Sequential Attack Detection in Recommender Systems

Mehmet Aktukmak, *Student Member, IEEE,* Yasin Yilmaz, *Senior Member, IEEE,* and Ismail Uysal, *Member, IEEE*

***Abstract*—Recommender systems are widely used in electronic commerce, social media and online streaming services to provide personalized recommendations to the users by exploiting past ratings and interactions. This paper 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. The proposed framework consists of a latent variable model, which 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 mixed data-type attributes such as age and gender. The representations are then exploited to generate univariate statistics to be efficiently used in a CUSUM-like sequential detection algorithm that can quickly detect persistent attacks while maintaining low false alarm rates. We apply our proposed framework to three different real-world datasets and exhibit superior performance in comparison to the existing baseline algorithms for both attack profile and sequential detection. Furthermore, we demonstrate robustness to different attack strategies and configurations.**

***Index Terms*—Recommender systems, cyber-attack detection, quickest detection, latent variable model, variational inference.**

I. 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 [2]. 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].

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], [4]. 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. The anomalies can then be detected through these features [4]. Many supervised learning approaches try to classify the test users based on their rating profiles [5]. In this case, the assumption is the availability of the rating profiles of both 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 performance of such algorithms degrades quickly. To remedy this, we only consider the semi-supervised attack detection in our study which assumes no a priori information about the attack type.

The main source of information for collaborative filtering is the rating history of user profiles. Recently, researchers have begun looking into incorporating side information, including the user attributes, item features, network structure, social friend/trust network, etc. to improve the recommendation quality [6]. Significant improvements in recommendation accuracy have been achieved by using such additional information specifically for cold start scenarios [7]. 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 from side information sources. Up to now, the proposed attack detection algorithms considered solely the ratings and did not focus on the side information. Intuitively, the evidence of anomaly coming from the ratings can be aggregated with the evidence of anomaly coming from the side information.

In this paper, we consider the user attributes as an additional source of evidence for anomaly detection[1]. This approach is quite promising since, for the attackers, accessing the attributes of genuine users registered in a system requires much higher in-depth knowledge or sophisticated attack capabilities than accessing ratings. In this study, we assume the attacker creates fake profiles by overlooking the compatibility between the ratings and attributes of 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 cold start user/items, we now consider exploiting the side information to improve attack detection performance by detecting any mismatch between the attributes and ratings of fake users.

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

[1]Our preliminary results were presented in [8].

vulnerable to false alarms in such scenarios [9]. On the other hand, if the system observes persistent outliers in a short time interval, there is a good probability that indicates a real attack. In this study, 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 classical CUSUM sequential change detection algorithm, which accumulates log-likelihood ratios (LLR) [10], [11], instead of which we use a novel likelihood-based statistic. Sequential attack detection provides two advantages over sample-by-sample profile detection: (i) lowering the false alarm rate by not labeling the non-persistent outlier users as attackers, and (ii) increasing the true positive rate by accumulating the small evidences when the attack is hard to detect but still persistent.

Sequential change detection methods suit well to quickly detecting anomalies while satisfying a desired false alarm constraint. However, the classical sequential change detection algorithms, such as CUSUM, require the knowledge of the probability distributions of both genuine and anomalous data [10], [11]. In recommender systems, it can be assumed that the genuine data is available to the operator to fit a probability distribution. However, the vectors associated with the rating profiles of users are very high-dimensional and sparse. They also exhibit complex interactions with the vectors of other users which makes it very hard or otherwise intractable to model such data by using high dimensional multivariate probability distributions. The case is even worse for the fake profiles. There are various attack types, which means it is practically very difficult to model all kinds of anomalies with some parametric multivariate distributions. In fact, one can safely assume the anomalous distribution is totally unknown. To overcome these problems, we propose to extract a univariate statistic given the high dimensional sparse rating vectors and the side information of both the users and the items. Moreover, through single dimensional statistics, we aim to lower the computational complexity of the detection algorithm for efficient implementation in time-sensitive online settings.

As shown in Fig. 1, our proposed detection system consists of a data fusion module, which combines the ratings with user and item attributes, and a sequential decision making module for timely detection while controlling the false alarm rate. After training, the proposed algorithm is capable of processing the sequentially arriving data in real-time with computationally efficient updates.

For efficient attack detection in recommender systems, the main challenges for obtaining useful univariate statistics are threefold: the statistics should (i) inherently include the compatibility between the ratings and the attributes, (ii) be sufficiently informative to distinguish between the genuine and attack users, and (iii) be easy to compute for quick evaluation in online settings. To address these challenges, we first propose, in the first module shown in Fig. 1, a latent variable model to embed the observed data of genuine users composed of sparse rating vectors and mixed data-type user and item attributes. For flexibility, we consider two data types for the attributes: the numerical and categorical valued, since these data types are the most common to represent the

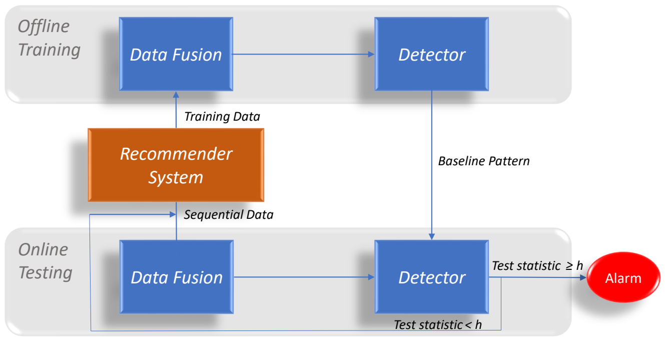


Fig. 1: General structure of the proposed attack detection framework

attributes a user or an item can exhibit (such as age and gender for the users, and year and genre for the items). The obtained embeddings lie in a lower dimensional sub-manifold, and it exhibits a unified latent representation of the compatibility between the attributes and the ratings. Then, in the second module (Fig. 1), the trained latent variable model is used to measure such compatibility for the sequentially received test entries in an online fashion by producing univariate statistics which would deviate from the nominal values in case of an attack. 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 for each observed data entry in an online setting.

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

- We present a latent variable model to fuse 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 genuine users.
- We propose a univariate statistic to distinguish between the attackers and the genuine users. The statistic is easily computed in an online fashion through the trained latent variable model parameters.
- We propose a novel sequential detection algorithm for recommender systems by exploiting the univariate statistic computed under the genuine latent space.
- We demonstrate performance improvements over both sample-by-sample attack profile detection algorithms and sequential detection algorithms that can only use the observed ratings through comprehensive experiments on three popular real world datasets.

II. Related Work

The literature includes many examples for attack types that can be performed on recommender systems [3], [4]. Subsequently, a range of attack detection algorithms have been introduced to mitigate such attacks [5]. We discuss the popular attack types before briefly explaining the detection algorithms by categorizing into static and temporal models.

A. Attack Types

Researchers have developed a range of attack prototypes to demonstrate and study the effects of the attack profile injection. To form a realistic profile, attackers do not only give ratings for the target items, but they also select some filler items and fill them as well. The attack types basically differ from each other by the selection criteria of 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 [12]. Average attack used item specific mean values instead of the global mean by exploiting more in-depth knowledge [13]. Bandwagon [12] and popular item [14] attacks were proposed to decrease the amount of knowledge required for the average attack while having a similar effect by using the most popular (liked or rated) items as filler. Some attack types [15], [16] require specific knowledge of the user ratings. While they are not easy to implement in practice, they can be used as target performance bounds. Attack types specialized for nuking items were proposed in [17]. 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 method was proposed in [18]. This method similarly uses the most popular items as filler by modifying the average attack to hide the malicious activity. [19] 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 [2]. A recent paper introduced adversarial algorithms to learn the detection strategy which increases the effectiveness of attack types [20].

B. *Static Detection Algorithms*

Among the attack detection algorithms, it is common to ignore the temporal relations between both the users and ratings. In this scenario, the observed data consists of observed genuine and attack user ratings without any temporal dependency. The algorithms of this type generally form distinctive features by using the ratings to distinguish the attack profiles from the genuine ones. By assuming similarity between the influential [21] and attack users, a set of features was proposed early in the field [22]. These features were extended to include the unusual number of ratings as an anomaly indicator in [23] and the sum of squared deviations of the rating from the mean values in [24]. In general, the features were then used in either unsupervised clustering (mostly k-means) [21]–[24] or supervised classification (kNN, SVM) [25], [26] for final decision. PCA algorithm was used in [27] to classify highly correlated user profiles as potential attack users by exploiting the principal components of the user covariance matrix. A few components associated with the small eigenvalues were used as features. A Neyman-Pearson statistical test based on LLRs was proposed to identify attack types by exploiting the item selection preferences of the users [19]. A similar algorithm was based on a latent variable model by examining the entropy of the rating distributions [28]. A probabilistic model based on beta distribution was proposed in [29]. Graph-based algorithms [30]–[32] were also used to identify the most highly correlated groups as attackers. It should be noted that all the algorithms discussed in this subsection are examples of batch anomaly detection which require the entire data for evaluation with no direct extension for online settings.

C. *Temporal Detection Algorithms*

To take into account temporal relations between the ratings, [33] used the features associated with span, frequency and mount properties. Instead of hand-crafted features, [34] developed a hidden Markov model to learn temporal rating behavior and identify the attack users with hierarchical clustering. In [35], 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 [9]. SVM-based algorithms use kernel mapping to determine the decision region within nominal data [36], [37]. However, it is not always straightforward to choose and compute kernel functions for the defined problem. The sliding window based Nearest Neighbor (NN) method in [38] generates a graph after each observation to compute a NN statistic, which makes it computationally costly. The quickest detection framework [10], [11] requires exact knowledge of the data distribution before and after the change. It uses LLRs to detect change time as quickly as possible by controlling the false alarm rates, which makes it suitable to the problem at hand if one can overcome the issue of inferring parametric distributions for the anomalous and genuine data. For closely related application of attack detection in reputation systems, [39] 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. [40] improved this approach by assuming categorically distributed ratings and proposing a Generalized Likelihood Ratio (GLR) algorithm by replacing unknown attack distribution with the one estimated from online data by using maximum likelihood estimation (MLE).

III. Proposed Framework

In this section, we provide the details of our framework to detect sequential attacks in recommender systems. First, we explain the proposed latent variable model for fusing mixed data-type ratings and side information, as well as variational expectation-maximization (EM) algorithm to infer the model parameters from training data. Then, we explain how to compute an anomaly statistic using the trained model, and to sequentially detect persistent attacks in an online fashion.

A. *Latent Variable Model for Data Fusion*

We assume that the observed data consists of mixed data-type attributes for each user, mixed data-type attributes for each item, and numerical valued ratings. The users are indexed by $i \in \{1, \dots, I\}$, where I is the total number of users. The items are indexed by $j \in \{1, \dots, J\}$, where 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 representation of the model is shown in Fig.2. 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 [41]. Subsequently, zero mean spherical Gaussian priors are assigned as

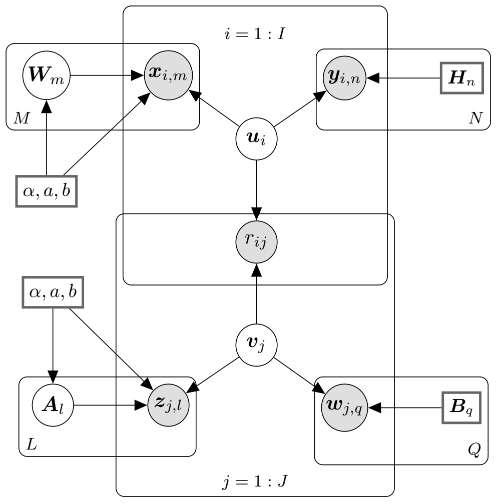


Fig. 2: Graphical model representation of the linear latent variable model. The upper part is for the users, the lower part is for the items, and the intersection is for the ratings. $\boldsymbol{x}_{i,m}$ and $\boldsymbol{z}_{j,l}$ denote real-valued attributes while $\boldsymbol{y}_{i,n}$ and $\boldsymbol{w}_{j,q}$ denote categorical attributes.

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

where $\boldsymbol{u}_i \in \mathbb{R}^K$ 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 following linear Gaussian model:

$$p(\boldsymbol{x}_{i,m} | \boldsymbol{u}_i, \boldsymbol{W}_m, \boldsymbol{\Sigma}_{xm}) = \mathcal{N}(\boldsymbol{x}_{i,m} | \boldsymbol{W}_m \boldsymbol{u}_i, \boldsymbol{\Sigma}_{xm}). \tag{2}$$

Here, the mean of the mth real-valued observation $\boldsymbol{x}_{i,m} \in \mathbb{R}^{D_m}$ is a linear function of the latent variable $\boldsymbol{u}_i$ as in factor analysis (FA) models [42], 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. $\boldsymbol{W}_m \in \mathbb{R}^{D_m \times K}$ 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 overfitting, especially when the number of observations is small and/or there are missing observations [43]. We treat them as unknown and define random variables to regularize the problem by assigning conjugate priors. Since $\boldsymbol{W}_m$ 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 [43]:

$$\begin{aligned} p(\boldsymbol{W}_{m,d}, \sigma^2_{xm,d}) =& \mathcal{N}(\boldsymbol{W}_{m,d} | 0, \sigma^2_{xm,d} \alpha^{-1} \boldsymbol{I}_{D_m}) \\ & IG(\sigma^2_{xm,d} | a, b), \end{aligned} \tag{3}$$

where a and b corresponds to the shape and scale parameters of the distribution of each diagonal element $\sigma^2_{xm,d}$ of $\boldsymbol{\Sigma}_{xm}$, where $d \in \{1, \dots, D_m\}$, respectively. A zero mean prior is assigned for the dth component of factor loading matrix $\boldsymbol{W}_{m,d}$, where α denotes how strong the belief for this prior is. We model the categorical-valued user attributes $\boldsymbol{y}_{i,n} \in \{1, \dots, M_n + 1\}$ by using a linear softmax model:

$$p(\boldsymbol{y}_{i,n} | \boldsymbol{u}_i, \boldsymbol{H}_n) = Cat(\boldsymbol{y}_{i,n} | \mathcal{S}(\boldsymbol{H}_n \boldsymbol{u}_i)), \tag{4}$$

where $n \in \{1, \dots, N\}$ and N is the total number of categorical attributes. $\boldsymbol{H}_n \in \mathbb{R}^{M_n \times K}$ 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 $\boldsymbol{W}_m$, this parameter is treated as fixed by following [44] 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 [45] or by absorbing them into the factor loading matrices [44]. 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 [46]–[48], we use dot product Gaussian conditional distribution for each observed rating since it is simple, reasonable and extensively used:

$$p(r_{ij} | \boldsymbol{u}_i, \boldsymbol{v}_j) = \mathcal{N}(r_{ij} | \boldsymbol{u}_i^T \boldsymbol{v}_j, c^{-1}), \tag{5}$$

where c corresponds to the confidence parameter for the ratings. It can be treated as fixed [45] and optimized by using MLE or as hyper-parameter [46] 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. One can easily obtain the item side expressions for $\{\boldsymbol{v}_j, \boldsymbol{z}_{j,l}, \boldsymbol{A}_{l,d}, \boldsymbol{\Sigma}_{zl}, \boldsymbol{w}_{j,q}\}$ by exploiting Fig. 2 and user side expressions. We next show how to infer the latent variables $\{\boldsymbol{u}_i, \boldsymbol{v}_j\, \boldsymbol{W}_m, \boldsymbol{A}_l, \boldsymbol{\Sigma}_{xm}, \boldsymbol{\Sigma}_{zl}\}$ and the model parameters $\{\boldsymbol{H}_n, \boldsymbol{B}_q, c\}$ that maximize the likelihood given the training data $\{r_{ij}, \boldsymbol{x}_{i,m}, \boldsymbol{y}_{i,n}, \boldsymbol{z}_{j,l}, \boldsymbol{w}_{j,q}\}$.

B. Inferring Latent Variable Model

During model inference, we are interested in retaining the uncertainties for the user and item latent variables $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ as much as possible since they are central to the model (i.e., they model both the ratings and the user/item attributes as shown in Fig. 2). Hence, we compute their complete posterior distributions. We perform maximum a posteriori (MAP) estimation for other latent variables $\{\boldsymbol{W}_m, \boldsymbol{A}_l, \boldsymbol{\Sigma}_{xm}, \boldsymbol{\Sigma}_{zl}\}$. We seek MLE solution for fixed model parameters $\{\boldsymbol{H}_n, \boldsymbol{B}_q, c\}$, and cross-validation is performed for hyperparameters $\{\lambda, \alpha, a, b\}$. EM is a powerful tool for inferring latent variable models like ours with posterior inference and fixed point estimation [43]. In the E-step of EM, the posterior distributions of latent variables are inferred and the sufficient statistics for parameter estimation are obtained. In the M-step, MLE/MAP estimations are computed with the sufficient statistics from the E-step.

In this section, we only show the derivations for the user part of the model to avoid redundancy. 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.

1) Approximation

The joint probability of the model includes two different distribution factors; Gaussian and categorical. When exact inference is not tractable, it is common to use approximations to perform variational inference [43]. The user/item latent variables, $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, have Gaussian priors and most of the features, ratings and real-valued attributes, are modeled with Gaussian conditional distributions. Thus, it is reasonable to seek Gaussian posteriors due to Bernstein-von Mises theorem [49] since the number of Gaussian variables is much larger than the number of categorical variables. To perform Gaussian approximation, we approximate the categorical likelihoods with a quadratic lower bound. Specifically, we employ the Bohning bound [50], which has been used in [44] and [51] to provide a lower bound for the LogSumExp (LSE) function in the categorical likelihoods. After the bound is applied, the log likelihood of the nth categorical user attribute takes the following form:

$$\begin{aligned}\log p(\boldsymbol{y}_{i,n}|\boldsymbol{u}_i) \geq \boldsymbol{y}_{i,n}^T\boldsymbol{H}_n\boldsymbol{u}_i - \frac{1}{2}\boldsymbol{u}_i^T\boldsymbol{H}_n^T\boldsymbol{F}_{u,n}\boldsymbol{H}_n\boldsymbol{u}_i \\ + \boldsymbol{g}_{i,n}^T\boldsymbol{H}_n\boldsymbol{u}_i - \boldsymbol{e}_{i,n}.\end{aligned} \tag{6}$$

There are three parameters emerging from the bound:

$$\boldsymbol{F}_{u,n} = \frac{1}{2}(\boldsymbol{I}_{M_n} - \frac{1}{M_n+1}\boldsymbol{1}_{M_n}\boldsymbol{1}_{M_n}^T), \tag{7}$$

$$\boldsymbol{g}_{i,n} = \boldsymbol{F}_{u,n}\boldsymbol{\psi}_{i,n} - \mathcal{S}(\boldsymbol{\psi}_{i,n}), \tag{8}$$

$$\boldsymbol{e}_{i,n} = \frac{1}{2}\boldsymbol{\psi}_{i,n}^T\boldsymbol{F}_{u,n}\boldsymbol{\psi}_{i,n} - \mathcal{S}(\boldsymbol{\psi}_{i,n})^T\boldsymbol{\psi}_{i,n} + \text{lse}(\boldsymbol{\psi}_{i,n}), \tag{9}$$

where $\boldsymbol{g}_{i,n}$ and $\boldsymbol{e}_{i,n}$ 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. The LSE function is given by $\text{lse}(\boldsymbol{\psi}_{i,n}) = \log(1 + \sum_{k=1}^{M_u}\exp^{\boldsymbol{\psi}_{i,n,k}})$.

2) E-step

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

$$q(\boldsymbol{u}_i|\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{r}_i, \boldsymbol{V}) = \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{m}_{ui}, \boldsymbol{\Sigma}_{ui}), \tag{10}$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_J]$ is the matrix of item latent variables. After updating the user latent variables, the item latent variables are updated in the same way by fixing the user latent variables. In short, two successive E-steps are performed for the users and the items. Update equations are derived by following the procedure for linear Gaussian systems [43] (See Appendix A for details of the inference). The covariance matrix of each user is computed as:

$$\begin{aligned}\boldsymbol{\Sigma}_{ui} =&(\lambda_u\boldsymbol{I}_K + \sum_{n=1}^{N}\boldsymbol{H}_n^T\boldsymbol{F}_{u,n}\boldsymbol{H}_n + \sum_{m=1}^{M}\boldsymbol{W}_m^T\boldsymbol{\Sigma}_{xm}^{-1}\boldsymbol{W}_m \\ &+ c(E[\boldsymbol{V}\boldsymbol{O}_i\boldsymbol{V}^T]))^{-1},\end{aligned} \tag{11}$$

and the mean vector is computed with following expression:

$$\begin{aligned}\boldsymbol{m}_{ui} =&\boldsymbol{\Sigma}_{ui}(c(E[\boldsymbol{V}]\boldsymbol{O}_i\boldsymbol{r}_i) + \sum_{n=1}^{N}\boldsymbol{H}_n^T(\boldsymbol{y}_{i,n} + \boldsymbol{g}_{i,n}) \\ &+ \sum_{m=1}^{M}\boldsymbol{W}_m^T\boldsymbol{\Sigma}_{xm}^{-1}\boldsymbol{x}_{i,m}),\end{aligned} \tag{12}$$

where $\boldsymbol{O}_i \in \mathbb{R}^{J\times J}$ is the matrix whose elements are binary indicators of which items user i rated. Lastly, free variational parameter $\boldsymbol{\psi}_{i,n}$ is updated, $\boldsymbol{\psi}_{i,n} = \boldsymbol{H}_n\boldsymbol{m}_{ui}$, until convergence.

3) M-step

In the M-step, we compute MAP estimators for the latent variables $\{\boldsymbol{W}_m, \boldsymbol{A}_l, \boldsymbol{\Sigma}_{xm}, \boldsymbol{\Sigma}_{zl}\}$ and MLE for the model parameters $\{\boldsymbol{H}_n, \boldsymbol{B}_q, c\}$. By setting the gradient of complete data log-likelihood with respect to $\boldsymbol{W}_m$ to zero, we obtain the MAP estimator as:

$$\boldsymbol{W}_m = \Big[\sum_i \boldsymbol{x}_{i,m}E[\boldsymbol{u}_i]^T\Big]\Big[\alpha\boldsymbol{I}_K + \sum_i E[\boldsymbol{u}_i\boldsymbol{u}_i^T]\Big]^{-1}. \tag{13}$$

Here, the sum over the second moment of $\boldsymbol{u}_i$ and the sum over the outer product of the real-valued user attribute and the mean vector are the sufficient statistics calculated in advance in the E-step. α is the regularization constant coming from the NIG prior. For the factor loading matrix $\boldsymbol{H}_n$ of the categorical attributes, the form is slightly different due to the MLE and applied quadratic bound:

$$\boldsymbol{H}_n = \Big[\sum_i \boldsymbol{F}_{u,n}^{-1}(\boldsymbol{y}_{i,n} + \boldsymbol{g}_{i,n})E[\boldsymbol{u}_i]^T\Big]\Big[\sum_i E[\boldsymbol{u}_i\boldsymbol{u}_i^T]\Big]^{-1}. \tag{14}$$

Although the sum over the second moment statistic is the same with $\boldsymbol{W}_m$, the part involving mean changes. The categorical observations $\boldsymbol{y}_{i,n}$ are linearly transformed to real-valued pseudo observations through the intermediate parameters given by Eq. (7 - 9). MAP estimation for the noise covariance matrix of the real-valued user attributes is given as:

$$\begin{aligned}\boldsymbol{\Sigma}_{xm} =&\text{diag}\Big\{\frac{1}{I+2(a+1)}\big[2b + \sum_k \boldsymbol{W}_{m,:,k}^2\alpha + \sum_i \boldsymbol{x}_{i,m}\boldsymbol{x}_{i,m}^T \\ &- 2\boldsymbol{x}_{i,m}E[\boldsymbol{u}_i]^T\boldsymbol{W}_m^T + \boldsymbol{W}_mE[\boldsymbol{u}_i\boldsymbol{u}_i^T]\boldsymbol{W}_m^T\big]\Big\},\end{aligned} \tag{15}$$

where $\boldsymbol{W}_m$ is updated in advance as in FA models [42]. Lastly, the confidence parameter for the observed ratings, whose indices are kept in set Ω in tuples, is updated as follows:

$$c = \frac{1}{|\Omega|}\sum_{i,j\in\Omega}E[(r_{ij} - \boldsymbol{u}_i^T\boldsymbol{v}_j)^2]. \tag{16}$$

The updates are iterated (E-step for the users $\rightarrow$ E-step for the items $\rightarrow$ M-step) until the model likelihood or the model variables converge. The inferred latent variable model provides a multi-modal probability distribution for the genuine data. We next explain how to detect attacks in an online fashion by computing an anomaly statistic using the likelihood of test data under the genuine probability distribution.

C. Attack Detection

Proposed 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 $\Theta = \{\boldsymbol{W}_m, \boldsymbol{A}_l, \boldsymbol{\Sigma}_{xm}, \boldsymbol{\Sigma}_{zl}, \boldsymbol{H}_n, \boldsymbol{B}_q, c\}$ to detect anomalies within the test users registered to the system.

1) Attack Statistic

We propose to extract an attack statistic for each test user by exploiting the trained latent variable model. We assume the registration to the recommender system requires the users to enter their attributes. A registered user can then assign ratings for different items in the system. Subsequently, we evaluate the likelihood of ratings and attributes for each new user under the proposed latent variable model. 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, we have the following latent variable:

$$\boldsymbol{u}_t \sim p(\boldsymbol{u}_t | \boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,M}, \boldsymbol{y}_{t,1}, \ldots, \boldsymbol{y}_{t,N}, \Theta, q(\boldsymbol{V})). \quad (17)$$

The posterior distribution is conditioned on the observed user attributes, the latent space parameter set Θ, and the variational distribution of the item latent variables $q(\boldsymbol{V})$. The item latent variables are inferred during EM training and kept fixed afterwards. Fortunately, it is easy to compute the posterior of $\boldsymbol{u}_t$ for a new user as it corresponds to one E-step for the user t using the equations (11) and (12). After the inference, the rating likelihood is computed through the dot-product modeling by $p(r_{tj}|\boldsymbol{u}_t, \boldsymbol{v}_j) = \mathcal{N}(r_{tj}|\boldsymbol{u}_t^T \boldsymbol{v}_j, c^{-1})$. Since $\boldsymbol{u}_t$ and $\boldsymbol{v}_j$ are random variables, one should take their randomness into account instead of using only the inner product of the mean values. We compute a closed form expression for the expected log-likelihood, as shown in Appendix B, to evaluate how likely the test data is under the learned nominal model. Since we eventually infer whether there is an anomaly in the system, we do not need the exact rating likelihoods, but only need an accurate comparison of the likelihood of a test instance with the nominal likelihoods. Since c is fixed, the log-likelihood is proportional to $-(r_{tj} - \boldsymbol{u}_t^T \boldsymbol{v}_j)^2$, which is a random variable. We compute its expectation using the first and second moment of $\boldsymbol{u}_t^T \boldsymbol{v}_j$ as follows:

$$\ell_{tj} = \sum_{a=1}^{K} [\sigma_{ra}^2 + \mu_{ra}^2 - 2\mu_{ra} r_{tj}] + r_{tj}^2 \quad (18)$$

The details are given in Appendix B.

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

User-based Approach: In this approach, the attack statistic is sequentially updated for each test user over time, i.e., user indices become time indices. Specifically, d_t denotes the anomaly score of user t:

$$d_t = \frac{1}{|\Omega_t|} \sum_{j \in \Omega_t} \ell_{tj}, \quad (19)$$

where Ω_t is the set of items the user t has rated. Since ℓ_{tj} is proportional to the negative log-likelihood, the anomaly score for user t, d_t, which is the average ℓtj over the items rated by user t, is expected to produce high values for the attack users and low values for the genuine users. Note that the user latent variables are conditioned on the user attributes, hence d_t will be high whenever the attributes of a test user are not compatible with his ratings profile with respect to the genuine latent space parameters. This property of the anomaly score d_t satisfies the necessary condition as a formal and informative metric to distinguish between the attack and genuine users by exploiting the compatibility between the attributes and the ratings. The anomaly score can also be easily computed as it simply requires averaging over ℓ_{tj}, which has a closed form expression as shown in (18).

To have a baseline for anomaly scores to compare against in the online test phase, in the offline phase of the algorithm, we first partition the training set into two sets, train the latent variable model using the first set, and evaluate and store d_i for each genuine user in the second set. Note that the anomaly score of a test user does not have an absolute meaning; it can be only understood relative to the scores of genuine (nominal) users. Then, to compare the anomaly score of each test user with the nominal baseline from training, we compute the tail probability (p-value) of each d_t in the online test phase:

$$p_t = \frac{1}{I} \sum_{i=1}^{I} 1_{\{d_i > d_t\}}, \quad (20)$$

which is simply the fraction of the genuine users whose scores are higher than d_t. To convert this probability measure into a real-valued statistic we next compare the computed p-value with a statistical significance level such as the commonly used 0.05 in the log scale:

$$s_t = \log \frac{\alpha}{p_t}. \quad (21)$$

Item-based Approach: In this approach, the ratings of test users are observed on the item basis. 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 negative log-likelihood term ℓ_{tj} for each item is directly used as an anomaly score, i.e., $d_{tj} = \ell_{tj}$. Contrary to the user-based approach, we only consider the ratings of the users who rated item j to evaluate the p-value as:

$$p_{tj} = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} 1_{\{d_{ij} \geq d_{tj}\}} \quad (22)$$

where Ω_j is the set of the genuine users who rated item j assuming the test user t rated the item j. Next, anomaly score s_{tj} for each item user t rates is computed as follows:

$$s_{tj} = \log \frac{\alpha}{p_{tj}}. \quad (23)$$

Note that here we have a separate time index tj for each item j, which only proceeds when a user rates it, whereas in the user-based approach, there is only one time detection task with time index t.

2) Sequential Detector

One can decide if a user or an entry is anomalous by observing only the attack statistics s_t or s_{tj}, which is representative of single-instance outlier detection. However, this p-value test is known to be prone to high false alarm rates [52] due to the existence of non-persistent anomalies including some genuine users rating new items in unexpected ways. Since attacks typically last for some duration, they correspond to persistent anomalies. To avoid the false alarms for non-persistent anomalies, considering the sequential nature of attacks, we propose a CUSUM-like sequential attack detection framework based on the derived attack statistic. The aim is to detect the attacks quickly while keeping the false alarm rate at an acceptable level. The proposed user-based sequential detector accumulates the attack statistic as follows:

$$g_t = \max\{g_{t-1} + s_t, 0\}, \quad g_0 = 0. \tag{24}$$

If this sequential statistic exceeds a predefined threshold h, we raise a system-wide alarm at time $t_a = \min\{t : g_t \geq h\}$. The threshold h is chosen to strike a balance between quick detection and low false alarm rate. Higher h values enable lower false alarm rates at the expense of longer detection delays, and vice versa for lower h values. The complete algorithm for the user-based sequential attack detector is given in Algorithm 1 [2]. The algorithm for the item-based detector follows the same lines with the following differences. In the item-based approach, a separate detector is set up for each item in the system:

$$g_{t,j} = \max\{g_{t-1,j} + s_{tj}, 0\}, \quad g_{0,j} = 0. \tag{25}$$

If any of the sequential statistics exceeds the threshold, a system-wide alarm is set at time $t_a = \min\{t : \max_j g_{t,j} \geq h\}$.

3) User Classification

After an 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 sequential statistic was zero before the alarm time, i.e., $t_b = \max\{t < t_a : g_t = 0\}$. The ending time of the attack corresponds to the first time when the sequential 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. Then individual classification is performed based on the attack statistics of these users such that if $s_t > 0$, the test user t is flagged as an attacker.

IV. Experimental Study

In this section, we first provide the details of the test environment including the attack generation procedure, the dataset descriptions, and brief explanation of the baseline algorithms. We then present experimental results to assess the robustness of the proposed framework.

[2]The code can be accessed from https://github.com/maktukmak/sequential-attack-detector

Algorithm 1: Sequential attack detection in recommender systems (User-based)

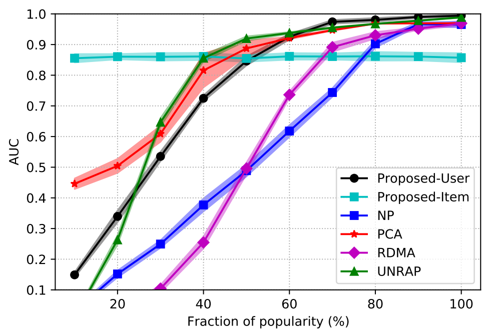
```
Offline Phase;
Inputs: $\boldsymbol{x}_{i,m}$, $\boldsymbol{y}_{i,n}$, $r_{ij}$, $\boldsymbol{z}_{j,l}$, $\boldsymbol{w}_{j,q}$;
for $iter_{train} := 1$ to ($iter_{max}$ or conv) do  Train loop
    for $iter_{var} := 1$ to (conv) do  Variational loop
        Infer user posterior ($\boldsymbol{m}_{\boldsymbol{u}i}$, $\boldsymbol{\Sigma}_{\boldsymbol{u}i}$);
        Update variational parameters $\boldsymbol{\psi}_{i,n}$, $\boldsymbol{e}_{i,n}$, $\boldsymbol{g}_{i,n}$;
    end
    for $iter_{var} := 1$ to (conv) do  Variational loop
        Infer item posterior ($\boldsymbol{m}_{\boldsymbol{v}j}$, $\boldsymbol{\Sigma}_{\boldsymbol{v}j}$);
        Update variational parameters $\boldsymbol{\psi}_{j,q}$, $\boldsymbol{e}_{j,q}$, $\boldsymbol{g}_{j,q}$;
    end
    MAP for global variables $\boldsymbol{W}_m$, $\boldsymbol{A}_l$, $\boldsymbol{\Sigma}_{\boldsymbol{x}m}$, $\boldsymbol{\Sigma}_{\boldsymbol{z}l}$;
    MLE for global variables $\boldsymbol{H}_n$, $\boldsymbol{B}_q$, $c$;
end
for $i := 1$ to $I$ do  Geniune anomaly score loop
    Compute score $d_i$;
end
Outputs: $\{d_i\}$;
Online Phase;
Inputs: $\boldsymbol{x}_{t,m}$, $\boldsymbol{y}_{t,n}$, $r_{tj}$, $\{d_i\}$;
for $t := 1$ to $T$ do  Test loop
    Infer user posterior ($\boldsymbol{m}_{\boldsymbol{u}t}$, $\boldsymbol{\Sigma}_{\boldsymbol{u}t}$);
    Compute score $d_t$, tail probability $p_t$, statistic $s_t$;
    Update $g_t$;
    if $g_t \geq h$ then
        Raise alarm at $t_a = t$;
        break;
    end
end
```

A. Test Environment

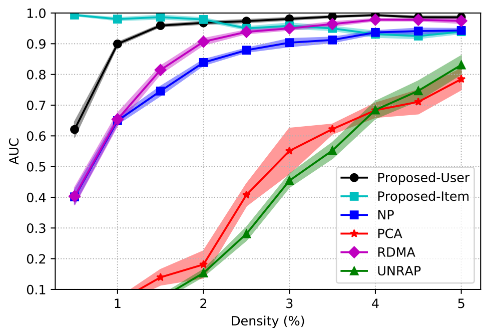
1) Attack Generation

As discussed in Sec. II-A, different mechanisms have been used in the literature to generate artificial attack profiles to show the vulnerability of the existing recommender systems. Generating an artificial attack profile requires filling the rating vector intelligently. Since 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 some filler items and rate them realistically to hide the activity and increase the impact of attack. The fraction of the number of filler items to the total number of items in the system, is called the filler size. Moreover, 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.

To generate a single attack profile, we utilize several popular attack types in the literature which differ by item selection, and ratings given for the filler items and target item (see Sec. II-A). We can subsequently group them into three categories as push, nuke and obfuscated attacks. Push attacks are mainly designed to create a positive prediction shift for a target item by entering high ratings. In the experiments, we consider random, average, and bandwagon attacks [12], [13]. Nuke attacks are conversely designed to create a negative prediction shift. Here, we implement reverse bandwagon and love-hate



(a) Obfuscation sensitivity when the AOP attack is injected



(b) Density sensitivity when the push attack is injected

Fig. 3: Characteristics of the two variants of the proposed algorithm with respect to the varying obfuscation and sparsity.

attacks for this category [17]. Lastly, the obfuscated attacks are used to hide the intention from the detection algorithms by changing the filler selection criteria and the ratings given for them. Popular and average over popular (AOP) attacks are selected as representatives of this category [18].

The attributes of the attack profiles are chosen randomly from the pool of genuine user attributes. For each attribute of an attack profile, a random genuine user is selected and the corresponding attribute is copied. After all attributes are generated, a final check is performed to avoid unrealistic combinations. For example, in the MovieLens dataset, regardless of the gender, the following occupations are avoided for the age group 0-20: academic/educator, executive/managerial, lawyer, retired, scientist, tradesman/craftsman. Similarly, for the age group 20-60, K-12 student is not used as occupation.

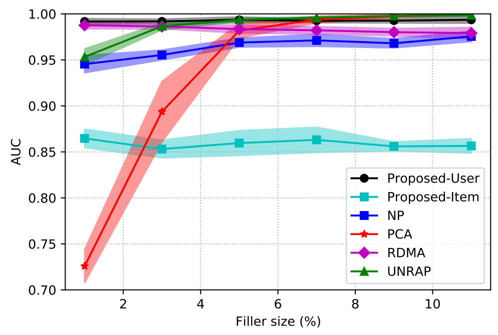
Another requirement to get quick and efficient results is that the attack has to be performed in a short time period. We name this type of attack as sequential attack. To generate a sequential attack, we generate a number of attack profiles. These profiles can be generated from either one of the single attack types or by mixing several of them randomly. In a real-world scenario, some genuine users may also interleave the attack sequence. To simulate this scenario, we hold 10% of randomly selected genuine users from the dataset and train the model with the rest of the genuine users. A test sequence is formed by mixing held-out genuine users (90%) and attack profiles (10%).

2) *Datasets*

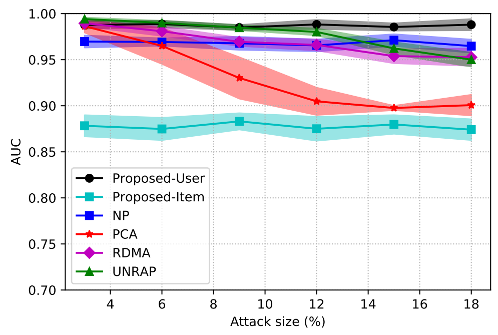
We consider three benchmark datasets, MovieLens (1M), BookCrossing and LastFM, which vary in both dataset size and concept (movies, books, music), to perform a robustness assessment of the proposed framework. The MovieLens dataset provides 1M ratings of approximately 4K users on 6K movies. All the users and the movies have mixed data-type attributes. Each user has age, gender and occupation information. We model the age as one dimensional real-valued attribute, the gender as binary categorical attribute, and the occupation as 21-class categorical attribute. Each item has the release year and genre information. We consider the release year as one-dimensional real-valued attribute. The genre information is provided as 19-dimensional binary vector where each genre is represented by one bit. Since it is not one-hot encoded, we model each bit as 2-class categorical attribute. BookCrossing dataset provides 1.1M ratings of 278K users on 270K books. Not all users have fully observed attributes thus we filter out the users with missing attributes. The remaining users have age and location information. We model the age as one-dimensional real-valued attribute, and the location as 8-class categorical attribute as we only consider the English-speaking countries. We filter the dataset to eliminate the imbalance of the countries by sampling the users evenly based on their respective countries, and to decrease the sparsity of the rating vectors of the users by pruning out the cold users and books which have very 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 filter out the users with missing attributes. The remaining users have age, gender and country information. We model the age as one-dimensional real-valued attribute, the gender as 2-class categorical attribute, and the country as 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 user rating vectors to eliminate the cold users and play counts with low number of ratings. The resulting dataset consists of 1.4M ratings of 26K users on 10K artists. 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 of 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.

3) *Baseline algorithms*

For performance comparison on sample-by-sample attack profile detection task, we consider four baseline algorithms. The first algorithm is a statistical test proposed in [18] called NP-test, which is based on LLR. The distribution of the genuine profiles and attack profiles are obtained based on 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 performs the classification. The second algorithm uses PCA to find out a metric associated with user correlations [27]. 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



(a) Push attack with varying filler items



(b) Push attack with varying attack profiles

Fig. 4: Average AUC scores when mixed push is performed with different attack configurations to the MovieLens dataset.

components, and the top users are flagged as attack profiles. The third approach RDMA [22] 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. The last algorithm called UnRAP [24] 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. As a summary, we consider the NP algorithm as a statistical test based on the LLR, PCA algorithm as a latent variable model that performs deterministic inference for the latent variables, i.e., principal components, and the RDMDA (2005) and UnRAP (2008) as two hand-crafted feature-based methods. These algorithms can cover a broad range of approaches to form the baselines. For performance comparison on sequential attack detection task, we consider two baseline algorithms based on CUSUM framework and originally proposed for the reputation systems. We call these algorithms as Mean-detector [39] and GLR [40]. 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. We pick these two algorithms since there has been no study that analyzes sequential detection performance in the recommender systems domain, and the reputation systems are a closely related application domain. Nonetheless, these algorithms can detect distribution changes by observing rating sequence of a particular item, hence easily applicable to the problem at hand.

4) *Metrics*

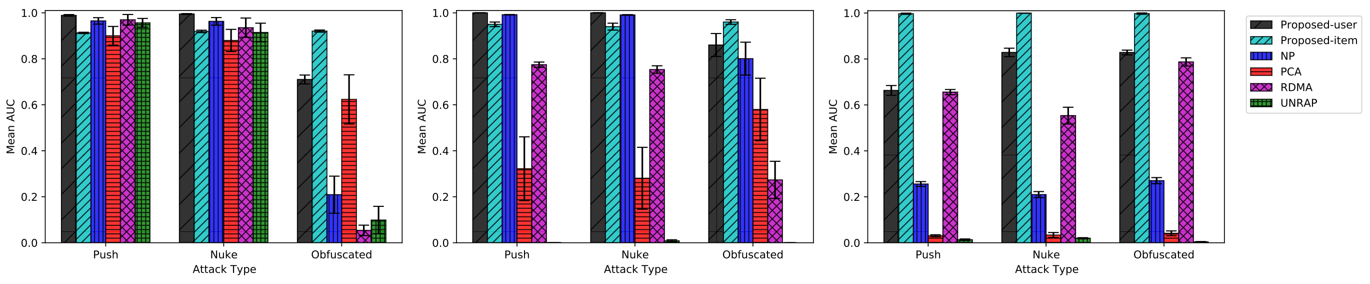
We use two metrics to assess the performance of the proposed framework and compare it with the baseline algorithms. For sample-by-sample profile detection, the problem can be regarded as binary classification, and one can use Area Under the Curve (AUC) metric to measure the classification error of the profiles. 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 thresholds. For the optimal sequential detection scheme, the overall criterion is to minimize average detection delay while satisfying a lower bound on the false alarm period [10]. 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 MDD is given as the average running length of the detector after the attack occurs. The lower values are better since we expect to detect the attack and raise an alarm as soon as it starts. False alarm period is the average period that the first false alarm occurs, i.e., the average time that the detection procedure stops when the past sequence is nominal. It is computed as the reciprocal of the average false alarm rate at a fixed detector threshold h. As a result, this plot demonstrates the trade-off for choosing the threshold. A lower threshold decreases the detection delay but could increase the false alarm rate, and vice versa.

B. *Robustness Assessment*

In this section, we first evaluate 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.

1) *Obfuscation and sparsity level*

To show the sensitivity of the variants with respect to the obfuscation level, we pick the AOP attack and change the fraction of popularity, i.e., the portion of the item set sorted based on popularity, while fixing the attack size as %5 and filler size as the dataset average, which is the average number of items genuine users in the system have rated. At each fraction, 100 different realizations of the attack profiles are generated. In each realization, the target item and the held out users are randomly selected among the pool of the items and the users in the dataset. Fig. 3a 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 a test user, 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, the nominal likelihoods of popular filler items suppresses the single target



(a) Movielens (b) LastFM (c) BookCrossing

Fig. 5: 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. Proposed algorithms provide stable and accurate results among different conditions.

item, and thus the user-based method tends to produce nominal anomaly scores, which reduces its detection performance. Note that the performances of other baseline algorithms also suffer when obfuscation exists. Indeed, the only method that exhibits stable performance throughout the range is the item-based method.

To show the sensitivity of the variants with respect to the dataset sparsity, we conduct experiments by varying the sparsity (conversely the density) of the MovieLens dataset by holding out randomly selected rating entries while keeping the filler and attack size configuration. In this setting, we design push attacks by evenly mixing random, average, and bandwagon attack types while generating fake profiles. The observation from Fig. 3b is that although both of the proposed variants provide reasonable performances among different sparsity conditions, the user-based method slightly degrades at high sparsity and the item-based method slightly degrades at low sparsity. The key reason is that rating prediction performance of the latent variable model defined in Section III-A relies on the number of observed ratings 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 noisy likelihoods results in more noisy anomaly scores than using only a single likelihood as in the case of the item-based method. On the other hand, the other baseline algorithms exhibits more significant performance degradation than the proposed methods when sparsity increases.

2) Attack configuration

We now pick the user-based method and compare it with the baseline algorithms to assess the robustness by changing the attack configuration, i.e, filler size and attack size. The filler size is a parameter for the attacker that is adjusted based on the purpose and the knowledge at hand. Hence, the attack detection performance should be robust to this possibility of varying fractions of filler items. We design an experiment to show the performances of the sample-by-sample attack profile detection algorithms with respect to varying fractions of filler items. We repeat 100 experiments by generating mixed push attacks, i.e., evenly distributed mixtures of random, average, and bandwagon attack types, of size 5% with a 10% hold-out set and running the algorithms on Movielens. We used a uniformly distributed average filler size in the range of 1% to 11% to place the average filler size of the dataset

TABLE I: Standardized partial AUC values bounded at %0.1 and %1 FPR. Note that under .50 means no detection at that rate.

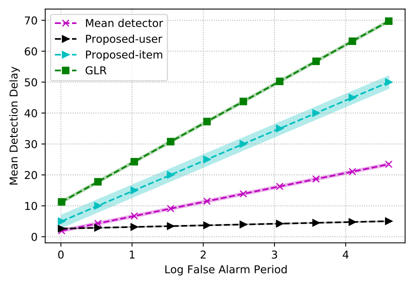
	0.1%			1%		
	Push	**Nuke**	**Obf.**	**Push**	**Nuke**	**Obf.**
Movielens						
User-based	.68±.08	**.79±.05**	.62±.04	**.94±.02**	**.97±.01**	.75±.01
Item-based	.59±.09	.50±.01	**.74±.01**	.91±.01	.89±.03	**.90±.02**
NP	.50±.00	.50±.00	.50±.00	.82±.07	.81±.08	.47±.00
PCA	**.77±.16**	.76±.01	.50±.00	.84±.15	.81±.11	.47±.00
RDMA	.56±.08	.52±.00	.50±.04	.84±.12	.71±.15	.47±.00
UNRAP	**.76±.10**	.75±.09	.50±.00	.90±.08	.85±.11	.47±.00
LastFM						
User-based	**.80±.02**	**.83±.01**	.62±.03	**.98±.01**	**.99±.01**	.75±.02
Item-based	.61±.05	.63±.06	**.68±.03**	.87±.04	.85±.04	**.88±.01**
NP	.69±.02	.67±.02	.58±.04	.95±.01	.95±.01	.84±.01
PCA	.50±.00	.50±.00	.50±.00	.47±.00	.47±.00	.47±.00
RDMA	.63±.01	.72±.02	.50±.00	.74±.01	.77±.01	.47±.00
UNRAP	.50±.00	.50±.00	.50±.00	.47±.00	.47±.00	.47±.00
Book						
User-based	.50±.00	.50±.00	.56±.04	.48±.00	.48±.00	.71±.01
Item-based	**.84±.07**	**.86±.01**	**.87±.08**	**.98±.01**	**.99±.00**	**.99±.01**
NP	.50±.00	.50±.00	.50±.00	.48±.00	.48±.00	.47±.00
PCA	.50±.00	.50±.00	.50±.00	.47±.00	.47±.00	.47±.00
RDMA	.50±.00	.50±.00	.50±.00	.48±.00	.47±.00	.50±.01
UNRAP	.50±.00	.50±.00	.50±.00	.47±.00	.47±.00	.47±.00

($\sim$6%) in the middle of the distribution. Fig. 4a shows the average AUC values of the algorithms in this setup. The results indicate that the proposed user-based algorithm is robust to the variations in filler size while the baseline algorithms have varying or lower performances. Item-based method shows stable performance over the range although it is slightly lower than the performances of other algorithms in this setting.

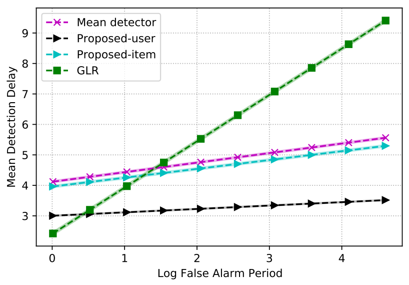
Similar to the filler size, the attack detection algorithm should be robust to varying attack sizes. To assess their performances, we perform an experiment where we change the attack size fraction from 3% to 18% while fixing the hold-out set at 10% and evaluate mean AUC in the case of a mixed push attack with a filler size of 6%. Fig. 4b shows the average results of 100 experiments. Similar to the experiments for the filler size, the results demonstrate the robustness of the user-based method for different attack sizes, and lower but stable performance of item-based method.

C. Attack detection performance

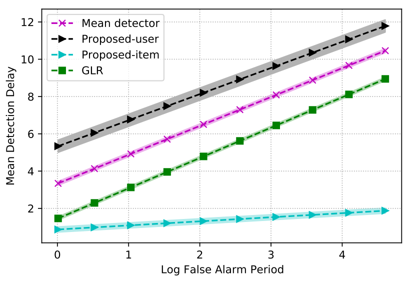
This section presents the overall performance evaluation in classification of the test user profiles as attacker or genuine.



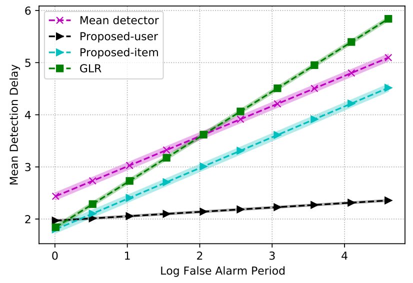
(a) Movielens-Push Attack



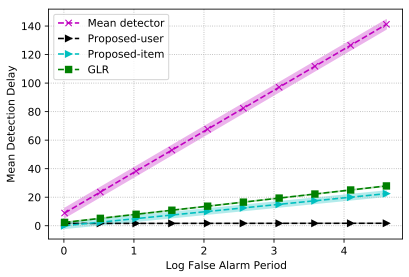
(b) Movielens-Nuke Attack



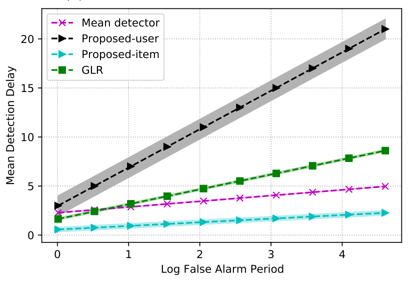
(c) Movielens-Obfuscated attack



(d) LastFM-Push Attack



(e) LastFM-Nuke Attack



(f) LastFM-Obfuscated Attack

Fig. 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.

To get the average performances, we integrate out the attack and filler size 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 each dataset. For the filler size, the average filler size of the dataset is first computed as $\bar{f}_s$, and then 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. We conduct experiments by using three different attack type settings such as mixed push (random, average, bandwagon), mixed nuke (reverse bandwagon, love-hate), and mixed obfuscation (popular, average over popular). The mixing is performed randomly with even probabilities for each profile. The resulting mean AUCs are presented in Fig. 5 for each dataset and each mixed attack type separately. We also report standardized partial AUC values at bounded false positive rates 1% and 0.1% in Table I. The results indicate that the two variants of the proposed framework provide consistently and exchangeably superior performance over other algorithms for all test conditions and datasets. The overall results are summarized as follows:

- The competing algorithms fail especially when the sparsity is very high as in BookCrossing dataset. PCA algorithm fails because the filler size is very low in the case of sparse datasets, which increases the correlation between all possible user profiles. NP fails as the histogram of item ratings gets broader, i.e., the distribution lacks 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. Item-based method gives the best performance in this low sparse setting.
- The obfuscated attack drastically degrades the performances of the baseline algorithms. For both Movielens and LastFM datasets, the item-based method provides most acceptable performance for this attack type. PCA fails because the correlation between the genuine and attack user rating vectors becomes higher in the obfuscated attack. 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 NP algorithm performs reasonably well on LastFM dataset.
- In the rest of the cases, in which push and nuke attacks are injected to MovieLens and LastFM datasets, all algorithms detects fairly well. We can observe that user-based method performs better than the rest of the algorithms. Note that the detection rate of baseline algorithms significantly reduces at bounded false positive rates.

TABLE II: Test setup

Dataset	**Attack**	**Pre-distribution**	**Post-distribution**
MovieLens	P	[.06 .16 .39 .31 .07]	[.05 .10 .10 .35 .45]
	N	[.01 .02 .19 .41 .37]	[.45 .35 .10 .10 .05]
	O	[.32 .19 .30 .16 .04]	[.05 .10 .10 .35 .45]
LastFM	P	[.85 .13 .02 .01 .01]	[.05 .05 .30 .30 .30]
	N	[.18 .45 .09 .18 .09]	[.45 .35 .10 .10 .05]
	O	[.54 .32 .07 .05 .02]	[.05 .05 .30 .30 .30]

D. Sequential detection performance

We now focus on the performance of 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 detect the changes in distribution parameters. To infer the parameters by using MLE, a long rating sequence for each item is necessary to get reasonable performance. Unfortunately, due to high sparsity of its rating matrix, BookCrossing dataset does not provide a sufficient number of ratings for almost any of the items it contains. To this end, we exclude it and use MovieLens and LastFM datasets to achieve a fair comparison. The test conditions are designed to provide distribution changes before and after the attack occurs. We consider mixed push, mixed nuke and mixed obfuscated attacks, respectively. Although these attacks are designed to provide highest (5) or lowest (1) ratings to the target item, we modify the attacks to enable categorical distributions. We set the probabilities of the distributions after the attack occurs and then generate independent samples. Table II shows the parameters before and after the attack in this setup. We fix the filler size to the average sparsity of each respective dataset $\bar{f}_s$, the hold-out size as 10%, and the attack size as 5%. Fig. 6 shows the sequential detection performances over 500 repetitions as MDD versus the false alarm period plots. The results show that two variants of the proposed framework provide superior performance over the range of different distribution changes as compared to the baseline algorithms. In particular, for both datasets, when push and nuke attacks are applied, the user-based method performs quicker detection than other two algorithms to ensure fixed false alarm rates. We also observe that Mean-detector performs slightly better than GLR in most conditions. In the case of obfuscation, the item-based method is robust and provides better performance than other 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 variety of scenarios.

V. Conclusion

In this paper, we address the problem of sequential attack detection in recommender systems. The proposed semi-supervised framework requires 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 attack 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 clear 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 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.

Appendix A
Variational Inference of Latent Variable Model

We resort to deterministic approximate inference method [53] to optimize the proposed latent variable model. Particularly, denoting the exact posterior as $p^*(\boldsymbol{U},\boldsymbol{V}|\mathcal{D})$, where $\boldsymbol{U}=\{\boldsymbol{u}_i\}_{1:I}$, $\boldsymbol{V}=\{\boldsymbol{v}_j\}_{1:J}$ and $\mathcal{D}=\{\boldsymbol{x}_{i,m},\boldsymbol{z}_{j,l},\boldsymbol{y}_{i,n},\boldsymbol{w}_{j,q},r_{ij}\}$, we approximate the posterior as $q(\boldsymbol{U},\boldsymbol{V})$, and try to minimize reverse KL-divergence, $\min_q \mathbb{KL}(q(\boldsymbol{U},\boldsymbol{V}|\mathcal{D}) \,||\, p^*(\boldsymbol{U},\boldsymbol{V}))$, as the cost function while using the mean-field factorization approximation to the posterior given as

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

and Gaussian approximation for each latent variable given as $q_i(\boldsymbol{u}_i)=\mathcal{N}(\boldsymbol{u}_i|\boldsymbol{m}_{ui},\boldsymbol{\Sigma}_{ui})$ and $q_j(\boldsymbol{v}_j)=\mathcal{N}(\boldsymbol{v}_j|\boldsymbol{m}_{vj},\boldsymbol{\Sigma}_{vj})$. We want to optimize the free parameters of the distributions $\{\boldsymbol{m}_{ui},\boldsymbol{\Sigma}_{ui},\boldsymbol{m}_{vj},\boldsymbol{\Sigma}_{vj}\}$. 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 negative log marginal likelihood as

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

Minimizing $L(q)$ leads q to be close to normalized posterior p^* since the log marginal data likelihood is constant. It is tractable to compute the log joint likelihood $\log p(\mathcal{D},\boldsymbol{U},\boldsymbol{V},\boldsymbol{W},\boldsymbol{A}|\Theta)$ of the proposed model as follows:

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

Particularly, the terms in above expression excluding constants are given below, for user latent variable prior $\log p(\boldsymbol{u}_i)$ as

$$=-\frac{1}{2}\log|\lambda_u^{-1}\boldsymbol{I}_K|-\frac{\lambda_u}{2}\boldsymbol{u}_i^T\boldsymbol{u}_i,$$

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

$$=-\frac{1}{2}\log|\boldsymbol{\Sigma}_{xm}|-\frac{1}{2}(\boldsymbol{x}_{i,m}-\boldsymbol{W}_m\boldsymbol{u}_i)\boldsymbol{\Sigma}_{xm}^{-1}(\boldsymbol{x}_{i,m}-\boldsymbol{W}_m\boldsymbol{u}_i)^T,$$

for factor loading matrix prior $\log p(\boldsymbol{W}_{m,d},\sigma^2_{xm,d})$ as

$$\begin{aligned}
&=-\frac{1}{2}\boldsymbol{W}_{m,d}^T(diag(\alpha)\sigma^{-2}_{xm,d})\boldsymbol{W}_{m,d}+(a-1)\log\sigma^{-2}_{xm,d}\\
&\quad-b\sigma^{-2}_{xm,d},
\end{aligned}$$

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

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

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

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

We then infer the parameters of the approximate posteriors in the E-step of the variational inference algorithm. To do that, by exploiting the "complete the square" approach for linear Gaussian systems, we first collect the terms that depend on $-\frac{1}{2}\boldsymbol{u}_i\boldsymbol{u}_i^T$ and sum them up to obtain posterior covariance $\boldsymbol{\Sigma}_{\boldsymbol{u}i}^{-1}$ for user i, which is given in Eq. (11). Then, the terms that depend on $\boldsymbol{u}_i$ are collected, summed up, and multiplied with posterior covariance to obtain mean as given in Eq. (12). In the M-step of the algorithm, we integrate out the latent variables $\boldsymbol{U}$ and $\boldsymbol{V}$, then perform maximum likelihood estimation for $\{\boldsymbol{H}_n, \boldsymbol{B}_q, c\}$ and maximum a posteriori estimation for $\{\boldsymbol{W}_m, \boldsymbol{A}_l\}$. To do that we first plug in the expectations of the user and item latent variables, i.e., for $\boldsymbol{u}_i \rightarrow \text{E}[\boldsymbol{u}_i]$ and for $\boldsymbol{u}_i\boldsymbol{u}_i^T \rightarrow \text{E}[\boldsymbol{u}_i\boldsymbol{u}_i^T]$, and then take the derivative of the joint log likelihood with respect to the parameter that we want to estimate and setting it to zero. To avoid clutter, we only give the related terms with the considered parameter. Particularly, the derivatives are given below for the factor loading matrix $\boldsymbol{W}_m$ of real-valued attribute m as

$$\frac{\partial}{\partial \boldsymbol{W}_m}\sum_i \mathbb{E}_{q_i(\boldsymbol{u}_i)}\big[-\frac{1}{2}\boldsymbol{W}_m^T(\text{diag}(\alpha)\boldsymbol{\Sigma}_{xm}^{-1})\boldsymbol{W}_m$$
$$-\frac{1}{2}(\boldsymbol{x}_{i,m} - \boldsymbol{W}_m\boldsymbol{u}_i)\boldsymbol{\Sigma}_{xm}^{-1}(\boldsymbol{x}_{i,m} - \boldsymbol{W}_m\boldsymbol{u}_i)^T\big],$$

for factor loading matrix $\boldsymbol{H}_n$ of categorical attribute n as,

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

for noise covariance matrix $\boldsymbol{\Sigma}_{xm}$ of attribute m as,

$$\frac{\partial}{\partial \sigma_{xm,d}^2}\Big[-\frac{1}{2}\log\sigma_{xm,d}^{-}\frac{1}{2}\boldsymbol{W}_{m,d}^T(diag(\alpha)\sigma_{xm,d}^{-2})\boldsymbol{W}_{m,d}$$
$$\sum_i \mathbb{E}_{q_i(\boldsymbol{u}_i)}\big[-\frac{1}{2}(\boldsymbol{x}_{i,m} - \boldsymbol{W}_m\boldsymbol{u}_i)\boldsymbol{\Sigma}_{xm}^{-1}(\boldsymbol{x}_{i,m} - \boldsymbol{W}_m\boldsymbol{u}_i)\big]$$
$$+(a-1)\log\sigma_{xm,d}^{-2} - \sigma_{xm,d}^{-2}b\Big],$$

and finally for precision c of the rating latent variables as

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

Appendix B
Closed form expression of rating likelihood

Using the fact that inner product can be written as

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

and defining $p(u_{ia}) = \mathcal{N}(\mu_{u,ia}, \sigma_{u,ia})$ and $p(v_{ja}) = \mathcal{N}(\mu_{v,ja}, \sigma_{v,ja})$, one can compute the variance of the product of two univariate 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

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

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.

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] 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.

[3] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[4] C. C. Aggarwal, "Recommender systems: The textbook," *Springer Publishing Company, Incorporated*, 2018.

[5] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: a review," *Artificial Intelligence Review*, pp. 1–29.

[6] 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.

[7] 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: https://doi.org/10.1016/j.neucom.2019.08.019

[8] ——, "Quick and accurate attack detection in recommender systems through user attributes," ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 348–352.

[9] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[10] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. prentice Hall Englewood Cliffs, 1993, vol. 104.

[11] H. V. Poor and O. Hadjiliadis, *Quickest detection*. Cambridge University Press, 2008.

[12] 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.

[13] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," *Beyond Personalization 2005*, p. 19.

[14] 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.

[15] 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*. Citeseer, 2005, pp. 13–23.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] ——, "Detection of shilling attacks in recommender systems via spectral clustering," in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] H. Chen, "Sequential change-point detection based on nearest neighbors," *Annals of Statistics*, vol. 47, no. 3, pp. 1381–1407, 2019.

[39] 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.

[40] 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.

[41] 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.

[42] Z. Ghahramani, G. E. Hinton *et al.*, "The em algorithm for mixtures of factor analyzers," Tech. Rep. CRG-TR-96-1, 1996.

[43] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[44] 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.

[45] 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.

[46] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[47] C. C. Johnson, "Logistic matrix factorization for implicit feedback data," in *Advances in neural information processing systems*, vol. 27, 2014.

[48] 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.

[49] A. W. Van der Vaart, *Asymptotic statistics*. Cambridge university press, 2000, vol. 3.

[50] D. Böhning, "Multinomial logistic regression algorithm," *Annals of the Institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992.

[51] Y. Yılmaz and A. O. Hero, "Multimodal event detection in twitter hashtag networks," *Journal of Signal Processing Systems*, vol. 90, no. 2, pp. 185–200, 2018.

[52] M. Baker, "Statisticians issue warning over misuse of p values," *Nature News*, vol. 531, no. 7593, p. 151, 2016.

[53] 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.