2009.
- [2] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016.
- [3] F. Ricci, L. Rokach, and B. Shapira, “Introduction to recommender systems handbook,” in *Recommender systems handbook*. Springer, 2011, pp. 1–35.
- [4] A. Gogna and A. Majumdar, “Matrix completion incorporating auxiliary information for recommender system design,” *Expert Systems with Applications*, vol. 42, no. 14, pp. 5789–5799, 2015.
- [5] Y. Liu, Y. Pan, Z. Sun, and D. Huang, “Statistical monitoring of wastewater treatment plants using variational bayesian pca,” *Industrial & Engineering Chemistry Research*, vol. 53, no. 8, pp. 3272–3282, 2014.
- [6] H. Wang, R. Zhao, and Y. Cen, “Rank adaptive atomic decomposition for low-rank matrix completion and its application on image recovery,” *Neurocomputing*, vol. 145, pp. 374–380, 2014.
- [7] S. Mercier, M. Mondor, B. Marcos, C. Moresoli, and S. Villeneuve, “Estimation of missing values in a food property database by matrix completion using pca-based approaches,” *Chemometrics and Intelligent Laboratory Systems*, vol. 166, pp. 37–48, 2017.

- [8] X.-Y. Pan, Y. Tian, Y. Huang, and H.-B. Shen, "Towards better accuracy for missing value estimation of epistatic miniarray profiling data by a novel ensemble approach," *Genomics*, vol. 97, no. 5, pp. 257–264, 2011.
- [9] M. Aktukmak, S. Mercier, and I. Uysal, "A neural net framework for accumulative feature-based matrix completion," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–6.
- [10] A. Ilin and T. Raiko, "Practical approaches to principal component analysis in the presence of missing values," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 1957–2000, 2010.
- [11] B. Walczak and D. Massart, "Dealing with missing data: Part i," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 1, pp. 15–27, 2001.
- [12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
- [13] Y. Shan and G. Deng, "Kernel pca regression for missing data estimation in dna microarray analysis," in *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 2009, pp. 1477–1480.
- [14] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transportation research part C: emerging technologies*, vol. 34, pp. 108–120, 2013.
- [15] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, "Non-linear pca: a missing data approach," *Bioinformatics*, vol. 21, no. 20, pp. 3887–3895, 2005.
- [16] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

- [17] J. Fan and T. Chow, “Deep learning based matrix completion,” *Neurocomputing*, vol. 266, pp. 540–549, 2017.
- [18] M. H. Nguyen and F. Torre, “Robust kernel principal component analysis,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1185–1192.
- [19] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [20] A. Folch-Fortuny, F. Arteaga, and A. Ferrer, “Pca model building with missing data: New proposals and a comparative study,” *Chemometrics and Intelligent Laboratory Systems*, vol. 146, pp. 77–88, 2015.
- [21] S. Mercier and I. Uysal, “Noisy matrix completion on a novel neural network framework,” *Chemometrics and Intelligent Laboratory Systems*, vol. 177, pp. 1–7, 2018.
- [22] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio, “Llorma: Local low-rank matrix approximation,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 442–465, 2016.
- [23] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [24] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.
- [25] I.-C. Yeh, “Modeling slump flow of concrete using second-order regressions and artificial neural networks,” *Cement and concrete composites*, vol. 29, no. 6, pp. 474–480, 2007.
- [26] H. Canada, *Nutrient Value of Some Common Foods - Booklet*, 2008.
- [27] M. Lichman, *UCI Machine Learning Repository, Wine*. Irvine, CA: University of California, School of Information and Computer Science, 2008.

- [28] M. Forina, C. Armanino, S. Lanteri, and E. Tiscornia, “Classification of olive oils from their fatty acid composition,” in *Food research and data analysis: proceedings from the IUFOST Symposium, September 20-23, 1982, Oslo, Norway/edited by H. Martens and H. Russwurm, Jr.* London: Applied Science Publishers, 1983., 1983.
- [29] M. Lichman, *UCI Machine Learning Repository, Protein Tertiary Structure*. Irvine, CA: University of California, School of Information and Computer Science, 2008.
- [30] ———, *UCI Machine Learning Repository, Abalone*. Irvine, CA: University of California, School of Information and Computer Science, 2008.
- [31] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 3, 2014.
- [32] A. Taneja and A. Arora, “Recommendation research trends: review, approaches and open issues,” *International Journal of Web Engineering and Technology*, vol. 13, no. 2, pp. 123–186, 2018.
- [33] M. Aktukmak, Y. Yilmaz, and I. Uysal, “A probabilistic framework to incorporate mixed-data type features: Matrix factorization with multimodal side information,” *Neurocomputing*, vol. 367, pp. 164 – 175, 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231219311464>
- [34] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [35] A. Ilin and T. Raiko, “Practical approaches to principal component analysis in the presence of missing values,” *Journal of Machine Learning Research*, vol. 11, pp. 1957–2000, 2010.
- [36] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational Inference: A Review for Statisticians,” *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

- [37] A. W. Van der Vaart, *Asymptotic statistics*. Cambridge university press, 2000, vol. 3.
- [38] A. Mnih and R. R. Salakhutdinov, “Probabilistic matrix factorization,” in *Advances in neural information processing systems*, 2008, pp. 1257–1264.
- [39] R. Salakhutdinov and A. Mnih, “Bayesian Probabilistic Matrix Factorization using MCMC,” *25th International Conference on Machine Learning (ICML- 2008)*, 2008.
- [40] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio, “Llorma: local low-rank matrix approximation,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 442–465, 2016.
- [41] K. Wang, W. X. Zhao, H. Peng, and X. Wang, “Bayesian probabilistic multi-topic matrix factorization for rating prediction,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-Janua, pp. 3910–3916, 2016.
- [42] W. Ma, Y. Wu, M. Gong, C. Qin, and S. Wang, “Local Probabilistic Matrix Factorization for Personal Recommendation,” *Proceedings - 13th International Conference on Computational Intelligence and Security, CIS 2017*, vol. 2018-Janua, pp. 97–101, 2018.
- [43] Z. Ghahramani, G. E. Hinton *et al.*, “The em algorithm for mixtures of factor analyzers,” *University of Toronto*, vol. Vol. 60, 1996.
- [44] D. Li, C. Chen, W. Liu, T. Lu, N. Gu, and S. Chu, “Mixture-rank matrix approximation for collaborative filtering,” in *Advances in Neural Information Processing Systems*, 2017, pp. 477–485.
- [45] S. Zhang, L. Yao, and A. Sun, “Deep learning based recommender system: A survey and new perspectives,” *arXiv preprint arXiv:1707.07435*, 2017.
- [46] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, “AutoRec : Autoencoders Meet Collaborative Filtering,” *WWW 2015 Companion: Proceedings of the 24th International Conference on World Wide Web*, pp. 111–112, 2015.

- [47] S. Li, J. Kawale, and Y. Fu, “Deep collaborative filtering via marginalized denoising auto-encoder,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 811–820.
- [48] J. Fan and T. Chow, “Deep learning based matrix completion,” *Neurocomputing*, vol. 266, pp. 540–549, 2017.
- [49] D. Agarwal and B.-C. Chen, “Regression-based latent factor models,” *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (2009)*, p. 19, 2009.
- [50] Y.-D. Kim and S. Choi, “Scalable variational Bayesian matrix factorization with side information,” *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS), 2014*, 2014.
- [51] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro, “Kernelized probabilistic matrix factorization: Exploiting graphs and side information,” in *Proceedings of the 2012 SIAM international Conference on Data mining*. SIAM, 2012, pp. 403–414.
- [52] Z. Wang, J. Liang, and R. Li, “Exploiting user-to-user topic inclusion degree for link prediction in social-information networks,” *Expert Systems with Applications*, vol. 108, pp. 143–158, 2018.
- [53] I. Porteous, A. U. Asuncion, and M. Welling, “Bayesian matrix factorization with side information and dirichlet process mixtures.” in *AAAI*, 2010.
- [54] C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.
- [55] D. Blei, M. Jordan, and A. Y. Ng, “Latent Dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

- [56] M. Jamali and M. Ester, “A matrix factorization technique with trust propagation for recommendation in social networks,” in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 135–142.
- [57] M. Saveski and A. Mantrach, “Item cold-start recommendations: learning local collective embeddings,” in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 89–96.
- [58] M. E. Khan, Y. J. Ko, and M. Seeger, “Scalable collaborative bayesian preference learning,” in *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, vol. 33, no. EPFL-CONF-196605, 2014, pp. 475–483.
- [59] X. Li and J. She, “Collaborative variational autoencoder for recommender systems,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 305–314.
- [60] S. Rendle, “Factorization machines,” in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.
- [61] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, “Fast context-aware recommendations with factorization machines,” in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011, pp. 635–644.
- [62] A. Taneja and A. Arora, “Cross domain recommendation using multidimensional tensor factorization,” *Expert Systems with Applications*, vol. 92, pp. 304–316, 2018.
- [63] M. M. Khan, R. Ibrahim, M. Younas, I. Ghani, and S. R. Jeong, “Facebook interactions utilization for addressing recommender systems cold start problem across system domain,” *Journal of Internet Technology*, vol. 19, no. 3, pp. 861–870, 2018.

- [64] I. Fernández-Tobías, I. Cantador, P. Tomeo, V. W. Anelli, and T. Di Noia, “Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization,” *User Modeling and User-Adapted Interaction*, pp. 1–44.
- [65] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota, “Response prediction using collaborative filtering with hierarchies and side-information,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 141–149.
- [66] M. Kula, “Metadata embeddings for user and item cold-start recommendations,” *CEUR Workshop Proceedings*, vol. 1448, pp. 14–21, 2015.
- [67] A. Elbadrawy and G. Karypis, “User-specific feature-based similarity models for top-n recommendation of new items,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 33, 2015.
- [68] I. Barjasteh, R. Forsati, D. Ross, A.-H. Esfahanian, and H. Radha, “Cold-start recommendation with provable guarantees: A decoupled approach,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1462–1474, 2016.
- [69] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, “Learning attribute-to-feature mappings for cold-start recommendations.” Citeseer.
- [70] Z. Zhang, Y. Liu, Z. Zhang, and B. Shen, “Fused matrix factorization with multi-tag, social and geographical influences for poi recommendation,” *World Wide Web*, pp. 1–16.
- [71] A. Klami, S. Virtanen, and S. Kaski, “Bayesian exponential family projections for coupled data sources,” *arXiv preprint arXiv:1203.3489*, 2012.
- [72] M. E. Khan, G. Bouchard, K. P. Murphy, and B. M. Marlin, “Variational bounds for mixed-data factor analysis,” in *Advances in Neural Information Processing Systems*, 2010, pp. 1108–1116.

- [73] Y. Yilmaz and A. O. Hero, “Multimodal factor analysis,” *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, vol. 2015-Novem, no. 2, pp. 1–6, 2015.
- [74] Y. Yilmaz and A. O. Hero, “Multimodal event detection in twitter hashtag networks,” *Journal of Signal Processing Systems*, vol. 90, no. 2, pp. 185–200, 2018.
- [75] D. Böhning, “Multinomial logistic regression algorithm,” *Annals of the institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992.
- [76] K. P. Murphy, *Machine learning: a probabilistic perspective*, 2012.
- [77] Y. Koren, “Factor in the neighbors: Scalable and accurate collaborative filtering,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, p. 1, 2010.
- [78] F. Ricci, L. Rokach, and B. Shapira, “Introduction to recommender systems handbook,” in *Recommender systems handbook*. Springer, 2011, pp. 1–35.
- [79] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, “An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.
- [80] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in neural information processing systems*, 2001, pp. 556–562.
- [81] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.
- [82] D. Lemire and A. Maclachlan, “Slope one predictors for online rating-based collaborative filtering,” in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 471–475.

- [83] T. George and S. Merugu, “A scalable collaborative filtering framework based on co-clustering,” in *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE, 2005, pp. 4–pp.
- [84] M. F. Dacrema, P. Cremonesi, and D. Jannach, “Are we really making much progress? a worrying analysis of recent neural recommendation approaches,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 101–109.
- [85] J. Lin, “The neural hype and comparisons against weak baselines,” in *ACM SIGIR Forum*, vol. 52, no. 2. ACM New York, NY, USA, 2019, pp. 40–51.
- [86] M. Ludewig and D. Jannach, “Evaluation of session-based recommendation algorithms,” *User Modeling and User-Adapted Interaction*, vol. 28, no. 4-5, pp. 331–390, 2018.
- [87] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel, “Improvements that don’t add up: ad-hoc retrieval results since 1998,” in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 601–610.
- [88] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters,” *arXiv preprint arXiv:1709.06560*, 2017.
- [89] Z. C. Lipton and J. Steinhardt, “Troubling trends in machine learning scholarship,” *arXiv preprint arXiv:1807.03341*, 2018.
- [90] K. Wagstaff, “Machine learning that matters,” *arXiv preprint arXiv:1206.4656*, 2012.
- [91] T. Ebesu, B. Shen, and Y. Fang, “Collaborative memory network for recommendation systems,” in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 515–524.
- [92] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, “Leveraging meta-path based context for top-n recommendation with a neural co-attention model,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

- [93] H. Wang, N. Wang, and D.-Y. Yeung, “Collaborative deep learning for recommender systems,” in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.
- [94] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.
- [95] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, “Spectral collaborative filtering,” in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 311–319.
- [96] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, “Stochastic variational inference,” *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.
- [97] M. P. O. Mahony, N. J. Hurley, and G. C. M. Silvestre, “An Evaluation of Neighbourhood Formation on the Performance of Collaborative Filtering,” pp. 215–228, 2004.
- [98] M. Aktukmak, Y. Yilmaz, and I. Uysal, “Quick and accurate attack detection in recommender systems through user attributes,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 348–352. [Online]. Available: <https://doi.org/10.1145/3298689.3347050>
- [99] C. C. Aggarwal, “Recommender systems: The textbook,” *Springer Publishing Company, Incorporated*, 2018.
- [100] M. Si and Q. Li, “Shilling attacks against collaborative recommender systems: a review,” *Artificial Intelligence Review*, pp. 1–29.
- [101] M. Aktukmak, Y. Yilmaz, and I. Uysal, “A probabilistic framework to incorporate mixed-data type features: Matrix factorization with multimodal side information,” *Neurocomputing*, vol. 367, pp. 164 – 175, 2019.

- [102] N. Peach, M. Basseville, and I. V. Nikiforov, “Detection of Abrupt Changes: Theory and Applications.” *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, vol. 158, no. 1, p. 185, 1995.
- [103] H. V. Poor and O. Hadjiliadis, *Quickest detection*. Cambridge University Press, 2008.
- [104] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, “A review of novelty detection,” *Signal Processing*, vol. 99, pp. 215–249, 2014.
- [105] E. U. Seyitoglu, A. A. Yavuz, and M. O. Ozmen, “Compact and resilient cryptographic tools for digital forensics,” in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.
- [106] E. U. A. Seyitoglu, “Efficient forward-secure and compact signatures for the internet of things (iot),” Ph.D. dissertation, University of South Florida, 2020.
- [107] J.-B. Subils, J. Perez, P. Liu, S. Engram, C. Cetin, D. Goldgof, N. Ebner, D. Oliveira, and J. Ligatti, “A dual-task interference game-based experimental framework for comparing the usability of authentication methods,” in *12th International Conference on Human System Interaction (HSI)*. IEEE, 2019.
- [108] S. K. Lam and J. Riedl, “Shilling recommender systems for fun and profit,” *Thirteenth International World Wide Web Conference Proceedings, WWW2004*, pp. 393–402, 2004.
- [109] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, “Identifying attack models for secure recommendation,” *Beyond Personalization 2005*, p. 19.
- [110] M. O. Mahony, N. J. Hurley, and G. C. M. Silvestre, “An Evaluation of the Performance of Collaborative Filtering,” *Proceedings of the 14th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS), 17th–19th*, pp. 164–168, 2003.

- [111] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Effective attack models for shilling item-based collaborative filtering systems,” in *Proceedings of the WebKDD Workshop*. Cite-seer, 2005, pp. 13–23.
- [112] R. Burke, B. Mobasher, and R. Bhaumik, “Limited knowledge shilling attacks in collaborative filtering systems,” *Proceedings of 3rd International Workshop on Intelligent Techniques for Web Personalization in conjunction with The 19th International Joint Conference on Artificial Intelligence*, no. January, pp. 17–24, 2005.
- [113] D. C. Wilson and C. E. Seminario, “When power users attack: Assessing impacts in collaborative recommender systems,” *RecSys 2013 - Proceedings of the 7th ACM Conference on Recommender Systems*, pp. 427–430, 2013.
- [114] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness,” *ACM Transactions on Internet Technology*, vol. 7, no. 4, 2007.
- [115] N. Hurley, Z. Cheng, and M. Zhang, “Statistical attack detection,” in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 149–156.
- [116] Z. Cheng and N. Hurley, “Effective diverse and obfuscated attacks on model-based recommender systems,” in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 141–148.
- [117] K. Christakopoulou and A. Banerjee, “Adversarial attacks on an oblivious recommender,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 322–330.
- [118] A. M. Rashid, G. Karypis, and J. Riedl, “Influence in ratings-based recommender systems: An algorithm-independent approach,” *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, pp. 556–560, 2005.

- [119] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th annual ACM international workshop on Web information and data management*. ACM, 2005, pp. 67–74.
- [120] C. A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: Detection of profile injection attacks," *Service Oriented Computing and Applications*, vol. 1, no. 3, pp. 157–170, 2007.
- [121] K. Bryan, M. O'Mahony, and P. Cunningham, "Unsupervised retrieval of attack profiles in collaborative recommender systems," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 155–162.
- [122] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 542–547.
- [123] H. Xia, B. Fang, M. Gao, H. Ma, Y. Tang, and J. Wen, "A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique," *Information Sciences*, vol. 306, pp. 150–165, 2015.
- [124] B. Mehta, T. Hofmann, and P. Fankhauser, "Lies and propaganda: Detecting spam users in collaborative filtering," *International Conference on Intelligent User Interfaces, Proceedings IUI*, pp. 14–21, 2007.
- [125] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *User Modeling and User-Adapted Interaction*, vol. 19, no. 1-2, pp. 65–97, 2009.
- [126] C. Li and Z. Luo, "Detection of shilling attacks in collaborative filtering recommender systems," in *2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. IEEE, 2011, pp. 190–193.

- [127] Z. Yang, Z. Cai, and X. Guan, “Estimating user behavior toward detecting anomalous ratings in rating systems,” *Knowledge-Based Systems*, vol. 111, pp. 144–158, 2016.
- [128] Z. Zhang and S. R. Kulkarni, “Graph-based detection of shilling attacks in recommender systems,” in *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2013, pp. 1–6.
- [129] —, “Detection of shilling attacks in recommender systems via spectral clustering,” in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.
- [130] Z. Yang, L. Xu, Z. Cai, and Z. Xu, “Re-scale adaboost for attack detection in collaborative filtering recommender systems,” *Knowledge-Based Systems*, vol. 100, pp. 74–88, 2016.
- [131] T. Tang and Y. Tang, “An effective recommender attack detection method based on time SFM factors,” *2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011*, pp. 78–81, 2011.
- [132] F. Zhang, Z. Zhang, P. Zhang, and S. Wang, “Ud-hmm: An unsupervised method for shilling attack detection based on hidden markov model and hierarchical clustering,” *Knowledge-Based Systems*, vol. 148, pp. 146–166, 2018.
- [133] R. Bhaumik, C. Williams, B. Mobasher, and R. Burke, “Securing collaborative filtering against malicious attacks through anomaly detection,” in *Proceedings of the 4th Workshop on Intelligent Techniques for Web Personalization (ITWP’06), Boston*, vol. 6, 2006.
- [134] B. Sch, J. C. Platt, J. Shawe-taylor, and R. C. Williamson, “Schölkopf et al. - 2001 - Estimating the support of a high-dimensional distribution.pdf,” vol. 1471, pp. 1443–1471, 2001.
- [135] G. Rätsch, S. Mika, B. Schölkopf, and K. R. Müller, “Constructing boosting algorithms from SVMs: An application to one-class classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, 2002.

- [136] M. Wu and J. Ye, “A small sphere and large margin approach for novelty detection using training data with outliers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2088–2092, 2009.
- [137] H. Chen, “Sequential change-point detection based on nearest neighbors,” *Annals of Statistics*, vol. 47, no. 3, pp. 1381–1407, 2019.
- [138] Y. Liu and Y. L. Sun, “Anomaly Detection in Feedback-based Reputation Systems through Temporal and Correlation Analysis,” *2010 IEEE Second International Conference on Social Computing*, pp. 65–72, 2010.
- [139] S. Li and X. Wang, “Quickest attack detection in multi-agent reputation systems,” *IEEE Journal on Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 653–666, 2014.
- [140] A. Ilin and T. Raiko, “Practical approaches to principal component analysis in the presence of missing values,” *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 1957–2000, 2010.
- [141] C. C. Johnson, “Logistic matrix factorization for implicit feedback data,” in *Advances in neural information processing systems*, vol. 27, 2014.
- [142] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. Blei, “Bayesian nonparametric poisson factorization for recommendation systems,” in *Artificial Intelligence and Statistics*, 2014, pp. 275–283.
- [143] D. Böhning, “Multinomial logistic regression algorithm,” *Annals of the Institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992.
- [144] G. Lorden *et al.*, “Procedures for reacting to a change in distribution,” *The Annals of Mathematical Statistics*, vol. 42, no. 6, pp. 1897–1908, 1971.

- [145] T. Hofmann, “Probabilistic latent semantic indexing,” in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 50–57.
- [146] M. Khan, S. Mohamed, B. Marlin, and K. Murphy, “A stick-breaking likelihood for categorical data analysis with latent gaussian models,” in *Artificial Intelligence and Statistics*, 2012, pp. 610–618.
- [147] O. Cappé and E. Moulines, “On-line expectation–maximization algorithm for latent data models,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.
- [148] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 2013.
- [149] E. Cambria, B. Schuller, Y. Xia, and C. Havasi, “New avenues in opinion mining and sentiment analysis,” *IEEE Intelligent systems*, vol. 28, no. 2, pp. 15–21, 2013.
- [150] M. J. Wainwright, M. I. Jordan *et al.*, “Graphical models, exponential families, and variational inference,” *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

Appendix A: Copyright Permissions

9/10/2020

Rightslink® by Copyright Clearance Center



RightsLink®



Home



Help



Email Support



Sign in



Create Account



A Neural Net Framework for Accumulative Feature-based Matrix Completion

Conference Proceedings: 2018 International Joint Conference on Neural Networks (IJCNN)

Author: Mehmet Aktukmak

Publisher: IEEE

Date: July 2018

Copyright © 2018, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

© 2020 Copyright - All Rights Reserved | Copyright Clearance Center, Inc. | Privacy statement | Terms and Conditions
Comments? We would like to hear from you. E-mail us at customer@copyright.com

<https://s100.copyright.com/AppDispatchServlet#formTop>

1/1

The permission above is for the use of materials in Chapter 2.



Elsevier Science & Technology Journals - License Terms and Conditions

This is a License Agreement between Mehmet Aktukmak ("You") and Elsevier Science & Technology Journals ("Publisher") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier Science & Technology Journals, and the CCC terms and conditions.

All payments must be made in full to CCC.

Order Date	10-Sep-2020	Type of Use	Republish in a thesis/dissertation
Order license ID	1062188-1	Publisher	ELSEVIER BV
ISSN	0925-2312	Portion	Chapter/article

LICENSED CONTENT

Publication Title	Neurocomputing	Rightsholder	Elsevier Science & Technology Journals
Article Title	A probabilistic framework to incorporate mixed-data type features: Matrix factorization with multimodal side information	Publication Type	Journal
		Start Page	164
		End Page	175
		Volume	367
Date	01/01/1989		
Language	Dutch		
Country	Netherlands		

REQUEST DETAILS

Portion Type	Chapter/article	Rights Requested	Main product
Page range(s)	164-175	Distribution	Worldwide
Total number of pages	11	Translation	Original language of publication
Format (select all that apply)	Electronic	Copies for the disabled?	No
Who will republish the content?	Academic institution	Minor editing privileges?	Yes
Duration of Use	Life of current edition	Incidental promotional use?	Yes
Lifetime Unit Quantity	Up to 499	Currency	USD

NEW WORK DETAILS

Title	Heterogeneous Data Fusion and Attack Detection in Recommender Systems	Institution name	University of South Florida
		Expected presentation date	2020-10-30
Instructor name	Mehmet Aktukmak		

ADDITIONAL DETAILS

The permission above is for the use of materials in Chapter 3.



ACM (Association for Computing Machinery) - License Terms and Conditions

This is a License Agreement between Mehmet Aktukmak ("You") and ACM (Association for Computing Machinery) ("Publisher") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by ACM (Association for Computing Machinery), and the CCC terms and conditions.

All payments must be made in full to CCC.

Order Date	21-Sep-2020	Type of Use	Republish in a thesis/dissertation
Order license ID	1064603-1	Publisher Portion	ACM, Inc. Chapter/article
ISBN-13	978-1-4503-6243-6		

LICENSED CONTENT

Publication Title	Proceedings of the 13th ACM Conference on Recommender System	Country	United States of America
Date	01/01/2019	Rightsholder	ACM (Association for Computing Machinery)
Language	English	Publication Type	Conference Proceeding

REQUEST DETAILS

Portion Type	Chapter/article	Rights Requested	Main product
Page range(s)	1-4	Distribution	Worldwide
Total number of pages	4	Translation	Original language of publication
Format (select all that apply)	Electronic	Copies for the disabled?	No
Who will republish the content?	Academic institution	Minor editing privileges?	Yes
Duration of Use	Life of current edition	Incidental promotional use?	Yes
Lifetime Unit Quantity	Up to 499	Currency	USD

NEW WORK DETAILS

Title	Heterogeneous Data Fusion and Attack Detection in Recommender Systems	Institution name	University of South Florida
Instructor name	Mehmet Aktukmak	Expected presentation date	2020-10-30

ADDITIONAL DETAILS

Order reference number	N/A	The requesting person / organization to appear on the license	Mehmet Aktukmak
------------------------	-----	---	-----------------

REUSE CONTENT DETAILS

9/23/2020 <https://marketplace.copyright.com/rs-ui-web/mp/license/56ab63e9-6d75-4fc0-8a68-01a013c44c44/b20117c3-5b7f-420f-be21-ad4d49d9d...>

Title, description or numeric reference of the portion(s)	N/A	Title of the article/chapter the portion is from	Quick and accurate attack detection in recommender systems through user attributes
Editor of portion(s)	N/A	Author of portion(s)	Mehmet Aktukmak
Volume of serial or monograph	N/A	Issue, if republishing an article from a serial	N/A
Page or page range of portion	348-352	Publication date of portion	2019-01-01

PUBLISHER SPECIAL TERMS AND CONDITIONS

CREDIT LINE: Mehmet Aktukmak, Yasin Yilmaz, and Ismail Uysal. 2019. Quick and accurate attack detection in recommender systems through user attributes. In Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19). Association for Computing Machinery, New York, NY, USA, 348–352. DOI:<https://doi.org/10.1145/3298689.3347050>

CCC Reproduction Terms and Conditions

1. Description of Service; Defined Terms. This Reproduction License enables the User to obtain licenses for reproduction of one or more copyrighted works as described in detail on the relevant Order Confirmation (the "Work(s)"). Copyright Clearance Center, Inc. ("CCC") grants licenses through the Service on behalf of the rightsholder identified on the Order Confirmation (the "Rightsholder"). "Reproduction", as used herein, generally means the inclusion of a Work, in whole or in part, in a new work or works, also as described on the Order Confirmation. "User", as used herein, means the person or entity making such reproduction.
2. The terms set forth in the relevant Order Confirmation, and any terms set by the Rightsholder with respect to a particular Work, govern the terms of use of Works in connection with the Service. By using the Service, the person transacting for a reproduction license on behalf of the User represents and warrants that he/she/it (a) has been duly authorized by the User to accept, and hereby does accept, all such terms and conditions on behalf of User, and (b) shall inform User of all such terms and conditions. In the event such person is a "freelancer" or other third party independent of User and CCC, such party shall be deemed jointly a "User" for purposes of these terms and conditions. In any event, User shall be deemed to have accepted and agreed to all such terms and conditions if User reproduces the Work in any fashion.
3. Scope of License; Limitations and Obligations.
 - 3.1. All Works and all rights therein, including copyright rights, remain the sole and exclusive property of the Rightsholder. The license created by the exchange of an Order Confirmation (and/or any invoice) and payment by User of the full amount set forth on that document includes only those rights expressly set forth in the Order Confirmation and in these terms and conditions, and conveys no other rights in the Work(s) to User. All rights not expressly granted are hereby reserved.
 - 3.2. General Payment Terms: You may pay by credit card or through an account with us payable at the end of the month. If you and we agree that you may establish a standing account with CCC, then the following terms apply: Remit Payment to: Copyright Clearance Center, 29118 Network Place, Chicago, IL 60673-1291. Payments Due: Invoices are payable upon their delivery to you (or upon our notice to you that they are available to you for downloading). After 30 days, outstanding amounts will be subject to a service charge of 1-1/2% per month or, if less, the maximum rate allowed by applicable law. Unless otherwise specifically set forth in the Order Confirmation or in a separate written agreement signed by CCC, invoices are due and payable on "net 30" terms. While User may exercise the rights licensed immediately upon issuance of the Order Confirmation, the license is automatically revoked and is null and void, as if it had never been issued, if complete payment for the license is not received on a timely basis either from User directly or through a payment agent, such as a credit card company.
 - 3.3. Unless otherwise provided in the Order Confirmation, any grant of rights to User (i) is "one-time" (including

<https://marketplace.copyright.com/rs-ui-web/mp/license/56ab63e9-6d75-4fc0-8a68-01a013c44c44/b20117c3-5b7f-420f-be21-ad4d49d9d06b>

2/4

The permission above is for the use of materials in Chapter 4.

Appendix B: Closed Form Expression of Rating Likelihood

Using the fact that inner product can be written as

$$\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{v}_j = \sum_{a=1}^K u_{ia} v_{ja}$$

and defining $p(u_{ia}) = \mathcal{N}(\mu_{u,ia}, \sigma_{u,ia})$, $p(v_{ja}) = \mathcal{N}(\mu_{v,ja}, \sigma_{v,ja})$, and $p(u_{ia}v_{ja}) = \mathcal{N}(\mu_{ra}, \sigma_{ra})$, one can compute the variance of the product of two uni-variate Gaussian random variables as:

$$\sigma_{ra}^2 = (\sigma_{u,ia}^2 + \mu_{u,ia}^2)(\sigma_{v,ja}^2 + \mu_{v,ja}^2) - \mu_{u,ia}^2 \mu_{v,ja}^2$$

and the mean as:

$$\mu_{ra} = \mu_{u,ia} \mu_{v,ja}$$

Next, the expected squared error is given as

$$\begin{aligned} E[(\hat{r}_{ij} - r_{ij})^2] &= E[\hat{r}_{ij}^2] - 2E[\hat{r}_{ij}]r_{ij} + r_{ij}^2 \\ &= \sum_{a=1}^K [\sigma_{ra}^2 + \mu_{ra}^2 - 2\mu_{ra}r_{ij}] + r_{ij}^2 \end{aligned}$$

Since the precision of the ratings c is modeled as fixed unknown, the rating likelihood is equal to the negative expected squared error plus a constant term, i.e.,

$$\begin{aligned}\log p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j) &= \log \mathcal{N}(r_{ij}|\mathbf{u}_i^T \mathbf{v}_j, c^{-1}) \\ &\propto -E[(\hat{r}_{ij} - r_{ij})^2] + \text{const.}\end{aligned}\tag{B.1}$$

To this end, we use the expected squared error instead of the likelihood because the constant terms and sign do not effect the empirical distributions.

Appendix C: Variational Inference of Latent Variable Model

We resort to the deterministic approximate inference method [150] to optimize the proposed latent variable model. Particularly, denoting the exact posterior as $p^*(\mathbf{U}, \mathbf{V}|\mathcal{D})$, where $\mathbf{U} = \{\mathbf{u}_i\}_{1:L}$, $\mathbf{V} = \{\mathbf{v}_j\}_{1:J}$, and $\mathcal{D} = \{\mathbf{x}_{i,m}, \mathbf{z}_{j,l}, \mathbf{y}_{i,n}, \mathbf{w}_{j,q}, r_{ij}\}_{L,J,M,N,L,Q}$, we approximate the posterior as $q(\mathbf{U}, \mathbf{V})$ and try to minimize reverse KL-divergence as the cost function:

$$\min_q \mathbb{KL}(q(\mathbf{U}, \mathbf{V}|\mathcal{D}) || p^*(\mathbf{U}, \mathbf{V})),$$

where we use factorized approximation to the posterior as

$$q(\mathbf{U}, \mathbf{V}) = \prod_i q_i(\mathbf{u}_i) \prod_j q_j(\mathbf{v}_j),$$

and Gaussian approximation for each latent variable as

$$q_i(\mathbf{u}_i) = \mathcal{N}(\mathbf{u}_i | \mathbf{m}_{ui}, \Sigma_{ui}),$$

$$q_j(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j | \mathbf{m}_{vj}, \Sigma_{vj}).$$

We want to optimize the free parameters of the distributions $\{\mathbf{m}_{ui}, \Sigma_{ui}, \mathbf{m}_{vj}, \Sigma_{vj}\}$ to minimize the objective above. However, as the variational inference implies, instead of using the normalized posterior p^* , one can use unnormalized posterior, i.e, the joint likelihood p , to obtain an upper bound to the marginal negative likelihood as

$$L(q) = \mathbb{KL}(q || p) = \mathbb{KL}(q || p^*) - \log p(\mathcal{D}).$$

Minimizing $L(q)$ instead will lead to q to be close to normalized posterior p^* since the marginal data likelihood is constant. Since it is tractable to compute joint log likelihood, following this approach, we first give the joint log $p(\mathcal{D}, \mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{A}|\Theta)$ of the proposed model as

$$\begin{aligned}
&= \sum_{m=1}^M \left[\sum_{i=1}^I \log p(\mathbf{x}_{i,m}|\mathbf{u}_i, \mathbf{W}_m, \boldsymbol{\Sigma}_{xm}) + \log p(\mathbf{W}_m, \boldsymbol{\Sigma}_{xm}) \right] \\
&+ \sum_{l=1}^L \left[\sum_{j=1}^J \log p(z_{j,l}|\mathbf{v}_j, \mathbf{A}_l, \boldsymbol{\Sigma}_{zl}) + \log p(\mathbf{A}_l, \boldsymbol{\Sigma}_{zl}) \right] \\
&+ \sum_{n=1}^N \sum_{i=1}^I \log p(\mathbf{y}_{i,n}|\mathbf{u}_i, \mathbf{H}_n) + \sum_{q=1}^Q \sum_{j=1}^J \log p(\mathbf{w}_{j,q}|\mathbf{v}_j, \mathbf{B}_q) \\
&+ \sum_{i=1}^I \log p(\mathbf{u}_i) + \sum_{j=1}^J \log p(\mathbf{v}_j) + \sum_{i=1}^I \sum_{j=1}^J \log p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j).
\end{aligned}$$

Particularly, the terms inside the above expression are given below respectively, for the user latent variable prior $\log p(\mathbf{u}_i)$ as

$$= -\frac{K}{2} \log(2\pi) - \frac{1}{2} \log|\lambda_u^{-1} \mathbf{I}_K| - \frac{\lambda}{2} \mathbf{u}_i^T \mathbf{u}_i,$$

Gaussian conditional for real valued user attributes $\log p(\mathbf{x}_{i,m}|\mathbf{u}_i, \mathbf{W}_m, \boldsymbol{\Sigma}_{xm})$ as

$$\begin{aligned}
&= -\frac{D_m}{2} \log(2\pi) - \frac{1}{2} \log|\boldsymbol{\Sigma}_{xm}| \\
&- \frac{1}{2} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i) \boldsymbol{\Sigma}_{xm}^{-1} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i)^T,
\end{aligned}$$

Factor loading matrix prior $\log p(\mathbf{W}_m, \boldsymbol{\Sigma}_{xm})$ as

$$\begin{aligned}
&= -\frac{D_m}{2} \log(2\pi) - \frac{1}{2} \log|\boldsymbol{\Sigma}_{xm}| - \frac{1}{2} \mathbf{W}_m^T (\text{diag}(\alpha) \boldsymbol{\Sigma}_{xm}^{-1}) \mathbf{W}_m \\
&+ \sum_d (a-1) \log \boldsymbol{\Sigma}_{xm,dd}^{-1} - \boldsymbol{\Sigma}_{xm,dd}^{-1} b,
\end{aligned}$$

Categorical conditional for discrete user attributes $\log p(\mathbf{y}_{i,n} | \mathbf{u}_i, \mathbf{H}_n)$ as

$$\begin{aligned}
&= \log \frac{e^{\mathbf{y}_{i,n}^T \boldsymbol{\eta}_{C,in}}}{1 + \sum_{k=1}^{M_{u,n}} e^{\boldsymbol{\eta}_{C,in,k}}} \\
&= \mathbf{y}_{i,n}^T \boldsymbol{\eta}_{C,in} - \text{lse}(\boldsymbol{\eta}_{C,in}) \\
&\geq \mathbf{y}_{i,n}^T \boldsymbol{\eta}_{C,in} - \frac{1}{2} \boldsymbol{\eta}_{C,in}^T \mathbf{F}_{u,n} \boldsymbol{\eta}_{C,in} + \mathbf{g}_{i,n}^T \boldsymbol{\eta}_{C,in} - \mathbf{e}_{i,n} \\
&\geq \mathbf{y}_{i,n}^T \mathbf{H}_n \mathbf{u}_i + \mathbf{g}_{i,n}^T \mathbf{H}_n \mathbf{u}_i - \mathbf{e}_{i,n} - \frac{1}{2} \mathbf{H}_n \mathbf{u}_i^T \mathbf{F}_{u,n} \mathbf{H}_n \mathbf{u}_i,
\end{aligned}$$

and the rating conditional $\log p(r_{ij} | \mathbf{u}_i, \mathbf{v}_j)$ as

$$= -\frac{1}{2} \log(2\pi) - \frac{1}{2} \log c - \frac{c}{2} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j).$$

We then infer the parameters of the approximate posteriors in the E-step of the variational inference algorithm. To do that, by exploiting the completing the square approach for linear Gaussian systems, we first collect the terms that depend on $-\frac{1}{2} \mathbf{u}_i \mathbf{u}_i^T$ and sum them up as

$$\begin{aligned}
&\left[\lambda_u \mathbf{I}_K + \sum_{n=1}^N \mathbf{H}_n^T \mathbf{F}_{u,n} \mathbf{H}_n + \sum_{m=1}^M \mathbf{W}_m^T \boldsymbol{\Sigma}_{xm}^{-1} \mathbf{W}_m \right. \\
&\left. + c(E[\mathbf{V} \mathbf{O}_i \mathbf{V}^T]) \right],
\end{aligned}$$

which gives posterior covariance $\boldsymbol{\Sigma}_{ui}^{-1}$ for user i , which is also given in Eq. 4.16. Then, the terms that depend on \mathbf{u}_i are collected and summed up as

$$\left[c(E[\mathbf{V}] \mathbf{O}_i \mathbf{r}_i) + \sum_{n=1}^N \mathbf{H}_n^T (\mathbf{y}_{i,n} + \mathbf{g}_{i,n}) + \sum_{m=1}^M \mathbf{W}_m^T \boldsymbol{\Sigma}_{xm}^{-1} \mathbf{x}_{i,m} \right].$$

Multiplying this expression with posterior covariance leads to posterior mean as given in Eq. 4.17. In the M-step of the algorithm, we integrate out the latent variables \mathbf{U} and \mathbf{V} , then perform maximum likelihood estimation for $\{\mathbf{H}_n, \mathbf{B}_q, c\}$ and maximum a posteriori estimation for $\{\mathbf{W}_m, \mathbf{A}_l\}$ since the latter parameters have NIG priors. The procedure to do that is to first plug in the expectations of the user and item latent variables, i.e, for $\mathbf{u}_i \rightarrow \mathbb{E}[\mathbf{u}_i]$ and for $\mathbf{u}_i \mathbf{u}_i^T \rightarrow \mathbb{E}[\mathbf{u}_i \mathbf{u}_i^T]$, and then take the derivative of the joint log likelihood with respect to the parameter that we want to estimate in order and setting them to zero. Here, we only give the related terms with the considered parameter that are to be taken the derivative to avoid clutter. Particularly, the derivatives are given as for the factor loading matrix \mathbf{W}_m of real valued attribute m as,

$$\begin{aligned} \frac{\partial}{\partial \mathbf{W}_m} \sum_i \mathbb{E}_{q_i(\mathbf{u}_i)} & \left[-\frac{1}{2} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i) \boldsymbol{\Sigma}_{xm}^{-1} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i)^T \right. \\ & \left. - \frac{1}{2} \mathbf{W}_m^T (\text{diag}(\alpha) \boldsymbol{\Sigma}_{xm}^{-1}) \mathbf{W}_m \right], \end{aligned}$$

For the factor loading matrix \mathbf{H}_n of categorical valued attribute n as,

$$\frac{\partial}{\partial \mathbf{H}_n} \sum_i \mathbb{E}_{q_i(\mathbf{u}_i)} [\mathbf{y}_{i,n}^T \mathbf{H}_n \mathbf{u}_i + \mathbf{g}_{i,n}^T \mathbf{H}_n \mathbf{u}_i - \frac{1}{2} \mathbf{H}_n \mathbf{u}_i^T \mathbf{F}_{un} \mathbf{H}_n \mathbf{u}_i]$$

For the noise covariance matrix $\boldsymbol{\Sigma}_{xm}$ of real valued attribute m as,

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\Sigma}_{xm,dd}} \sum_i \mathbb{E}_{q_i(\mathbf{u}_i)} & \left[-\frac{1}{2} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i) \boldsymbol{\Sigma}_{xm}^{-1} (\mathbf{x}_{i,m} - \mathbf{W}_m \mathbf{u}_i) \right. \\ & \left. - \frac{1}{2} \log |\boldsymbol{\Sigma}_{xm}| - \frac{1}{2} \mathbf{W}_m^T (\text{diag}(\alpha) \boldsymbol{\Sigma}_{xm}^{-1}) \mathbf{W}_m \right. \\ & \left. + \sum_d (a-1) \log \boldsymbol{\Sigma}_{xm,dd}^{-1} - \boldsymbol{\Sigma}_{xm,dd}^{-1} b \right] \end{aligned}$$

and finally for precision c of the rating latent variables as

$$\frac{\partial}{\partial c} \sum_{i,j} \mathbb{E}_{q_i(\mathbf{u}_i), q_j(\mathbf{v}_j)} \left[-\frac{1}{2} \log c - \frac{c}{2} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j) \right].$$

About the Author

Mehmet Aktukmak received the B.S. degree in electrical and electronics engineering from Hacettepe University, Ankara, Turkey in 2009, the M.S. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey in 2012. From 2009 to 2017, he worked in a military technology company in Turkey as senior digital design engineer. He has been currently working towards the Ph.D. degree in electrical engineering at University of South Florida, Tampa, FL, USA since 2017. His research interests include real-time image/video processing, matrix completion, data fusion, variational inference, meta learning, and recommender systems.