Recommender Systems: A Study of Cold-Start and Attack Resilience

Author:

Sulthana Shams

Supervisor:

Prof. Douglas Leith

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

in the

School of Computer Science and Statistics

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

August 2024

Dedicated to

Ummichi, Vappichi and all my Teachers

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

I, the undersigned, agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Sulthana Shams

April 5, 2024

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Sulthana Shams

April 5, 2024

Acknowledgments

It gives me immense joy to have worked with my PhD advisor, Prof. Douglas Leith, whose unwavering support and guidance have been instrumental in bringing this thesis to life. My PhD was right in the middle of the pandemic, and without your guidance, this thesis would have been impossible. I express my sincerest gratitude to you, sir. Your brilliant insights, comments and discussions have been a wealth of information I'm proud to carry with me for the rest of my life.

I'm deeply indebted to Dr Prem Kumar Karumbu, my mentor, teacher and master's thesis advisor. My time with you has left a lasting impact on me. Your words of encouragement and advice to always approach my work with honesty, and sincerity and leave the rest to God has been a great source of strength for me. Thank you, sir, for instilling in me the joy of learning, the importance of discipline and the value of hard work and persistence.

None of this would have been possible without my parents, Mr Shamsudeen PM and Ms Roshna Shamsudeen. They taught me the value of education and most importantly kindness and respect for every being. Thank you for providing me with a wonderful space to learn and grow throughout my life. You have taught me the value of the simple act of waiting. You have supported me in every way humanly possible and this journey is simply unimaginable without you both. I would also like to thank my brother Dr Azad Shams for time and again reaching out with help I never realised I needed. You made my life easier.

To my husband, Dr Ghanim Fajish, thank you very much for your company and jovial jokes which, on the days that did not annoy me, have made me laugh and light-hearted. To Mma, Ppa, Molu and Mithu, thank you for all the joy and support that you sent my way.

To my best friends, Eva Rosemary Ronald and Dr. Sandhya T.P, thank you for always being available in my life. Chechi, I'm indebted to you for your help and kind words

whenever I needed it and Eva for telling me I can do it :). Thank you for all the ego boosts whenever I was down.

My time in the Lab was wonderful due to the presence of my colleagues: Thank you Daron, Pavlos, Apostoles, Jose, Erjona, Victor, Vaibhav, David, Dilina and Mohammad for creating a wonderful environment for research, engaging conversations and games of table tennis when the lab was getting too quiet. I wish I had more time to get to know you guys (all due to the pandemic).

I am deeply appreciative of the support from Science Foundation Ireland, which has funded my scholarship. I am indebted to Dublin City Libraries. My time in Dublin would be empty without all the friends you gave me over the last 4 years.

I owe a debt of gratitude to composers Ilayaraja, AR Rahman, and Ludovico Einaudi, whose works provided solace during many sleepless nights and luminaries Dr APJ Abdul Kalam, Dr Stephen Hawking, and Dr Kalpana Chawla, whose works and life introduced me to Science and always inspired me when things were tough. A Special thanks to Albus Percival Wulfric Brian Dumbledore, whose wisdom always appeared when I needed it most.

Finally, I express my deepest gratitude to Trinity College Dublin and IIITDM for providing me with all that I need in my learning journey.

Sulthana Shams

University of Dublin, Trinity College
August 2024

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Every so often, we're required to choose between options we haven't personally explored especially when there are a potentially overwhelming number of options that a service may offer. In these scenarios, we rely on recommendations from others in our daily lives, whether through word of mouth, movie and book reviews printed in newspapers, or general polls. Recommender Systems (RS) support users in this decision-making process by aiding in the discovery of relevant content and products [Resnick and Varian, 1997].

RS is utilized in diverse sectors, spanning e-commerce, entertainment, and content platforms. Some of the popular applications are in platforms like Spotify [Pérez-Marcos and Batista, 2018], Amazon [Smith and Linden, 2017] and Netflix [Gomez-Uribe and Hunt, 2016] enhancing user engagement and delivering personalized recommendations. Netflix, for instance, employs recommendation algorithms to simplify movies and TV show discovery for users to watch. Likewise, Amazon showcases recommendations on its homepage based on user's previous purchases and browsing history. These systems streamline the process of discovering movies, music and products amidst the many options.

Recommending items through the approach of interest similarity, often referred to as Collaborative Filtering (CF) [Goldberg et al., 1992], holds great appeal in various domains such as books, CDs, and movies. CF assumes that users with similar tastes, typically measured by their ratings for certain items, will have similar preferences. However, this method does not always yield accurate results. The primary challenge arises from the inherent sparsity of data, where individuals have interacted with only a limited subset of available items in the platform [Sarwar et al., 2000].

To address this issue effectively, a promising strategy involves categorizing users into

clusters based on shared interests. By grouping people into clusters with similar preferences, we can harness the collective behaviours and preferences of users within each cluster. This approach allows for improved recommendations as users within the same cluster tend to exhibit higher degrees of similarity compared to the broader user population. This concept of user clustering in RS is not a new one. In [Ungar and Foster, 1998, Xue et al., 2005, Hofmann, 2004], a range of techniques (K-means and Gibbs sampling, a cluster-based smoothing system, and a cluster-based latent semantic model respectively) are proposed which cluster users and items in an unsupervised manner to improve the accuracy of prediction and scalability problems. For instance, Netflix has adopted this strategy by segmenting its vast user base of over 93 million individuals into approximately 2,000 clusters based on taste [Gomez-Uribe, 2016, Rodriguez, 2017].

Moreover, recent research has unveiled another dimension to user clustering in RS: the potential for enhancing user privacy [Checco et al., 2017]. Users submit ratings using group identities. Each group contains many thousands of users so it is very hard to guess which ratings were submitted by any specific user providing a strong "hiding in the crowd" type of privacy. Importantly, it also gives state-of-the-art recommendation performance i.e. there is no trade-off between accuracy and privacy here. More recently, Google's Privacy Sandbox initiative aims to address privacy concerns by categorizing users into interest groups based on their browsing behavior [Ravichandran and Vassilvitskii, 2009, Bindra, 2021]. This approach offers a privacy-preserving way for advertisers, publishers, and ad tech providers to deliver personalized ads.

Discovering user communities based on shared preferences is a trending topic in the realm of RS. In the following section, we discuss the RS Cold Start Problem and explore how user clustering can offer a solution.

1.1 User Clustering and Cold Start Problem

The effectiveness of a RS is always determined by its ability to offer recommendations that align with a user's preferences. While existing CF approaches to recommendation perform satisfactorily for already existing users in the system, it fails completely for new users, because the system has no or minimal knowledge about their preference history. This is a well-known problem called a *Cold-Start Problem* and is a popular research focus [Elahi et al., 2016, Elahi et al., 2018]. One approach to mitigate the cold start problem

is to harvest side information about the new users from their social media preferences and declared demographic data [Shi et al., 2014, Son, 2016]. However, side information is not always available. Therefore we are interested in estimating the cold user's preferences without any side information.

User clustering emerges as a powerful tool in this context. We exploit the fact that users can often be grouped into clusters based on the similarity of their preferences. This allows accelerated learning of new user preferences since the task becomes one of identifying which cluster a user belongs to. Eliciting feedback on a select sequence of items could help the RS calibrate the preferences of new users [Rashid et al., 2002, Harpale and Yang, 2008]. We introduce an online learning approach to quickly and reliably learn the preferences of new users by presenting carefully chosen items and receiving bandit feedback. Once the cluster is estimated, the system recommends items liked by members of that cluster using any CF approach.

While user clustering holds promise for RS new user recommendations, it also introduces security concerns. In the following section, we discuss the implications of data poisoning attacks within RS, particularly in the context of user clustering.

1.2 User Clustering and Data Poisoning Attacks

The vulnerability of RS to potential manipulation by malicious users was initially investigated by [O'Mahony et al., 2002, Lam and Riedl, 2004, Mobasher et al., 2005]. These attacks involve injecting false profiles to either promote or demote specific items and are called Data Poisoning/Shilling Attacks in RS. Push attacks elevate targeted items, while nuke attacks seek to demote them. Such fraudulent ratings and profiles can severely undermine the RS robustness.

In a user cluster-based RS, clusters formed around shared traits or preferences become appealing targets for attackers. These attackers can take advantage of the cluster by entering and controlling it to alter recommendations and target content to a sizable user base. The study by [Mobasher et al., 2005] was among the earliest to investigate the effects of attacks focused on a specific user segment of RS. They demonstrate that such attacks are not spread evenly across all users; they are directed towards a specific subset of the user base. This strategy of limiting the impact eliminates suspicion and reduces the chance of being discovered.

Clustering benefits have been studied in advertising as well [Epasto et al., 2021, Xie and Phoha, 2001, Geyik et al., 2015]. Platforms like Facebook, Instagram, and Amazon provide targeted advertising and information-sharing services, enabling content creators and publishers to choose user segments based on various demographic factors [1]. Naturally, RS in these commercial settings, face the risk of unethical behaviours due to strong incentives. Amazon, for instance, has faced issues with fake reviews flooding its product listings leading to their undeserved high rankings [He et al., 2021] [2]. Additionally, competitors have been known to flood negative reviews to diminish the ranking of rival products. For instance, in Amazon, organized efforts by anti-vaccine supporters have aimed to suppress pro-vaccine content [3]. Similarly, Samsung faced penalties for hiring spammers to post negative fake reviews about HTC smartphones [4].

Moreover, the utilization of user cluster-based advertising infrastructure can potentially give rise to issues related to discriminatory targeting of specific clusters. An illustrative instance of this concern lies in the exploitation of platforms like Facebook's targeted advertising system, which has been used to exclude individuals based on their race or gender when delivering advertisements related to housing or employment opportunities [Angwin and Parris Jr., 2016, Datta et al., 2014].

As a result, the advantages of clustering for privacy and advertising also raise worries about its possible role in aiding targeted attacks. In the second and third part of our research, we study the vulnerability of MF-based RS to targeted data poisoning attacks and drawing insights from our research, we propose a defence strategy aimed at mitigating the impact of data poisoning attacks on user communities.

1.3 Contributions of the Thesis

This thesis addresses two primary challenges encountered in RS: the Cold Start Problem and Robustness to Data Poisoning Attacks. While our investigation of the Cold Start Problem is not limited to MF, our study of data poisoning concentrates on the Standard MF-based recommendation technique. We choose this approach due to its scalability, capability to handle missing data, and superior prediction accuracy compared to other standard CF methods [Koren et al., 2009]. Additionally, it serves as the foundational

[1] Amazon Web Services: [Katidis and christianbonzelet, 2022], Facebook: [Meta, 2023]
[2] [Simonetti, 2022, Dwoskin and Timberg, 2018]
[3] [Guarino, 2018]
[4] [Bates, 2013]

recommendation model based on latent factors upon which subsequent enhancements have been constructed [He et al., 2017, Gao et al., 2023, Xue et al., 2017].

User preferences can be captured via either implicit (click streams, content consumption duration, browsing history etc) or explicit user feedback (user ratings, like-dislike buttons). Our research focuses on explicit user feedback, which primarily consists of user ratings on items. This form of feedback directly reveals user preferences, including both positive (high ratings) and negative (low ratings) feedback. While our current study doesn't utilize implicit feedback for gathering user preference information, it's worth noting that our work could be extended to incorporate implicit feedback in addition to explicit feedback in the future.

To summarise, we make 3 main contributions to Recommender Systems:

- **Fast Cold-Start for Recommender System New Users:**

    1. We propose a method of finding items called 'distinguisher items' whose bandit feedback provides a lot of information that helps distinguish between the groups and converge to the correct group quickly and thereby learn the preferences of the cold user.
    2. The Cluster Bandit Algorithm that exploits distinguisher items to quickly learn the correct group of a cold user with low regret
    3. Regret analysis of the proposed algorithm
    4. A performance evaluation of the proposed algorithm using real data-sets

- **Evaluating Impact of User-Cluster Targeted Attacks in MF-RS**

    1. A systematic study of data poisoning attacks targeted at a group of users and identify the factors contributing to the effectiveness of attacks in MF-based RS.
    2. An analysis of the individual effects of attacks on latent feature matrices and explore how they contribute to the propagation of targeted attacks in a MF-based RS. Studies investigating the role of latent feature matrices are new to the literature.
    3. Illustrate findings with real-world datasets and show that a simple attack strategy using limited knowledge of user preferences suffices to target a specific user group precisely

- **Attack Detection Using Item Vector Shift in MF-RS**

    1. A new approach for detecting shilling attacks that make use of item preference vectors
    2. The Item Vector Deviation (IVD) strategy provides an unsupervised and attack model-free strategy requiring limited training data to produce favourable outcomes.
    3. Demonstrate the effectiveness of the strategy using real-world datasets.

1.4 Publications

Material contained in this thesis has been published separately as follows:

1.4.1 Published

- Shams, S., Anderson, D., and Leith, D. (2021). Cluster-Based Bandits: Fast Cold-Start for Recommender System New Users, page 1613–1616. Association for Computing Machinery, New York, NY, USA
- S. Shams and D. J. Leith, "Improving Resistance of Matrix Factorization Recommenders To Data Poisoning Attacks," (2022) Cyber Research Conference - Ireland (Cyber-RCI), Galway, Ireland

1.4.2 Submitted

- Shams, S. and Leith, D. (2023). "Evaluating Impact of User-Cluster Targeted Attacks in Matrix Factorisation Recommenders". Association for Computing Machinery, Transactions on Recommender Systems
- Shams, S. and Leith, D. (2023). "Attack Detection Using Item Vector Shift in Matrix Factorisation Recommenders". Association for Computing Machinery, Transactions on Privacy and Security

1.5 Structure of the Thesis

In Chapter 2, we present a brief overview of key concepts in Recommender Systems that are relevant to the problems addressed in later chapters. We introduce the basic Cold

Start Problem and Data Poisoning Attack. We also discuss the various approaches to cold start problem and discuss the standard attack strategies and detection methods available in the literature.

In Chapter 3, we study the problem of cold start and cast it in the framework of user clustering to obtain a low regret approach to quickly and reliably learn a cold user's group (and thereby preferences).

In Chapter 4, we study targeted attacks and their impact on MF-based recommenders. We are interested in analysing the nature and extent of the changes the feature matrices undergo when fake ratings are introduced into a RS.

In Chapter 5, we re-visit the robustness problem to attacks in a MF based RS. We discuss the existing approaches to data poisoning defence available in the literature in this chapter. Based on our conclusions from the study in Chapter 4, we propose an innovative defence strategy aimed at mitigating the impact of attacks in a MF-based RS.

Finally, in Chapter 6, we wrap up the thesis by summarizing our contributions and laying out potential avenues for future research in this field.

The proofs of Theorems/Lemmas/Propositions in each chapter are provided in the Appendix at the end of the thesis.

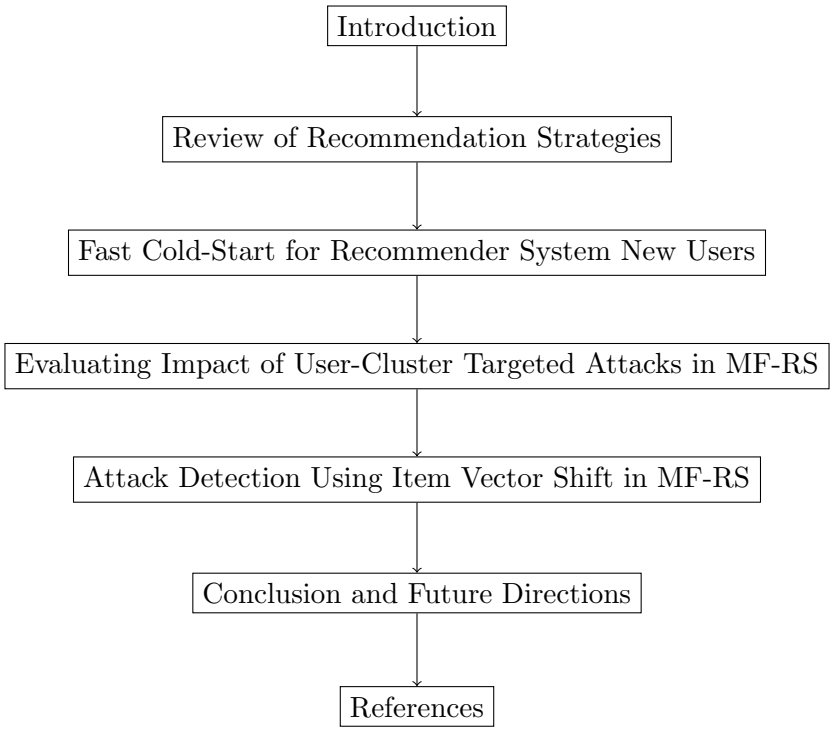


Figure 1.1: Thesis Structure

Chapter 2

Review of Recommendation Strategies

Two different approaches are widely adopted to design RS: Content-based Filtering and Collaborative Filtering. A comprehensive survey of RS algorithms can be found in [Ko et al., 2022].

The *Content-Based Filtering* model [Lops et al., 2011] is the most basic model within the overall RS model and was mainly used in early recommendation models. Content-based filtering focuses on the characteristics and attributes of items themselves. It analyzes item features such as genre, keywords, or descriptions to identify similarities and make recommendations based on a user's preferences. For example, if a user has shown a preference for action movies, a content-based RS will recommend similar action movies. Content-based strategies require gathering external information that might not be available or easy to collect.

On the other hand, *Collaborative Filtering* (CF) [Goldberg et al., 1992] utilizes the behaviour and preference interactions (Explicit and Implicit) of users i.e. it looks for patterns in user-item interactions, such as ratings or purchase history, to identify similarities between users or items. A comprehensive survey of CF algorithms can be found in the survey papers [Chen et al., 2018, Su and Khoshgoftaar, 2009].

Determining which approach is better depends on various factors. Content-based Filtering is advantageous when dealing with new users or items, as it relies solely on item characteristics and does not require historical user data. However, it may struggle to suggest items outside a user's known preferences. Collaborative Filtering, on the other

hand, can offer more diverse recommendations by leveraging the wisdom of the crowd, but it may face challenges when dealing with cold-start problems or sparse data due to its inability to address the system's new products and users. In this aspect, content filtering is superior.

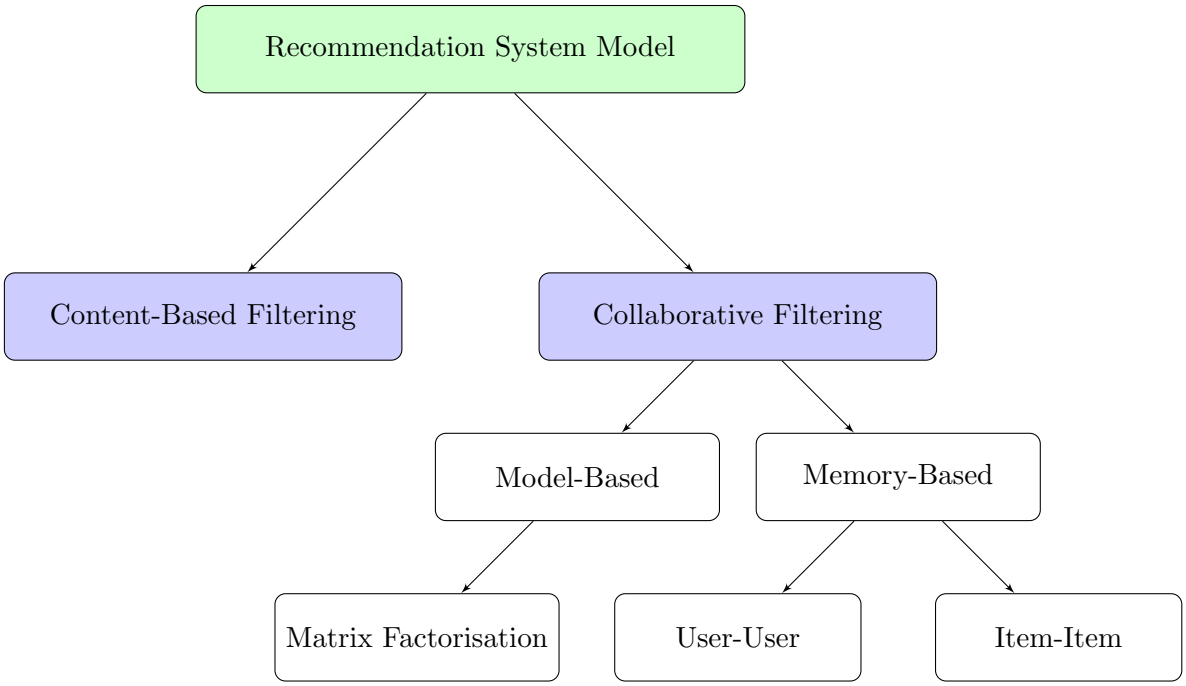


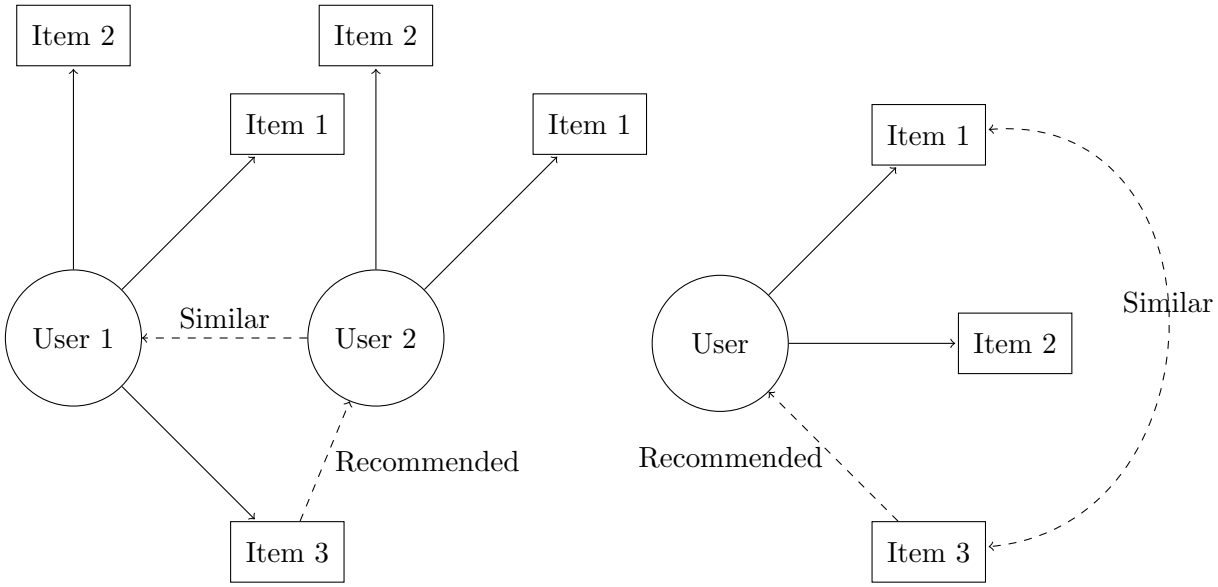
Figure 2.1: Overview of Recommendation Strategies

Traditional CF can be divided into the two methods: *memory-based and model-based methods*. Some examples of memory based are user-user and item-item approaches [Koren, 2010, Sarwar et al., 2001]. *User-User* CF recommends items based on the preferences of similar users, while *Item-Item* CF recommends items based on the preferences of users who have shown interest in similar items.

For example, consider Figure 2.2(a), user-user method identifies like-minded users (in this case user 2 has also rated item 1,2) who can complement each other's ratings. Since user 1 has not seen item 3 but which is liked by user 2, it is recommended to user 1. Similarly, in item-item method illustrated in Figure 2.2(b), we have item 1 well liked by a user. To predict a user's rating for item 3, we would look for the item 3's nearest neighbor that this user actually rated (in this case, item 1).

These memory based CF methods use the user rating data to calculate the similarity between users or items and make recommendations according to those calculated similarity values. This similarity function can take many forms, such as correlation between

ratings or measuring angle between the rating vectors. Similarity values between items are measured pairwise by observing all the users who have rated both the items, hence their accuracy depends on the availability of common rated items among the users. In practice we might not have such items. One other shortcoming of this algorithm is the scalability: It is not suitable for practical applications when dealing with large amounts of users and items because computing similarities between all pairs of users or items is expensive [Sarwar et al., 2000].



(a) User-User Collaborative Filtering

(b) Item-Item Collaborative Filtering

Figure 2.2: Memory-Based Collaborative Filtering Techniques

To improve the prediction accuracy and overcome the limitations of memory based CF methods, model-based CF approaches have been widely proposed [Hofmann, 2004]. Among these model-based CF methods, *Matrix Factorization* (MF) is state of the art method implemented for recommendation tasks. MF is a technique that has the original ratings matrix decomposed into two low-rank matrices. The factorisation aims to capture underlying latent factors that influence user-item interactions. By decomposing the user-item interaction matrix into lower-dimensional matrices, it can learn latent representations that capture user preferences and item characteristics. This lowers the effective dimension and cuts down on computational complexity. This allows for personalized recommendations based on similar users or similar items, enhancing the accuracy and relevance of the recommendations.

Recently there have been a range of improvements and expansions to the fundamental

MF. While the inner product approach, which linearly combines latent features, has its merits, it may not be sufficient to capture the complex structure of user interaction data. For instance, [He et al., 2017] explores the use of deep neural networks for learning the interaction function from data. In NCF (Neural Collaborative Filtering) user and item feature vectors are fed into a multi-layer neural architecture to map these latent vectors to prediction scores. Similarly [Xue et al., 2017] propose a novel matrix factorization model with neural network architecture as a simple nonlinear generalization of MF.

Traditionally, when updating parameters for a specific user, only the items interacted with by that user are considered. However, recent advancements incorporate Graph-based Neural Networks (GNN) [Gao et al., 2023], which aggregate user neighborhood embeddings, expanding beyond just first-order neighbors. GNN frameworks enhance MF by creating higher-quality embeddings using neighborhood information. By leveraging data from a node and its one-hop neighbors, GNNs generate context-aware representations for each node in a graph.

These enhancements build upon the foundational technique of Matrix Factorization, prompting a deeper exploration of MF.

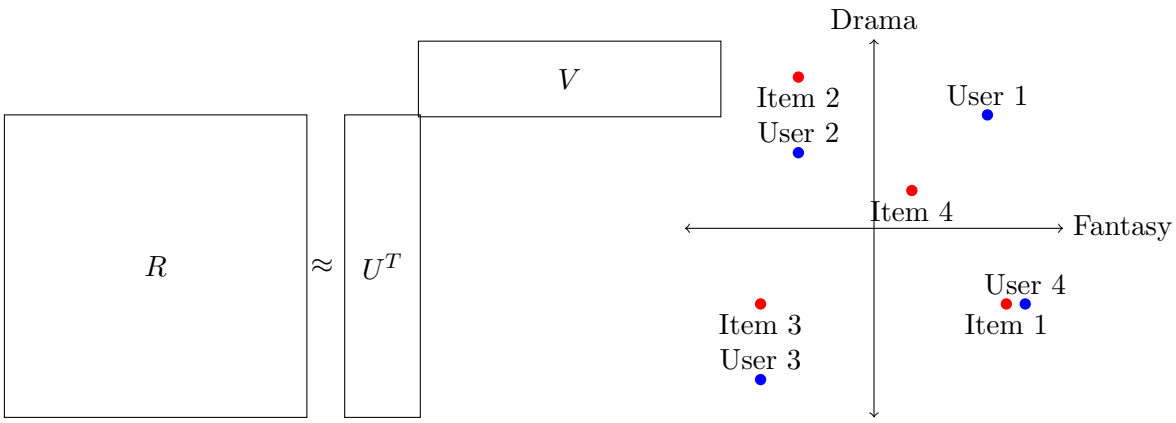
2.1 Standard Matrix Factorisation Recommenders

The Netflix Prize competition brought MF-based RS into the spotlight. Netflix used MF models to predict user ratings for movies based on historical data, enabling them to provide personalized movie recommendations to their subscribers.

MF algorithms have gained popularity in RS due to several key reasons. Firstly, these algorithms demonstrate excellent scalability, making them well-suited for handling large volumes of data. This scalability is particularly valuable in scenarios where massive datasets need to be processed efficiently.

Secondly, MF techniques are effective in handling missing values, which commonly occur in RS. Since users typically rate only a small fraction of the available items, the majority of the ratings are missing. MF algorithms can effectively estimate these missing ratings, providing accurate predictions for user-item interactions. Compared to older algorithms, MF exhibits higher prediction accuracy [Lee et al., 2012]. This enhanced accuracy contributes to improved recommendation performance, leading to more relevant and personalized suggestions for users.

2.1.1 A Basic MF Model



(a) Matrix Factorization (b) 2-Dimensional Feature Space

Figure 2.3: Matrix Factorization and Feature Space

Given a RS with n users and m items, MF works by decomposing the original sparse user-item interaction matrix $R \in \mathbb{R}^{n \times m}$ into two low-rank matrices $U \in \mathbb{R}^{d \times n}$ and $V \in \mathbb{R}^{d \times m}$ where their row dimension d is typically much less than n and m [Koren et al., 2009]. This allows us to uncover latent features and the association between users and items to these latent features in the d dimensional latent space. The two low-rank matrices U and V associate a user's inclination towards the latent features and an item's degree of membership towards those latent features respectively. Their matrix product $U^T V$ approximates R and predicts the missing entries of the original sparse matrix. We call U the user-feature matrix and V the item-feature matrix. The standard approach to obtain U, V given user-item ratings is to minimize the sum of squared error over the set of all observed ratings, i.e.

$$\min_{U,V} \sum_{(i,j) \in \mathcal{O}} (R_{i,j} - U_i^T V_j)^2 + \lambda \left(\sum_i \|U_i\|^2 + \sum_j \|V_j\|^2 \right) \tag{2.1}$$

Where $\mathcal{O}$ is the set of (user, item) rating pairs, $R_{i,j}$ is the rating of item j by user i, $U_i \in \mathbb{R}^{d \times 1}$ and $V_j \in \mathbb{R}^{d \times 1}$ are the column vectors that describe the preferences associated to user i and item j respectively; $\|.\|$ is the Euclidean norm and λ is a regularisation weight.

Thus MF characterizes both items and users by vectors of factors inferred from item rating patterns. Figure 2.3(b), illustrates this idea for a simplified example in two dimen-

sions ($d = 2$). Consider two hypothetical dimensions characterized as fantasy and drama. Figure 2.3(b) shows where several movie items and a few fictitious users might fall on these two dimensions. Items are recommended in such an MF-based RS based on the proximity of the item vectors to the user vectors in the latent space. For example, we would expect User 4 to love Item 1 and hate Item 2, and to rate Item 4 about average.

In the field of RS, several challenges contribute to the complexity of achieving good recommendation performance. Among these challenges, two particularly noteworthy ones have emerged as focal points: the cold start problem and data poisoning attacks. These challenges pose obstacles to the accurate and secure functioning of recommenders.

2.2 Cold Start Problem in Recommenders

The cold start problem arises when new users or items enter the system, and limited or no data is available to make personalized recommendations. This challenge hampers the system's ability to understand the preferences and behaviours of these new users or items, resulting in less accurate or relevant recommendations. Addressing the cold start problem is crucial to ensure a positive user experience and encourage user engagement from the early stages of system interaction.

2.2.1 Overview of Cold Start Strategies

Content/Side Information Based Cold Start Recommendation

A common strategy to mitigate the cold-start problem is to gather side information of user and items alongside the available rating data. For a survey of side information-based approaches, see [Shi et al., 2014, Son, 2016].

These approaches [Barjasteh et al., 2016, Hawashin et al., 2018, Lam et al., 2008, Odić et al., 2013, Park and Chu, 2009, Heidari et al., 2022] collect information such as user declared demographic information (eg: age, gender, marital status etc) and item features (i.e. genre, cast, manufacturer, production year etc.) to tackle cold-start problems. Such contextual data contain abundant additional information about the user interests or item features and provide an opportunity to improve the recommendation quality.

For example, [Choi, 2014] introduces a collaborative recommendation method that considers user preferences for content types alongside individual item ratings. They derive a user-content type rating matrix by averaging individual item ratings for each content

type. Consequently, when specific item ratings are missing but content type information is available, user-content type rating data can be used to predict the preferences of new users.

[Hawashin et al., 2018] groups users based on demographic factors, using hidden interests within these groups for recommendations. Similarly, [Lam et al., 2008] leverages declared demographic factors like age, gender, and occupation to model and predict ratings for new users. More recently, [Heidari et al., 2022] use deep learning models to incorporate side information in the framework of MF.

However all these approaches depend strongly on such contextual information which may not always available to exploit due to privacy concerns.

Interview Based Cold Start Recommendation

Next, we will discuss works that investigate acquiring good recommendations for a cold user without any prior side information. A line of research focuses on interview-based methods for cold-start recommendation whereby a small number of items are selected as questions, and a new user is required to give feedback to these questions. The feedback can be either rating values or opinion labels (e.g., like or dislike). A key component in this approach is how to effectively choose only a small number of questions to query users and obtain a decent recommendation performance. Several attempts to select items using information theory have been discussed in [Rashid et al., 2008, Rashid et al., 2002], including entropy, popularity and coverage of the items. In a similar tone, [Harpale and Yang, 2008] focuses on learning the cold user's preferences without the assumption that they can provide rating for any queried item. Instead of rating individual items, [Chang et al., 2015] ask users to rate groups of items by providing a presentation of tag-labeled movie clusters.

Recently decision trees [Golbandi et al., 2011, Sun et al., 2013] are also used to present items to users for feedback. In [Golbandi et al., 2011], the recommender system repeatedly asks new users to pick their preference from a pair of movies. Authors in [Sun et al., 2013], focus on building a single decision tree with each node asking multiple questions instead of a single one as in [Golbandi et al., 2011].

Representative Based Cold Start Recommendation

These works [Amatriain et al., 2009, Liu et al., 2011a, Shi et al., 2017a] investigate how to predict a new user's ratings based on a small set of user ratings. Authors in [Amatriain et al., 2009], predict the preferences of new users using an external set of expert users (movie critics online), whose ratings are weighted according to their similarity to the target user. [Liu et al., 2011a] attempts to identify a set of most representative items based on observed ratings from the existing user set rather than an independent third-party. They cluster items into a small number k. Each user is represented using k items. Hence they only need to ask a new user to rate k representative items to recommend other items.

Since the set of representatives are chosen to have a good coverage ie. a good proportion of users have rated the set of representative items, they tend to be less discriminative in characterizing fine-grained interests of a group. Building on this work, authors in [Shi et al., 2017a] propose two levels of interview, first level of interview to dynamically create meaningful user groups using decision trees and second level to identify finer interests within the group. Items are provided for the users to rate and based on the response, a decision tree assigns the user to a group.

Bandit Algorithms in Recommender System

Recent research has explored innovative approaches to address the cold start problem by framing it as a multi-armed bandit problem, as evident in studies such as [Li et al., 2010, Felício et al., 2017, Nguyen et al., 2014, Hong et al., 2020, Galozy and Nowaczyk, 2023].

For instance, [Li et al., 2010] introduced the LinUCB algorithm, which recommends articles by dynamically selecting them based on shared user and article contextual information. In a related vein, [Nguyen et al., 2014] extended LinUCB by incorporating past user ratings as contextual information, eliminating the need for user-side information.

In a different approach, [Felício et al., 2017, Hong et al., 2020] proposed a framework that combines offline-learned states or clusters with traditional UCB and Thompson sampling exploration techniques, enhancing online learning efficiency. These algorithms play the best arm given context under its corresponding latent state/cluster.

However, recent research as outlined in [Galozy and Nowaczyk, 2023], akin to our

approach involving distinguisher items, reaffirms the importance of information-gathering arms. These arms may offer lower immediate rewards but yield substantial long-term benefits in terms of enhancing state discrimination capabilities within the latent bandit problem.

2.3 Data Poisoning Attacks in Recommenders

Data poisoning attacks, involve deliberate manipulation of the recommenders by malicious actors. These attacks aim to influence the recommendations made to users or promote specific items for personal gain or to spread misinformation. By injecting biased or malicious data into the system, attackers can distort the recommendation process, leading to biased or deceptive outcomes. Data poisoning attacks pose a threat to the reliability, fairness, and trustworthiness of RS. A survey on attack models are given by [Lam and Riedl, 2004, Patel et al., 2015, Mobasher et al., 2007, Mingdan and Li, 2020, Burke et al., 2005]

2.3.1 General Form of an Attack Profile

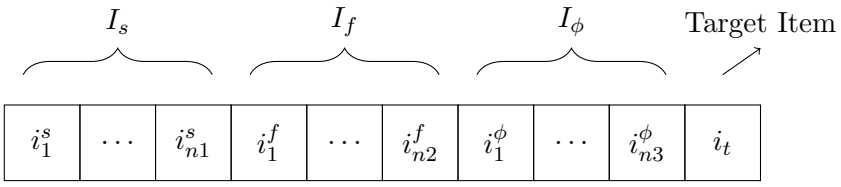


Figure 2.4: The General Form of an Attack Profile

The attack profile is represented as an n-dimensional vector of ratings, where n represents the total number of items within the system. This profile is divided into four distinct parts, as illustrated in Figure 2.4.

- Null Items (I_ϕ): This partition of $n3$ items have no ratings in the profile.
- Target Item (i_t): A specific item, referred to as the target item, receives a rating deliberately chosen to manipulate its recommendations. Typically, this rating is set to either the maximum or minimum possible value, depending on the type of attack being executed.

- Selected Items (I_s): Sometimes, a set of $n1$ items are specially selected and usually assigned high ratings to align the profile with the preferences of users who favour these products.

- Filler Items (I_f): To complete the profile, a set of $n2$ filler items is included. The selection of items for I_f, along with the ratings assigned to them, plays a crucial role in defining the attack model and its characteristics.

Two basic attack models, originally introduced in [Lam and Riedl, 2004] are Random Attack and Average Attack. We extend the Average Attack to specifically target user clusters, which we refer to as Targeted Attack. In this thesis, the focus is on item promotion in RS, where the target item i_t receives high ratings in attack profiles.

- Random Attack: In Random attacks ratings to filler items (I_f) in each rating profile are chosen randomly (centered around the overall average rating in the database.) and I_s is empty.

- Average Attack: Average attacks are a more sophisticated variation: the ratings for filler items (I_f) in attack profiles are distributed around the mean for each item and I_s is empty.

- Targeted Attack: Another sophisticated attack is where the ratings for filler items (I_f) in attack profiles are distributed around the cluster-wise mean for each item in the user cluster and I_s is empty.

We focus on these attacks, as they offer substantial impact while requiring minimal knowledge of the RS. High-knowledge attacks, involving complete knowledge of the RS and its parameters, are less likely to occur. Attackers seek to maximize impact while minimizing costs, making these standard attacks of interest. It's worth noting that acquiring knowledge of these standard attacks is not particularly challenging.

The Random attack is the simplest form and is essentially a zero-knowledge attack. The Average and Target attacks assume access to mean ratings of items. However, it's important to highlight that we don't need the mean rating of every single item within the RS to execute these attacks. Ratings for items considered popular in the RS are relatively easy to obtain and can be utilized for these attacks. The popularity of these items can be assessed using external data sources and doesn't necessarily rely on RS-specific data

[Mobasher et al., 2007]. This is because most users tend to rate only a small fraction of the entire product space, as there's a limit to how many movies, products, or items a person can interact with. Hence, an attacker only needs to focus on a portion of the product space to make their attack effective.

Similarly, for Target attacks, attackers can leverage general domain knowledge about item features to target users who have preferences for specific types of items. For instance, they might identify products with genres or attributes similar to those they want to promote. By identifying users who have rated these related items highly within the RS, the attacker can locate potential targets. In the RS, it's feasible to create fictitious profiles closely resembling real ones by calculating the mean ratings provided by this group of users for other items.

Popular platforms like Amazon and Goodreads publicly display user ratings and reviews for items or books they've engaged with. By filtering users based on the adversary's knowledge of the target item, an attacker can pinpoint user clusters likely to have an inclination towards the target item. The attacker can then emulate the choice patterns of these target users by aggregating ratings from these selected users for other items.

Furthermore, the availability of publicly accessible rating datasets has simplified the process of obtaining preference ratings. For instance, a complete viewing history of any user can be reconstructed with limited adversary knowledge (requiring only 2-8 ratings per user), which doesn't need to be highly precise, from standard datasets like the Netflix Dataset [Narayanan and Shmatikov, 2006]. Additionally, the existence of overlapping sources of preference information can aid in identifying target users. For example, Amazon might potentially re-identify users on competing websites by comparing their purchase history with reviews posted on those sites, enabling them to tailor marketing strategies for these customers. The concept of leveraging overlapping sources was explored and demonstrated by [Frankowski et al., 2006] using machine learning datasets and online forums.

2.4 Attack Detection in Recommender Systems

Different shilling detection techniques that rely primarily on obtaining the signatures of genuine user profiles have been developed to defend by identifying and eliminating false user profiles in RS. Numerous criteria for spotting these anomalies are investigated in

[Burke et al., 2006, Williams et al., 2007, Chirita et al., 2005]. But a major obstacle in user focused detection method is that it can recognize the distinctive signature of only the known attack model. For newer or hybrid attack models, the detection methods may fall short. Unsupervised attack tactics were therefore required.

In one approach presented in [Mehta and Nejdl, 2009], the authors concentrate on the collective behavior of fake user profiles rather than individual profiles. They leverage unsupervised dimensionality reduction techniques to exploit similarity patterns within shilling user profiles, effectively distinguishing them from genuine user profiles. While this method proves effective for attacks demonstrating strong correlations among malicious profiles, [Cheng and Hurley, 2009] highlights that certain attack strategies may deviate from this pattern, making dimensionality reduction-based detection less accurate. They propose a Neyman-Pearson (NP) statistical test to detect random, average and bandwagon attacks. The probability that a new user is an attack profile was determined based on the overlap of the item selections of the new user and the genuine users in the training set.

Another model-free approach involves detecting shilling attacks by identifying abnormalities in item rating distributions, as discussed in [Bhaumik et al., 2006, Zhang et al., 2006, O'Mahony et al., 2006, Yang et al., 2018]. For instance, [Zhang et al., 2006] employs item anomaly detection techniques based on sample averages and sample entropy within time series data, albeit with a focus on dense items with a minimum of 500 ratings. In contrast, [O'Mahony et al., 2006, Yang et al., 2018] examine deviations between actual and predicted ratings for target items to flag abnormal ratings.

Furthermore, some studies, such as [Mehta and Nejdl, 2008, Hidano and Kiyomoto, 2020, Resnick and Sami, 2007, Christakopoulou and Banerjee, 2019, Xu et al., 2020], aim to construct manipulation-resistant RSs designed to mitigate the impact of injected fake ratings and protect the system from shilling attacks.

Chapter 3

Addressing the User Cold Start Problem by Leveraging User Clusters

3.1 Introduction

While existing CF approaches to recommendation perform quite satisfactorily for already existing users in the system, it fails completely for new users, because the system has no knowledge about their preference history. This is a well known problem called a cold-start problem and is a popular research focus.

How to quickly and reliably learn the preferences of new users remains a key challenge in the design of RS. In this chapter we introduce a new type of online learning algorithm, cluster-based bandits, to address this challenge. This exploits the fact that users can often be grouped into clusters based on the similarity of their preferences, and this allows accelerated learning of new user preferences since the task becomes one of identifying which cluster a user belongs to and typically there are far fewer clusters than there are items to be rated. Clustering by itself is not enough however. Intra-cluster variability between users can be thought of as adding noise to user ratings. Deterministic methods such as decision-trees perform poorly in the presence of such noise. We identify so-called distinguisher items that are particularly informative for deciding which cluster a new user belongs to despite the rating noise. Using these items the cluster-based bandit algorithm is able to efficiently adapt to user responses and rapidly learn the correct cluster to assign

to a new user.

Before we proceed, we will review the major research directions in solving cold start problem.

3.2 Related Work

For a recent survey of solutions to the cold-start see [Elahi et al., 2016, Elahi et al., 2018]. Passive approaches include recommending popular items, use of item-based recommendation (once user starts rating items an item-based approach is used to recommend similar items), transfer learning from another RS previously used by a user, and asking new users to rate a fixed list of items. Examples of early work on active learning include IGCN (information gain through clustered neighbors) which uses a decision tree with user clusters as leaves [Rashid et al., 2008] and the ternary decision-tree approach of [Golbandi et al., 2011]. More recently, [Amatriain et al., 2009] uses representative items i.e. after completing the ratings matrix R, k columns of R are selected, the ratings of the other items are represented as a linear combination of these and during cold start a new user is asked to rate these representative items. This approach is extended to use a decision-tree approach by [Shi et al., 2017a]. In [Zhou et al., 2011] a MF approach is proposed whereby a decision-tree is trained to map from item ratings to the latent feature vector for a user. Use of multi-arm bandits for cold-start has also received attention. In [Felício et al., 2017] after completing the ratings matrix R its rows are clustered and the average ratings vector for each cluster is used as a representative user. During cold start an MAB is used to select the average ratings vector to use and the user is asked to rate next highest item in the vector – this is akin to the cluster-bandit algorithm with no exploration phase. In [Canamares et al., 2019] a MAB is used to select between recommender strategies, typically recommending popular items initially for a new user and later switching to a kNN or MF model.

3.3 Fast Cold-Start for Recommender System New Users

When a new user joins the system it initially has no knowledge of the preferences of the user and so would like to quickly learn these[1]. The RS therefore initially starts in an

[1]The system may have some general context regarding the user, e.g. the country/city they are located in, in which case learning is conditioned on this but the fundamental cold start task otherwise remains unchanged.

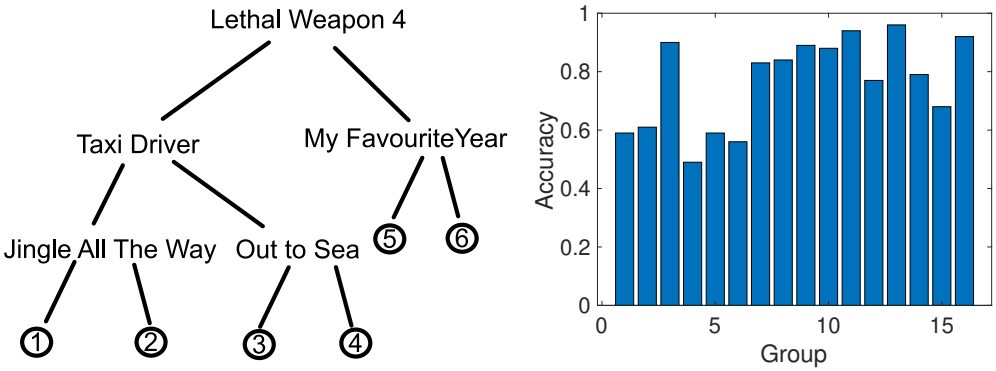


Figure 3.1: (a) Illustrating a movie recommender decision-tree (adapted from [Zhou et al., 2011]), (b) Decision-tree accuracy for Netflix data (16 groups).

"exploration" phase where the first few items that it asks the new user to rate are chosen with the aim of discovering the user's preferences. For brevity we focus on the simplest setup where a user explicitly rates items presented to them, e.g. on a 1-5 scale or binary like/dislike feedback, and the aim of the recommender system is to predict other items that the user may like.

One common approach to this new user cold-start task is to take ratings already collected from a population of users, use these to cluster users into groups and then train a decision-tree to learn a mapping from item ratings to the user group, see for example Figure 3.1(a). When a new user joins the system this decision-tree is used to decide which items the user is initially asked to rate and in this way the group to which the user belongs is initially estimated. Once the group is estimated, the system recommends items liked by members of that group e.g. using MF or another CF approach.

However, typically users clustered in the same group do not give identical ratings to an item. Rather there is a spread of ratings, and this intra-cluster variability between users can be thought of as adding noise to the ratings. Unfortunately, decision trees can easily make mistakes in the face of such noise. For example, Figure 3.1(b) shows the measured decision-tree accuracy for Netflix data clustered into 16 groups (see later for more details). It can be seen that the accuracy is as low as 50-60% for a number of groups.

Multi-arm bandits (MABs), and more generally online convex learning, has been the subject of much interest in recent years. However, naive application of standard bandit algorithms to the cold-start task leads to poor performance. If we think of each RS item as an arm of a MAB then we run into the difficulty that (i) there are many arms and so learning is slow and (ii) repeated pulls of the same arm tend to be highly correlated[2]. One

[2]Measurement studies indicate that when people are repeatedly asked to rate the same item on a scale of 1-5 then if they rate 1 or 5 they tend to consistently stick with that rating although when they rate 3 or

remedy is to associate an arm with each group rather than each item. For each group the available items are sorted in descending order of their predicted rating by users in that group. Pulling the arm for a group then corresponds to asking the user to rate the next item from this sorted list, i.e. the unrated item predicted to have the highest rating for members of the group. While this greatly reduces the number of arms in the MAB, as we will see later the learning rate remains very slow. This is because items rated highly by members of one group tend to also be rated highly by members of at least some of the other groups, and so the user ratings for these items do not serve to strongly distinguish between groups and so allow rapid learning.

In this paper we propose a novel cluster-based bandit algorithm that achieves fast learning in cold-start, e.g. for the standard Netflix dataset < 10 items need to be rated in order to reliably distinguish between 16 user groups and < 12 items to reliably distinguish between 32 groups. We show that the group of a user is identified with significantly higher accuracy than with a decision-tree without incurring higher regret i.e the learning performance is fundamentally superior to that of a decision-tree.

3.3.1 Group Indicator Vector

What we would like to measure is an indicator vector I i.e. a vector for which as we collect more user ratings the element $I(g)$ tends towards 1 when a new user belongs to group g and all other elements $I(h), h \neq g$ tend towards 0. We can obtain such a vector as follows. Start by defining

$$R_n(g,h) = \sum_{i=1}^{n-1} \alpha_i^n(g,h) \frac{r(v_i) - \mu(h,v_i)}{\mu(g,v_i) - \mu(h,v_i)} \tag{3.1}$$

where $v_i, i = 1, \ldots, n-1$ is the sequence of items that the new user has rated upto turn n, $r(v_i)$ is the new user's rating of item v_i, $\mu(g,v_i)$ is the mean rating of item v_i by users in group g and $\alpha_i^n(g,h)$ is a weighting such that $\sum_{i=1}^{n-1} \alpha_i^n(g,h) = 1$. An estimate of $\mu(g,v_i)$ is known, e.g. from ratings by existing users of the RS. Note that calculation of $R_n(g,h)$ only requires asking a user to rate each item once, although if the RS allows it then it is of course possible for sequence $v_i, i = 1, \ldots, n$ to contain duplicate elements i.e. for the same item to be rated multiple times.

4 they may change their rating back and forth between 3 and 4. That is, lumping ratings of 3-4 together in a single bucket these previous studies indicate that a user's rating of an item tends to be consistent.

To streamline notation suppose the new user is in group 1, we can always relabel the groups so that this holds[3]. Then $R_n(1,h) = \sum_{i=1}^{n-1} \alpha_i^n(1,h) \left(\frac{r(v_i)-\mu(h,v_i)}{\mu(1,v_i)-\mu(h,v_i)} \right) = \sum_{i=1}^{n-1} \alpha_i^n(1,h) \left(1 + \frac{r(v_i)-\mu(1,v)}{\mu(1,v_i)-\mu(h,v_i)} \right)$ for $h \neq 1$. Intuitively, the deviations $r(v_i) - \mu(1,v), i = 1, \ldots, n-1$ tend to fluctuate around 0, as otherwise there would be a consistent offset between the user's ratings and the group ratings in which case the user would better be assigned to a different group. Therefore $R_n(1,h) \to 1$ as $n \to \infty$ for $h \neq 1$. By the same argument, $R_n(g,1) = \sum_{i=1}^{n-1} \alpha_i^n(g,1) \left(\frac{r(v_i)-\mu(1,v)}{\mu(g,v_i)-\mu(1,v_i)} \right)$ for $g \neq 1$ and the deviations $r(v_i) - \mu(1,v), i = 1, \ldots, n-1$ tend to fluctuate around 0. Thus $R_n(g,1) \to 0$ as $n \to \infty$ for $g \neq 1$. Hence,

$$I(g) = \min_{h \in G \setminus \{g\}} R_n(g,h)$$

is an indicator vector of the desired form i.e. $I(1) \to 1$ and $I(g) \to 0$, $g \neq 1$ as $n \to \infty$. We then estimate the group $\hat{g}$ that the new user belongs to using

$$\hat{g} \in \arg\max_{g \in G} I(g)$$

The key to fast learning is that $I(g)$ converges quickly to a $(0,1)$ vector.

3.3.2 Fast Convergence

Intuitively, an item v helps to distinguish whether a user belongs to group g rather than group h when (i) the mean rating of v by users in group g is very different from that of users in group h i.e. $(\mu(g,v) - \mu(h,v))^2$ is large, and (ii) when the ratings tend to be consistent/reliable i.e. the variance $\sigma^2(g,v)$ is small. That is, we expect that

$$\Gamma_{g,h}(v) = \frac{(\mu(g,v) - \mu(h,v))^2}{\max\{\sigma^2(g,v), \sigma^2(h,v)\}}$$

is a measure of the ability of item v to distinguish group g from group h i.e. the larger $\Gamma_{g,h}(v)$ the better item v is at distinguishing group g from group h. For $R_n(g,h)$ to converge quickly we then want to choose the sequence of items $v_i, i = 1, \ldots, n-1$ that the new user is asked to rate so that $\Sigma_n(g,h) = \sum_{i=1}^{n-1} \Gamma_{g,h}(v_i)$ is large.

Another way to arrive at the same conclusion is to assume that for users belonging to group g the rating $r(v)$ of item v is i.i.d. subgaussian with mean $\mu(g,v)$ and variance $\sigma^2(g,v)$. Note that any bounded random variable is subgaussian.

[3]Of course we need to make sure that the learning algorithm we develop does not depend upon this relabeling.

With this assumption using standard concentration inequalities, Lemma 5,6 tell us that

$$Prob(|R_n(1,h) - 1| > \epsilon) \leq 2e^{-\frac{\epsilon^2}{2}\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)} \tag{3.2}$$

$$Prob(|R_n(g,1) - 0| > \epsilon) \leq 2e^{-\frac{\epsilon^2}{2}\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)} \tag{3.3}$$

when we select $\alpha_i^n(g,h) = \frac{\Gamma_{g,h}(v_i)}{\sum_{j=1}^{n-1}\Gamma_{g,h}(v_j)}$. That is, for $R_n(g,h)$ to converge quickly we want $\sum_{j=1}^{n-1}\Gamma_{g,h}(v_j)$ to be large. This suggests that selecting items with higher values of $\Gamma_{g,h}(v)$ provides more information to differentiate between groups, leading to quicker convergence toward the correct cluster 1 i.e. indicator vector $I(1) = \min_{h\neq 1}|R_n(1,h)| \to 1$ and the algorithm converges to the correct cluster with high probability.

3.3.3 Exploration vs Exploitation

The foregoing discussion suggests that for fast learning we want to initially ask the user to rate those items v_i for which $\Gamma_{g,h}(v_i)$ is largest. However, these may not be items that receive high ratings, and so there is a cost to this accelerated learning. We therefore want to limit the duration of the initial exploration phase and quickly switch to an exploitation phase where we recommend items that are predicted to be rated highly by the new user i.e. items for which $\mu(\hat{g}, v)$ is large where $\hat{g}$ is the estimated group of the new user. We refer to such highly-rated items as the "best items" within a cluster. Note that learning can still occur during the exploitation phase provided the items v_i rated have non-zero $\Gamma_{g,h}(v_i)$, but as we will see the learning rate can be much slower than during the exploration phase.

In addition to deciding when to switch from the initial exploration phase (selecting high $\Gamma_{g,h}(v)$ items) to the subsequent exploitation phase (selecting high $\mu(\hat{g}, v)$ items), within each phase we need to balance between confirming the new estimate and exploring the other possibilities. It may happen that a new users rating for an item is unusually high or low for their group and so we need to ask them to rate multiple items in order to correct for mistakes and gain confidence in the estimate $\hat{g}$ of the group to which they belong. To do this we employ a form of upper confidence bound (UCB) strategy. Namely, at step n we select the next item v to present to the user to be the unrated item for which learning rate $\Gamma_{\hat{g},h}(v)$ is largest and h is the group for which $\Sigma_n(\hat{g}, h)$ is lowest. Selecting h in this way means that we will tend to explore all pairs $(\hat{g}, h)$ of groups in such a way that we gain a similar amount of information, as measured by $\Sigma_n(\hat{g}, h)$, about each pair.

3.3.4 Algorithm

Pseudo-code for the exploration phase of the resulting bandit algorithm is given in Algorithm 1. Upon exiting its exploration phase the cluster-based bandit algorithm enters its exploitation phase: the algorithm is the same except that calls to Explore($\hat{g}$) are replaced by calls to Exploit($\hat{g}$), which selects the unrated item for which $\mu(\hat{g}, v)$ is largest.

The algorithm uses the parameter $C \in \mathbb{R}_+^{|G|}$ to maintain a pool M_n of **candidates**. If the proxies $\Sigma_n(g, h)$ are large then (3.2) says all $R_n(1, h) \to 1$. Hence $\min_h |R_n(1, h)| \to 1$ and the correct cluster is included with high probability. For the other clusters, (3.3) says $R_n(g, 1) \to 0$. Hence for $g \neq 1$ we have $\min_h |R_n(g, h)| \to 0$ and the wrong clusters are excluded with high probability. The parameter C gauges the confidence in the correctness of the included clusters. When $C = 0$, there are no candidates, while $C \geq 1$ treats all clusters as candidates. Hence, C must be in the interval $C \in (0, 1)$. From lemma 2, as C increases, the likelihood of exiting the algorithm with the wrong cluster tends to increase as well. Therefore, C needs to ensure that ideally, after enough pulls, only the correct cluster remains a candidate in subsequent turns.

If there are no candidates (see the **else** loop) the algorithm explores the cluster with the least confidence bound. Since large $\min_h \Sigma(1, h)$ means the correct cluster is more likely to be a candidate on subsequent turns, the algorithm tries to increase $\min_h \Sigma_n(1, h)$ by increasing the smallest $\Sigma_n(g, h)$.

As the proxies $\Sigma_n(g, h)$ increase (indicating more ratings are elicited from users), there's a higher likelihood that the correct cluster will be included, while the wrong clusters are likely to be excluded. However, it is important to remember that the distinguisher items in the exploration phase need not be items that receive high ratings. Thus, the algorithm has to conclude the exploration phase at some point to subsequently present the user with the best items. The parameter B limits the time allocated for exploration. The algorithm exits the exploration phase when $\Sigma_n(g) = \min_{h \neq g} \Sigma_n(g, h) \geq B$ for any cluster g and subsequently presents the user with the best items in the cluster.

When B is too small, we lack enough information to distinguish between clusters effectively. This increases the risk of selecting the wrong cluster, especially before hitting threshold B and moving to the exploitation phase. Conversely, if B is too large, the algorithm pulls suboptimal distinguisher items frequently, leading to increased regret.

Therefore, choosing B requires balancing accuracy and runtime considerations. Finding the ideal threshold involves considering the $\Gamma_{g,h}(v)$ of each potential correct group

to ensure sufficient information is gathered for accurate group identification. However, achieving this balance for all potential correct group choices remains challenging. Thus, for experimentation purposes, B was determined through a process of trial and error, choosing the one that resulted in optimum performance.

Algorithm 1: Cluster-based Bandit Algorithm

Input: Groups G, mean ratings $\mu(g, v)$ for item v and variances $\sigma(g, v)^2$; parameters $B, C \in \mathbb{R}_+$.

Select initial $\hat{g} \in G$, e.g. randomly, and Explore($\hat{g}$)
for $n = 1, 2, 3 \ldots .$ **do**
  Define the **candidates** as $M_n = \{g \in G : |\min_{h \neq g} |R_n(g, h)| - 1| \leq C)\}$
  if $M_n \neq \varnothing$ **then**
    $\hat{g} \in \arg\max_{g \in M_n} I(g)$
    Explore($\hat{g}$)
    if $\Sigma_n(\hat{g}) \geq B$ **then**
      Estimate new user belongs to group $\hat{g}$
      Exit exploration phase for $\hat{g}$
  else
    $(\hat{g}, h) \in \text{argmin}_{g, h \neq g} \Sigma_n(g, h)$
    Explore($\hat{g}$)

Algorithm 2: Explore($\hat{g}$)

$V_n = \{v_1, \ldots, v_{n-1}\}$ (set of items already rated by user)
$h \in \text{argmin}_{h \neq \hat{g}} \Sigma_n(\hat{g}, h) = \sum_{i=1}^{n} \Gamma_{\hat{g}, h}(v_i)$
Ask user to rate item $v_n \in \arg\max_{v \notin V_n} \Gamma_{\hat{g}, h}(v)$

3.4 Regret Analysis of Algorithm

Notation 1. *Let $\mathcal{G}$ be the set of all user clusters. Each user belongs to one group $g \in \mathcal{G}$. Let $d = |\mathcal{G}| - 1$ denote the number of pair-wise distinguisher item sets for each cluster $g \in G$ and V_N denote the set of all items rated by a user from turn $1, 2, \cdots, N - 1$. We define the following notations and assumptions:*

1. *Let $V(g) \subset V_N$ be the set of all items that serve as distinguishers for a cluster g i.e. given every cluster g has pairwise distinguisher items that distinguish it from all other clusters $h \neq g$, the set $V(g)$ encompasses all such pairwise distinguishers*

items that serve to distinguish cluster g from the rest.

2. *Let $V^*(g) \subset V_N$ be the set of all items that serve as best items i.e. items with large $\mu(g, v)$ for a cluster g. The definition of best items in a cluster may vary among RS. Typically, these items are predicted to have high ratings in the cluster, such as those equal to or greater than 4 on a 5-point scale.*

3. *Let $J_N(g)$ be the total number of times the distinguisher items of set $V(g)$ are selected over turns $1, 2, \cdots, N-1$.*

4. *Let $J_N^*(g)$ be the total number of times the best items of set $V^*(g)$ are selected over turns $1, 2, \cdots, N-1$.*

5. *Let Γ_g be the lowest learning rate provided by an item from $V(g)$ such that, $\Gamma_g \leq \min_{h \neq g} \Gamma_{g,h}(v_i)$ for $v_i \in V(g)$.*

6. *Let γ_g be the lowest learning rate provided by an item from set $V^*(g)$ such that $\gamma_g \leq \min_{h \neq g} \Gamma_{g,h}(v_i)$ holds for each item $v_i \in V^*(g)$.*

Assumption 1. *Suppose $g = 1$ is the correct cluster of the user.*

Assumption 2. *Define $\mu(1, v^*) = \theta$ for all $v^* \in V^*(1)$ and $\theta > 1$*

Assumption 3. *Any selected item $v \in V_N$ satisfies $\Gamma_{g,h}(v) \geq \delta$ where $\delta > 0$*

Assumption 4. *Define $V(g) \cap V^*(g) = \phi$.*

Assumption 1 is primarily for convenience in presenting the results and conducting the analysis. Assumption 2 indicates that all the best items in the correct cluster 1 share a common, high predicted mean value represented by θ. Assumption 3 ensures that the chosen items exhibit sufficient distinguishability among various clusters. In other words, having a distinguishability measure exceeding δ suggests that these selected items provide meaningful information for distinguishing between different user clusters. Assumption 4 implies that the set of distinguishing items $V(g)$ is distinct from the set of best items $V^*(g)$. While in practice, there might be clusters with unique preferences where their best item choices could also act as distinguishers, we maintain Assumption 4 for the purposes of analysis.

Under these assumptions, we define the following lemmas :

LEMMA 1 (Probability Bound of Eliminating the Correct Cluster). *For $T \geq 0$ and $C \in (0, 1)$, we define a turn M, such that*

$$M = \begin{cases} \min\{n \in \mathbb{N} : \Sigma_M(1) > T\} & \text{if } M \text{ exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Then we have cluster 1 not a candidate at turn M (line 3, Algorithm 1) with probability at most

$$2d \exp\left(-\frac{C^2}{2}T\right) \tag{3.4}$$

□

PROOF. For the event cluster 1 is not a candidate in line 3 to occur, $|R_M(1, h)|$ must deviate from 1 for any $h \neq 1$. Let E be the event that the first line of the definition is true. For any $h \neq 1$, we have

$$P(|R_M(1,h) - 1| > C) = P(E \text{ and } |R_M(1,h) - 1| > C) + P(E^c \text{ and } |R_M(1,h) - 1| > C)$$

In case E^c occurs, then $M = \infty$ and by tail bounds in (3.2), $P(|R_\infty(1, h) - 1| > C) = 0$ since $\sum_{i=1}^{\infty} \Gamma_{1,h}(v_i) \to \infty$ by assumption 3.

Hence, $P(|R_M(1, h) - 1| > C) = P(E \text{ and } |R_M(1, h) - 1| > C)$. Using the tail bounds in (3.2), the probability that any $R_M(1, h)$ deviates from 1 by a factor greater than C is at most

$$2 \exp\left(-\frac{C^2}{2} \sum_{i=1}^{M-1} \Gamma_{1,h}(v_i)\right) \leq 2 \exp\left(-\frac{C^2}{2}\Sigma_M(1)\right)$$

Given value of minimum accumulated information measure by turn M as $\Sigma_M(1) > T$, the event occurs with probability at most

$$2 \exp\left(-\frac{C^2}{2}T\right).$$

Finally, we take a union bound over all possible h to get the stated bound

$$\sum_{h \neq 1} 2 \exp\left(-\frac{C^2}{2}T\right) = 2d \exp\left(-\frac{C^2}{2}T\right)$$

□

LEMMA 2 (Probability Bound of Misidentification of Correct Cluster). *Let $g \neq 1$. For $T \geq 0$ and $C \in (0,1)$, we define a turn M, such that*

$$M = \begin{cases} \min\{n \in \mathbb{N} : \Sigma_M(g) > T\} & \text{if } M \text{ exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Then g is a candidate at turn M with probabilty at most

$$2\exp\left(-\frac{(1-C)^2}{2}T\right) \tag{3.5}$$

□

PROOF. From Equation (3.3), we know that any cluster $g \neq 1$ is eliminated as a candidate when $R_M(g,h) \to 0$ for any $h \in \arg\min_{h \neq g} |R_M(g,h)|$. Hence, to guarantee that cluster g is a candidate, we require the event $E(h) = \{||R_M(g,h)| - 1| \leq C\}$ to occur for all $h \neq g$. In other words, we need the intersection of events $\cap_{h \neq g} E(h)$ to occur. Therefore, we can say that the probability of cluster g being a candidate is at most $P(E(1))$ where

$$P(E(1)) = P(|R_M(g,1)| > 1 - C \text{ and } |R_M(g,1)| < 1 + C) \leq P(|R_M(g,1)| > 1 - C)$$

Let A be the event that the set on the first line of the definition of M is true. For any $g \neq 1$, we have

$$P(|R_M(g,1)| > 1 - C) = P(A \text{ and } |R_M(g,1)| > 1 - C) + P(A^c \text{ and } |R_M(g,1)| > 1 - C)$$

In case A^c occurs, then $M = \infty$ and by tail bounds in (3.3), $P(|R_\infty(g,1)| > 1 - C) = 0$ since $\sum_{i=1}^{\infty} \Gamma_{g,1}(v_i) \to \infty$ by assumption 3. Hence, $P(|R_M(g,1)| > 1 - C) = P(A \text{ and } |R_M(g,1)| > 1 - C)$.

Similar to the previous lemma, using tail bounds in (3.3), we can say that $|R_M(g,1)|$ tends to a value $\geq 1 - C$ with probability at most

$$2\exp\left(-\frac{(1-C)^2}{2}\sum_{i=1}^{M-1}\Gamma_{g,1}(v_i)\right) \leq 2\exp\left(-\frac{(1-C)^2}{2}\Sigma_M(1)\right)$$

For any value of $\Sigma_M(1) > T$, we can write this event occurs with probability at most

$$2\exp\left(-\frac{(1-C)^2}{2}T\right).$$

□

Suppose the algorithm runs for $N > 1$ turns. Using the lemma 1 and 2, we proceed to find the expected number of times we explore distinguishers of an incorrect group g i.e. items from set $V(g)$ over the turns $1, 2, \cdots, N-1$.

LEMMA 3. *For any $g \neq 1$ and $\Sigma_N(g) < B$, we have*

$$\mathbb{E}[J_N(g)] \leq \frac{1}{\Gamma_g}\frac{2}{\min\{C^2,(1-C)^2\}}\left(\log\left(2|\mathcal{G}|\right)+2\right)+1$$

□

PROOF. Suppose $J_N(g)\Gamma_g > x$ and both $x, J_N(g) > 1$. Let $M < N$ be the last turn where we selected an item from the set $V(g)$. Such a turn always exists since $J_N(g) > 1$. Thus we have that $J_M(g)\Gamma_g > T = x - \Gamma_g$ and on turn M we ran Explore(g). We claim that when $J_M(g)\Gamma_g > T$, then $\Sigma_M(g) > T$ (we prove this later). We have two possible events to consider at turn M given $\Sigma_M(g) > T$.

Event A_1) In this case, algorithm runs Explore(g) of line 6. This occurs when, at turn M, the following two conditions are met:

- C1: $g \in M_n$ represents the condition that g is a candidate
- C2: $g \in \arg\max_{h \in M_n} I(h)$ represents the condition that g is the optimal candidate choice based on the information I(g).

Consider the events E_1 and E_2, which correspond to the conditions $C1$ and $C2$ being met, respectively. Thus we have,

$$A_1 \subset E_1 \cap E_2$$

Therefore, the probability of running Explore(g) on line 6 is at most $P(E_1)$, which signifies the probability of any $g \neq 1$ being considered a candidate. This probability is bounded by equation (3.5) of lemma 2

$$P(A_1|\Sigma_M(g) > T) \leq P(E_1) = 2\exp\left(-\frac{(1-C)^2}{2}T\right) \tag{3.6}$$

Event A_2) In the second case, the algorithm runs Explore(g) of line 12. This occurs when, at turn M, two conditions are satisfied: C3 and C4.

- C3: $M_n = \phi$ represents the condition that there are no candidates.
- C4: $(g, h) \in \arg\min_{i,j \neq i} \Sigma_M(i, j)$ represents the condition that (g, h) is the optimal choice based on the information measure $\Sigma_M(g, h)$. i.e any cluster $i \neq g$ must satisfy,

$$\min_{j \neq i} \Sigma_M(i, j) > \min_{h \neq g} \Sigma_M(g, h)$$
$$\Sigma_M(i) > \Sigma_M(g)$$

For each $i \neq g$, consider the event $E(i) = \{\Sigma_M(i) > \Sigma_M(g) \text{ and } M_n = \phi\}$ and for $i = g$, event $E(g) = \{\Sigma_M(g) > T \text{ and } M_n = \phi\}$. The intersection of events $\cap_{i \in G} E(i)$ satisfies both $C3$ and $C4$ and ensures that $\Sigma_M(g) > T$ for all $g \in G$. Thus

$$A_2 = \cap_{i \in G} E(i)$$

Therefore, we can say that the probability of running Explore(g) at line 12 is at most $P(E(1))$ where $E(1) = \{\Sigma_M(1) > \Sigma_M(g)$ and 1 is not considered a candidate$\}$. The probability of this event is bounded by equation (3.4) in lemma 1. So we have,

$$P(A_2 | \Sigma_M(g) > T) \leq P(E(1)) = 2d \exp\left(-\frac{C^2}{2} T\right) \tag{3.7}$$

Since events A_1, A_2 are mutually exclusive, we can write

$$P(\text{Explore(g) at turn } M, \Sigma_M(g) > T) = P(A_1, \Sigma_M(g) > T) + P(A_2, \Sigma_M(g) > T) \tag{3.8}$$
$$\leq P(A_1 | \Sigma_M(g) > T) + P(A_2 | \Sigma_M(g) > T) \tag{3.9}$$

where the last inequality is obtained by applying $P(X, Y) = P(X|Y)P(Y) \leq P(X|Y)$ since $P(Y) \leq 1$.

Recall the claim that when $J_M(g)\Gamma_g > T$, then $\Sigma_M(g) > T$. We know that the accumulated sum of information for g up to turn M is given by $\Sigma_M(g) = \min_{h \neq g} \Sigma_M(g, h)$.

Thus we have

$$\Sigma_M(g) = \sum_{i=1}^{M-1} \Gamma_{g,h}(v_i) \tag{3.10}$$

$$= \Gamma_{g,h}(v_1) + \Gamma_{g,h}(v_2) + \cdots + \Gamma_{g,h}(v_{M-1}) \tag{3.11}$$

$$> \sum_{i=1}^{M-1} \mathcal{I}(v_i \in V(g)) \cdot \Gamma_g = J_M(g) \cdot \Gamma_g \tag{3.12}$$

Here $\mathcal{I}(v_i \in V(g))$ represents an indicator vector that is equal to 1 when an item from set $V(g)$ is selected at a turn i and 0 otherwise. Let F_1 be the event that $J_M(g)\Gamma_g > T$ and F_2 be the event that $\Sigma_M(g) > T$. Then $F_1 \subset F_2$ since F_1 implies F_2 but not vice versa i.e. $P(F_1) \leq P(F_2)$. Thus we can write,

$$P(\text{Explore(g) at step } M, J_M(g)\Gamma_g > T) \leq P(\text{Explore(g) at turn } M, \Sigma_M(g) > T)$$
$$\leq P(A_1|\Sigma_M(g) > T) + P(A_2|\Sigma_M(g) > T) \text{ Using 3.9}$$

Also we have, $J_N(g) = J_M(g) + 1$, so $\mathbb{E}[J_N(g)\Gamma_g] = \mathbb{E}[J_M(g)\Gamma_g] + \Gamma_g$ and

$$\mathbb{E}[J_M(g)\Gamma_g] = \int_0^\infty P(\text{Explore(g) at step } M, J_M(g)\Gamma_g > T)dT \tag{3.13}$$

$$\leq \int_0^\infty P(A_1|\Sigma_M(g) > T) + P(A_2|\Sigma_M(g) > T)dT = \int_0^\infty P(T)dT \tag{3.14}$$

We can bound the integral 3.14 over two segments: 0 to α and α to ∞.

$$\mathbb{E}[J_M(g)\Gamma_g] \leq \int_0^\alpha P(T)dT + \int_\alpha^\infty P(T)dT$$

Bounding the probability by 1 in the first segment

$$\leq \int_0^\alpha 1dT + \int_\alpha^\infty P(T)dT$$
$$= \alpha + \int_\alpha^\infty P(A_1|\Sigma_M(g) > T) + P(A_2|\Sigma_M(g) > T)dT$$
$$= \alpha + \int_\alpha^\infty 2\exp\left(-\frac{(1-C)^2}{2}T\right) + (2d)\exp\left(-\frac{C^2}{2}T\right)dT$$

To bound the above let $m = \min\{C^2, (1-C)^2\}$ and $\alpha = \frac{2}{m}\log(2d)$

$$\begin{aligned}
\mathbb{E}[J_M(g)\Gamma_g] &\leq \alpha + \frac{(2)2}{(1-C)^2}\exp\left(-\frac{(1-C)^2}{2}\alpha\right) + \frac{(2d)2}{C^2}\exp\left(-\frac{C^2}{2}\alpha\right) \\
&\leq \alpha + \frac{(2d)2}{m}\exp\left(-\frac{m}{2}\alpha\right) + \frac{(2d)2}{m}\exp\left(-\frac{m}{2}\alpha\right) = \alpha + \frac{8d}{m}\exp\left(-\frac{m}{2}\alpha\right) \\
&\text{Substituting } \alpha = \frac{2}{m}\log(2d), \text{ we get} \\
&= \alpha + \frac{8d}{m}\frac{1}{2d} = \frac{2}{m}\log(2d) + \frac{4}{m} = \frac{2}{\min\{C^2,(1-C)^2\}}(\log(2d)+2) \\
&< \frac{2}{\min\{C^2,(1-C)^2\}}(\log(2|\mathcal{G}|)+2)
\end{aligned}$$

Thus

$$\begin{aligned}
\mathbb{E}[J_N(g)\Gamma_g] &= \mathbb{E}[J_M(g)\Gamma_g] + \Gamma_g \\
&\leq \frac{2}{\min\{C^2,(1-C)^2\}}(\log(2|\mathcal{G}|)+2) + \Gamma_g
\end{aligned}$$

Divide by Γ_g to get

$$\mathbb{E}[J_N(g)] \leq \frac{1}{\Gamma_g}\frac{2}{\min\{C^2,(1-C)^2\}}(\log(2|\mathcal{G}|)+2) + 1$$

Note, we choose N here so that $J_N(g) > 1$ by assumption. But when there does not exist an N such that $J_N(g) > 1$ then we trivially have the bound that $J_N(g) \leq 1$. □

Observe that the expected number of times distinguisher items of cluster $g \neq 1$ are selected is inversely proportional to its corresponding Γ_g. Thus as the information provided by the distinguisher items of cluster g increases, the number of pulls decreases. Also, the term $\log(2|\mathcal{G}|) + 2$ represents the logarithmic dependence on the total number of user groups $|\mathcal{G}|$. This means that increasing the number of groups (thus increasing the number of pairwise distinguishers per group) will only result in a logarithmic increase in the expected number of pulls.

After transitioning from the exploration phase, the cluster-based bandit algorithm enters the exploitation phase: the algorithm is the same except that calls to Explore(g) are replaced by calls to Exploit(g), which selects the unrated item for which $\mu(g, v)$ is largest i.e. items that are predicted to be rated highly by a new user. We refer to this set of items presented as 'best items' of cluster g. Such items may have a reduced learning rate compared to distinguisher items.

For $N > 1$, the next step involves calculating the expected number of times the algorithm exploits the 'best' items of an incorrect group g i.e. items from set $V^*(g)$, over the turns $1, 2, \cdots, N-1$.

LEMMA 4. *For any $g \neq 1$ and $\Sigma_N(g) \geq B$, we have*

$$\mathbb{E}[J_N^*(g)] \leq \frac{1}{\gamma_g} \frac{2}{\min\{C^2, (1-C)^2\}} \left(\log\left(2|\mathcal{G}|\right) + 2\exp\left(-\frac{\min\{C^2, (1-C)^2\}B}{2} \right) \right) + 1$$

□

PROOF. Suppose $J_N^*(g)\gamma_g > x$ and both $x, J_N^*(g) > 1$. Let $M < N$ be the last turn where we selected an item from the set $V^*(g)$. Such a turn always exists since $J_N^*(g) > 1$. Thus we have that $J_M^*(g)\gamma_g > T = x - \gamma_g$ and on turn M we ran Exploit(g). We claim that when $J_M^*(g)\gamma_g > T$, then $\Sigma_M(g) > B + T$ (we prove this later).

As in the previous lemma, we have two possible events to consider at turn M given $\Sigma_M(g) > B + T$.

Event A_1) In this scenario, algorithm runs Exploit(g) of line 6. This occurs when, at turn M, the following two conditions are met:

- C1: $g \in M_n$ represents the condition that g belongs to the set M_n
- C2: $g \in \arg\max_{h \in M_n} I(h)$ represents the condition that g is the optimal choice based on the information I(g).

Consider the events $E1$ and $E2$, which correspond to the conditions $C1$ and $C2$ being met, respectively. Thus we have,

$$A_1 \subset E1 \cap E2$$

Therefore, the probability of running Exploit(g) on line 6 is at most $P(E1)$, which signifies the probability of $g \neq 1$ being considered as a candidate. This probability is bounded by equation (3.5) of lemma 2 given $\Sigma_M(g) > B + T$. Thus we have,

$$P(A_1 | \Sigma_M(g) > B + T) \leq P(E1) = 2\exp\left(-\frac{(1-C)^2}{2}(B+T) \right) \tag{3.15}$$

Event A_2) In this second case, algorithm runs Exploit(g) of line 12. This occurs when, at turn M, two conditions are satisfied: C3 and C4.

- C3: $M_n = \phi$ represents the condition that there are no candidates.
- C4: any cluster $i \neq g$ must satisfy the following to be able to Exploit(g),

$$\Sigma_M(i) > \Sigma_M(g)$$

For each $i \neq g$, consider the event $E(i) = \{\Sigma_M(i) > \Sigma_M(g) \text{ and } M_n = \phi\}$ and for $i = g$, event $E(g) = \{\Sigma_M(g) > B + T \text{ and } M_n = \phi\}$. The intersection of events $\cap_{i \in G} E(i)$ satisfies both $C1$ and $C2$ and ensures $\Sigma_M(g) > B + T$ for all $g \in G$. Thus

$$E = \cap_{i \in G} E(i)$$

Therefore, we can say that the probability of running Exploit(g) at line 12 is at most $P(E(1))$ where $E(1) = \{\Sigma_M(1) > \Sigma_M(g)$ and 1 is not considered a candidate$\}$. The probability of this event is bounded by equation (3.4) in lemma 1 given $\Sigma_M(g) > B + T$.

$$P(A_2 | \Sigma_M(g) > B + T) \leq P(E(1)) = 2d \exp\left(-\frac{C^2}{2}(B + T)\right) \tag{3.16}$$

As in the previous lemma, since events A_1, A_2 are mutually exclusive, we can write

$$P(\text{Exploit(g) at turn } M, \Sigma_M(g) > B + T) = P(A_1, \Sigma_M(g) > B + T) + P(A_2, \Sigma_M(g) > B + T) \tag{3.17}$$

$$\leq P(A_1 | \Sigma_M(g) > B + T) + P(A_2 | \Sigma_M(g) > B + T) \tag{3.18}$$

where the last inequality is obtained by applying $P(X, Y) = P(X|Y)P(Y) \leq P(X|Y)$ since $P(Y) \leq 1$.

Recall the claim that when $J^*_M(g)\gamma_g > T$, then $\Sigma_M(g) > B + T$. We know that the accumulated sum of information for g up to turn M is given by $\Sigma_M(g) = \min_{h \neq g} \Sigma_M(g, h)$.

Thus we have

$$\begin{aligned}\Sigma_M(g) &= \sum_{i=1}^{M-1} \Gamma_{g,h}(v_i) \\ &= \Gamma_{g,h}(v_1) + \Gamma_{g,h}(v_2) + \cdots + \Gamma_{g,h}(v_{M-1}) \\ &> B + \sum_{i=0}^{M-1} \mathcal{I}(v_i \in V^*(g)) \cdot \gamma_g = B + J^*_M(g)\gamma_g\end{aligned}$$

Note that the inclusion of the term B in the final inequality follows from assumption 4 that $V(g) \cap V^*(g) = \phi$. Therefore, the algorithm selects the best items of cluster g only when the accumulated information from distinguisher items of set $V(g)$ crosses the threshold of B.

Here, $\mathcal{I}(v_i \in V^*(g))$ represents an indicator vector that is equal to 1 when an item from set $V^*(g)$ is selected at a turn i and 0 otherwise. Let F_1 be the event that $J^*_M(g)\gamma_g > T$ and F_2 be the event that $\Sigma_M(g) > B + T$. Then $F_1 \subset F_2$ since F_1 implies F_2 but not vice versa i.e. $P(F_1) \leq P(F_2)$.

$$\begin{aligned}P(\text{Exploit(g) at turn } M, J^*_M(g)\gamma_g > T) &\leq P(\text{Exploit(g) called at turn } M, \Sigma_M(g) > B + T) \\ &\leq P(A_1|\Sigma_M(g) > B + T) + P(A_2|\Sigma_M(g) > B + T)\end{aligned}$$

Also we have, $J_N(g^*) = J_M(g^*) + 1$, so $\mathbb{E}[J_N(g^*)\gamma_g] = \mathbb{E}[J_M(g^*)\gamma_g] + \gamma_g$ and

$$\mathbb{E}[J_M(g^*)\gamma_g] = \int_0^\infty P(\text{Exploit(g) called at step } M, J_M(g^*)\gamma_g > T)dT \tag{3.19}$$

$$\leq \int_0^\infty P(A_1|\Sigma_M(g) > B + T) + P(A_2|\Sigma_M(g) > B + T)dT = \int_0^\infty P(T)dT \tag{3.20}$$

We can bound the integral 3.20 over two segments: 0 to α and α to ∞.

$$\mathbb{E}[J_M^*(g)\gamma_g] \leq \int_0^\alpha P(T)dT + \int_\alpha^\infty P(T)dT$$

Bounding the probability by 1 in the first segment

$$\leq \int_0^\alpha 1dT + \int_\alpha^\infty P(T)dT$$
$$= \alpha + \int_\alpha^\infty P(A_1|\Sigma_M(g) > B+T) + P(A_2|\Sigma_M(g) > B+T)dT$$
$$= \alpha + \int_\alpha^\infty 2\exp\left(-\frac{(1-C)^2}{2}(B+T)\right) + (2d)\exp\left(-\frac{C^2}{2}(B+T)\right)dT$$

To bound the above let $m = \min\{C^2, (1-C)^2\}$ and $\alpha = \frac{2}{m}\log(2d)$.

$$\mathbb{E}[J_M^*(g)\gamma_g] \leq \alpha + \frac{(2)2}{(1-C)^2}\exp\left(-\frac{(1-C)^2}{2}(B+\alpha)\right) + \frac{(2d)2}{C^2}\exp\left(-\frac{C^2}{2}(B+\alpha)\right)$$
$$\leq \alpha + \frac{(2d)2}{m}\exp\left(-\frac{m}{2}(B+\alpha)\right) + \frac{(2d)2}{m}\exp\left(-\frac{m}{2}(B+\alpha)\right)$$
$$= \alpha + \frac{8d}{m}\exp\left(-\frac{m}{2}(B+\alpha)\right) = \alpha + \frac{8d}{m}\exp\left(-\frac{mB}{2}\right)\exp\left(-\frac{m\alpha}{2}\right)$$

Substituting $\alpha = \frac{2}{m}\log(2d)$

$$= \alpha + \frac{8d}{m}\exp\left(-\frac{mB}{2}\right)\frac{1}{2d} = \frac{2}{m}\log(2d) + \frac{4}{m}\exp\left(-\frac{mB}{2}\right)$$
$$\leq \frac{2}{\min\{C^2,(1-C)^2\}}\left(\log(2|\mathcal{G}|) + 2\exp\left(-\frac{\min\{C^2,(1-C)^2\}B}{2}\right)\right)$$

Thus,

$$\mathbb{E}[J_N^*(g)\gamma_g] = \mathbb{E}[J_M^*(g)\gamma_g] + \gamma_g$$
$$\leq \frac{2}{\min\{C^2,(1-C)^2\}}\left(\log(2|\mathcal{G}|) + 2\exp\left(-\frac{\min\{C^2,(1-C)^2\}B}{2}\right)\right) + \gamma_g$$

Divide by γ_g to get

$$\frac{1}{\gamma_g}\frac{2}{\min\{C^2,(1-C)^2\}}\left(\log(2|\mathcal{G}|) + 2\exp\left(-\frac{\min\{C^2,(1-C)^2\}B}{2}\right)\right) + 1$$

Note, we choose N here so that $J_N^*(g) > 1$ by assumption. But when there does not exist an N such that $J_N^*(g) > 1$ then we trivially have the bound that $J_N^*(g) \leq 1$. □

Observe that the expected number of times best items of set $V^*(g)$ is exploited is

inversely proportional to its corresponding γ_g. Thus as the information provided by the best items of incorrect cluster g increases, the number of pulls decreases. So learning can still occur even if an incorrect cluster moves to the exploitation phase as long as γ_g is non-zero.

In the next theorem, the objective is to bound the regret accumulated up to a maximum turn $N > 1$. Under assumption 1-4, we define $\Delta(g, w) = \theta - \mu(1, w)$ for $w \in V(g)$ and set $\Delta(g) = \max_{w \in V(g)} \Delta(g, w)$. Similarly, $\Delta(g, w^*) = \theta - \mu(1, w^*)$ for $w^* \in V^*(g)$ and set $\Delta^*(g) = \max_{w^* \in V(g)} \Delta(g, w^*)$.

$\Delta(g), \Delta^*(g)$ denote the maximum regret when presenting an item from the set $V(g)$ and $V^*(g)$ of each group g to the user (of unknown cluster 1). Using Definition 1 (Appendix), we can express pseudo-regret for the algorithm up to turn N as

$$
\begin{aligned}
R_N &\leq \sum_g \mathbb{E}[J_N(g)]\Delta(g) + \sum_g \mathbb{E}[J_N^*(g)]\Delta^*(g) \\
&= \mathbb{E}[J_N(1)]\Delta(1) + \sum_{g \neq 1} \mathbb{E}[J_N(g)]\Delta(g) + \sum_{g \neq 1} \mathbb{E}[J_N^*(g)]\Delta^*(g) + \mathbb{E}[J_N^*(1)]\Delta^*(1) \\
&= \mathbb{E}[J_N(1)]\Delta(1) + \sum_{g \neq 1} \mathbb{E}[J_N(g)]\Delta(g) + \sum_{g \neq 1} \mathbb{E}[J_N^*(g)]\Delta^*(g) \text{ Since } \Delta^*(1) = 0 \text{ by Assumption 2}
\end{aligned}
$$

THEOREM 1. *Under assumption 1-4, for $B > 0$ and $m = \min\{C^2, (1-C)^2\}$, we have pseudo-regret R_N bounded by*

$$
(B+1)\frac{\Delta(1)}{\Gamma_1} + \sum_{g \neq 1}\left(\frac{L1(g)}{\Gamma_g} + 1\right)\Delta(g) + \sum_{g \neq 1}\left(\frac{L2(g)}{\gamma_g}\right)\Delta(g^*)
$$

where $L1(g) = \frac{2}{m}\left(\log\left(2|\mathcal{G}|\right) + 2\right) + \Gamma_g$ and $L2(g) = \frac{2}{m}\left(\log\left(2|\mathcal{G}|\right) + 2\exp\left(-\frac{mB}{2}\right)\right) + \gamma_g$ □

PROOF. Given pseudo-regret R_N:

$$
\mathbb{E}[J_N(1)]\Delta(1) + \sum_{g \neq 1} \mathbb{E}[J_N(g)]\Delta(g) + \sum_{g \neq 1} \mathbb{E}[J_N^*(g)]\Delta^*(g) \tag{3.21}
$$

Let's examine the first sum. In this context, there are two separate and independent cases to consider:

1. $\Sigma_N(1) \geq B$: This implies that there exists a turn $M < N$ such that

$$M = \max\{n \in \mathbb{N} : \Sigma_M(1) < B\}$$

At turn M, an item v_M is selected resulting in $\Sigma_{M+1}(1) \geq B$, and as a result, cluster 1 concludes its exploration phase. The selected item v_M at turn M could be from set $V(1)$ or any other subset of V_N. Therefore, we can say that $J_{M+1}(1) \leq J_M(1) + 1$. Also, $J_{M+1}(1) = J_N(1)$ because the algorithm can rate items from $V(1)$ only until turn M after which the exploitation phase begins for cluster 1.

To find $J_M(1)$, we have $\Sigma_M(1) < B$ and $J_M(1)$ is at most $\frac{B}{\Gamma_1}$ (using equation 3.12, $J_M(1)\Gamma_1 < \Sigma_M(1) < B$). We write,

$$J_N(1) = J_{M+1}(1) \leq J_M(1) + 1 < \frac{B}{\Gamma_1} + 1$$

2. $\Sigma_N(1) < B$: It implies $J_N^*(1) = 0$ because $V(1) \cap V^*(1) = 0$ and algorithm can only call Explore(1) i.e. rate items from $V(1)$ when $\Sigma_N(1) < B$. Thus, in this case, given $\Sigma_N(1) < B$, we write,

$$J_N(1) < \frac{B}{\Gamma_1}$$

Considering $J_N(1) < \max\left\{\frac{B}{\Gamma_1} + 1, \frac{B}{\Gamma_1}\right\}$, the first sum in equation 3.21 is bounded by $\left(\frac{B}{\Gamma_1} + 1\right)\Delta(1)$.

Next, we calculate the second and third sums of equation 3.21 which deal with selecting items from all $g \neq 1$. As before, we have two distinct cases possible:

1. $\Sigma_N(g) \geq B$: It implies that there exists a turn $R < N$ such that

$$R = \max\{n \in \mathbb{N} : \Sigma_R(g) < B\}$$

At turn R, an item v_R is selected resulting in $\Sigma_{R+1}(g) \geq B$, and as a result, cluster g concludes its exploration phase. The selected item v_R could be from set $V(g)$ or any other subset of V_N. Therefore, we can say that $J_{R+1}(g) \leq J_R(g) + 1$. Also, $J_{R+1}(g) = J_N(g)$ because the algorithm can rate items from $V(g)$ only until turn R after which the exploitation phase begins for cluster g.

To find $J_R(g)$, we have $\Sigma_R(g) < B$ and using lemma 3, $J_R(g)$ is at most $\frac{L1(g)}{\Gamma_g}$. We write,

$$J_N(g) = J_{R+1}(g) \leq J_R(g) + 1 < \frac{L1(g)}{\Gamma_g} + 1$$

Given $\Sigma_N(g) \geq B$, by lemma 4, $J_N^*(g) < \frac{L2(g)}{\gamma_g}$.

2. $\Sigma_N(g) < B$: This implies $J_N^*(g) = 0$ because $V(g) \cap V^*(g) = 0$ and algorithm can only call Explore(g) i.e. rate items from $V(g)$ when $\Sigma_N(g) < B$. To find $J_N(g)$, we have $\Sigma_N(g) < B$ and using lemma 3, $J_N(g)$ is at most $\frac{L1(g)}{\Gamma_g}$.

   Considering $J_N(g) = \max\left\{\frac{L1(g)}{\Gamma_g} + 1, \frac{L1(g)}{\Gamma_g}\right\}$ and $J_N^*(g) = \max\left\{\frac{L2(g)}{\gamma_g}, 0\right\}$. Thus the second sum of equation 3.21 is at most $\sum_{g \neq 1}\left(\frac{L1(g)}{\Gamma_g} + 1\right)\Delta(g)$ and the third sum is bounded by $\sum_{g \neq 1}\left(\frac{L2(g)}{\gamma_g}\right)\Delta(g^*)$.

   Finally, adding the first, second and third sums together, the regret accumulated is at most,

$$(B+1)\frac{\Delta(1)}{\Gamma_1} + \sum_{g \neq 1}\left(\frac{L1(g)}{\Gamma_g} + 1\right)\Delta(g) + \sum_{g \neq 1}\left(\frac{L2(g)}{\gamma_g}\right)\Delta(g^*) \tag{3.22}$$

□

We can see that the regret is independent of the number of turns N i.e. regret does not tend to infinity as the number of turns N tends to infinity. This is because the algorithm has prior knowledge about the cluster-wise reward distributions of items. If one has no prior knowledge of the distributions, then asymptotically (in N) a regret of order $\log(N)$ is unavoidable [Bubeck et al., 2013].

Further, the regret depends on the reciprocal of the information measure Γ_1, Γ_g and γ_g. As the information provided by the selected item increases, the regret decreases because the correct group is identified faster.

3.5 Performance Evaluation

3.5.1 Evaluation Setup

Datasets. We evaluate the performance of the cluster-based bandit algorithm for cold start on the standard Netflix dataset (480,189 people 17,770movies, 104M ratings from

1–5), the Jester dataset (73,421 people rating 100 jokes, 4.1M ratings from -10–10) and the Goodreads10K dataset (53,424 people rating10,000 books, 5.9M ratings from 1-5).

Clustering Users. We use training data to cluster users into groups and estimate the mean $\mu(g, v)$ and variance $\sigma(g, v)^2$ of the ratings by each group g for item v. We use the BLC matrix-factorization clustering algorithm [Checco et al., 2017] for this, although other clustering algorithms might also be used. We vary the number of groups/clusters from 4 to 32 and report results for each.

Baseline Algorithms. We compare the performance of the cluster-based bandit algorithm against an optimised CART decision tree. This is a strong baseline, with good performance for cold-start active learning. In addition, as a second baseline we use the cluster-based bandit algorithm but without the initial exploration phase that selects high $\Gamma_{g,h}(v)$ items. This allows the added value of the exploration phase to be measured.

Modelling New Users. We generate the item ratings of a new user from group g by making a single draw from the multivariate Gaussian distribution with mean $\mu(g, v)$ and variance $\sigma(g, v)^2$ for each item equal to that estimated from the training data. This has the advantage that we can easily generate large numbers of new users in a clean, reproducible manner. In addition, we also evaluated performance when drawing ratings from the empirical distribution of ratings for a group (so relaxing the Gaussian assumption) and also by generating user ratings via a water-filling approach i.e. split the data into training and test data. The ratings from users in the training data are used to estimate the mean $\mu(g, v)$ and variance $\sigma(g, v)^2$ of the ratings by each group g for item v. To evaluate the performance, pick a user from the test data and use their ratings. When we need a rating for an item that the test user has not rated, pick a second test user from the same group who has rated the item and merge the pair of user ratings. We found the performance of these setups to be very similar to simply drawing a new user from a Gaussian distribution.

Performance Metrics. We report the accuracy with which the group of a new user is estimated i.e. the fraction of times the correct group is estimated and also the regret i.e. $\sum_{i=1}^{n} r(v_i) - r(v_i^*)$ where v_i^* is the unrated item with highest rating by the new user (so v_1^* is the highest rated item, v_2^* the next highest and so on). We measure the convergence time as the number of items rated before the regret reaches 80% of its final value over a test run. Statistics are calculated over 1000 new users per group.

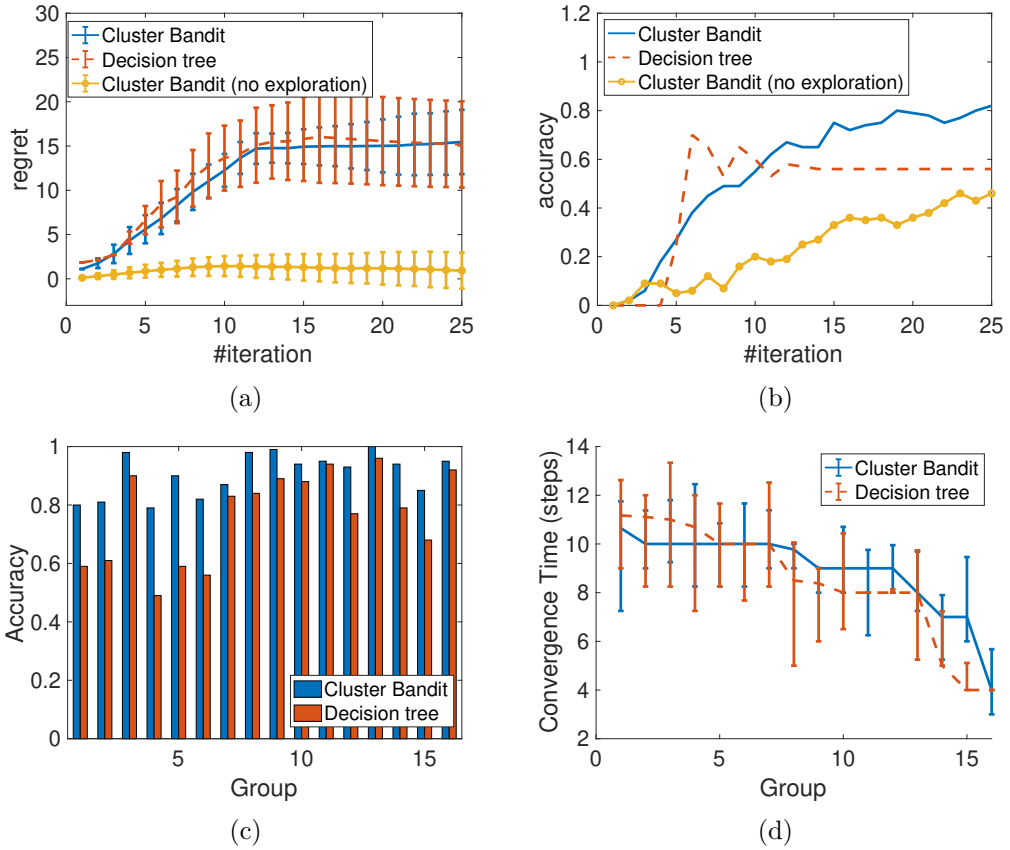


Figure 3.2: Performance measurements for Netflix dataset (16 groups). Solid lines indicate mean, error bars one std deviation (estimated over 1000 new users in each group).

3.5.2 Results

Algorithm 1 has three design parameters B and C. We use $B = 5$ and $C = 0.5$.

Figure 3.2 shows typical (i.e. representative of the full data) detailed performance measurements for the Netflix dataset with 16 groups. It can be seen from Figure 3.2(a) that the regret of the cluster-based bandit and decision-tree algorithms are similar while in Figure 3.2(b) it can be seen that the cluster-based bandit algorithm achieves a significantly higher accuracy than the decision-tree, and from Figure 3.2(c) this holds across all 16 groups.

Figure 3.2(d) shows the convergence time across all groups. This is similar for both algorithms, consistent with Figure 3.2(a). The fact that both algorithms have similar regret and convergence time yet the bandit algorithm achieves higher accuracy indicates that the cluster-based bandit algorithm uses a more efficient learning approach.

It can also be seen from Figure 3.2(a,b) that the learning performance of the cluster-based bandit without the initial exploration phase is rather poor: the accuracy is typically substantially worse than for the decision tree and the convergence time is slow. That is,

Dataset/Algo	Accuracy				Convergence Time			
	#Groups				#Groups			
	4	8	16	32	4	8	16	32
Netflix/DT	0.93	0.88	0.75	0.54	6.09	7.68	9.30	11.52
Netflix/CB$^-$	0.83	0.79	0.65	0.53	-	-	-	-
Netflix/CB	**0.99**	**0.96**	**0.91**	**0.75**	5.15	7.21	9.85	12.03
Jester/DT	0.71	0.55	0.46	0.34	6.22	7.90	8.66	10.15
Jester/CB$^-$	0.84	0.75	0.60	0.52	-	-	-	-
Jester/CB	**0.91**	**0.83**	**0.73**	**0.68**	4.61	6.46	8.92	10.48
Books/DT	0.88	0.69	0.62	0.41	7.80	9.01	9.82	7.37
Books//CB$^-$	0.82	0.72	0.6	0.55	-	-	-	-
Books//CB	**0.96**	**0.87**	**0.84**	**0.67**	6.05	7.02	9.20	7.32

Table 3.1: Mean accuracy and convergence time for Netflix, Jester and Books datasets vs #groups. Legend: DT=decision-tree, CB=cluster-based bandit algorithm, CB$^-$=cluster-based bandit algorithm without exploration phase. Dash $-$ indicates CB$^-$ failed to converge within 25 items/steps.

it is inefficient to use only the most highly-rated items during a cold start, as might be expected.

Table 3.1 presents summary accuracy and convergence time measurements for the Netflix dataset and also for the Jester and Books datasets. The values shown are averages over the groups (with 1000 new users per group, so e.g. with 16 groups the value shown is the average over 1000×16 users). It can be seen that the cluster-based bandit algorithm consistently achieves substantially higher accuracy than a decision tree and the cluster-based bandit algorithm without an exploration phase. For the standard Netflix dataset, < 10 items need to be rated to reliably distinguish between 16 user groups. Importantly, this improved accuracy does not come at the cost of slower convergence time. That is, the performance of the cluster-based bandit dominates that of the other algorithms and this data indicates that it should always be used in preference to them if higher accuracy is desired.

Further, we note from Table 3.1 that the proposed approach demonstrates superior performance over the decision tree, especially when the number of clusters is large. This is presumably because increasing the number of clusters from 4 to 32, reduces the distinction between them. This poses challenges in differentiating between ratings from two clusters due to the intra-cluster noise. Because of this noise, an unusual rating will lead the decision tree to follow a poor path from which it cannot recover because the decision tree itself is fixed. Thus the offline training of decision trees constrains their ability to adapt to such noisy feedback. Ensemble learning techniques like Random Forest are known to be less affected by noise. However, they require repeated sampling or querying from users (to average out the noise) which is not feasible in a RS set-up.

However, we expect that a dynamic construction of a tree based on user feedback may be more adaptable to such noise. For instance, the MCTS (Monte Carlo Tree Search) approach in [Rajapakse and Leith, 2022] has shown superior accuracy compared to the cluster-based bandit algorithm. MCTS dynamically selects the sequence of items presented to a new user by building a look-ahead search tree based on feedback. In contrast, decision trees are pre-built using training data, rendering online cluster-based bandit algorithms and MCTS more resilient to noise.

3.6 Conclusions

In this chapter we revisit the cold-start task for new users of a RS and propose a novel cluster-based bandit algorithm that achieves fast learning in cold-start, e.g. for the standard Netflix dataset < 10 items need to be rated in order to reliably distinguish between 16 user groups and < 12 items to reliably distinguish between 32 groups. We show that the group of a user is identified with significantly higher accuracy than with a decision-tree without incurring higher regret or longer learning time i.e the learning performance is fundamentally superior to that of a decision-tree.

Chapter 4

Evaluating Impact of User-Cluster Targeted Attacks in Matrix Factorisation Recommenders

4.1 Introduction

In the last chapter, we looked at how users in a RS can often be grouped by their interests, and indeed this observation motivates many advertising strategies [Epasto et al., 2021, Xie and Phoha, 2001, Geyik et al., 2015]. Recent research proposes exploiting such clustering of users in RS for their 'hiding in the crowd' privacy benefits [Checco et al., 2017, Ravichandran and Vassilvitskii, 2009, Bindra, 2021]. However, it is well known that RS are susceptible to data poisoning attacks [Lam and Riedl, 2004, Patel et al., 2015].

In poisoning attacks, an adversary adds fake profiles with carefully assigned ratings for selected items and attempts to target an item to promote/demote it by increasing/decreasing the item's rating in the system. Social media platforms have received attention for discriminatory and predatory targeting of user communities [Angwin and Parris Jr., 2016, Datta et al., 2014]. Further, adversaries have been known to use RS to influence users by introducing false product reviews [He et al., 2021] to promote misinformation. RS recommends items by learning user's preferences and contributes to shaping and developing new opinions and interests, thus influencing user behavior. Therefore, selective exposure to misinformation is a significant concern in RS.

This study explores the impact of data poisoning attacks targeting specific user clusters

in the standard MF based RS. To better understand this, recall the working mechanism of MF-based RS as described in Section 2.1.1. We have, U the user-feature matrix and V the item-feature matrix. When fake ratings are introduced into a RS, the feature matrices U, V undergo changes. It is crucial to analyse the nature and extent of these changes to conduct further research on the robustness of MF-based RS. This study aims to investigate the impact of poisoning attacks on the feature matrices U and V, an aspect that has not been previously investigated in the literature. In particular, we investigate attacks targeting specific user groups and how the changes to U and V matrices enable these attacks.

The standard MF approach is to update both U and V matrices when new ratings enter RS. To investigate how U and V are individually affected by the attacks and how their behaviors contribute to the propagation of targeted attacks in the RS, we consider updates to each matrix separately in response to the newly entered fake ratings [Shams and Leith, 2022], i.e. i) Hold V constant and update U to study the changes to U after attacks, ii) Hold U constant and update V to study the changes to V after attacks. The findings indicate that the impact of the attack is pronounced when V is updated in response to the introduction of fake ratings.

The effectiveness of attacks on feature matrices can be influenced by the choice of target items. Most work in the literature [Wu et al., 2021, Fang et al., 2020, Fang et al., 2018, Huang et al., 2021, Christakopoulou and Banerjee, 2019, Hu et al., 2019] study the effect of attacks on randomly chosen target items or on items with a small number of ratings. In this work, we take a broader view and consider various choices of target item to identify the factors contributing to attack effectiveness on the U, V matrices by analyzing how these matrices change after an attack on these target items. We conclude that certain items are more susceptible to attacks than others. Specifically, we have found that items with a lower number of ratings in the target cluster are particularly vulnerable. This is because the feature vectors for these items in matrix V can be easily manipulated, making them more susceptible to attacks. So the distribution of user and item latent features plays a significant role in the effectiveness of attacks targeting a user group.

Many recent works approach data poisoning attacks from an optimization perspective, requiring a high level of knowledge about the RS [Fang et al., 2020, Christakopoulou and Banerjee, 2019, Lin et al., 2020]. For this work, we revisit more straightforward attack strategies [Lam and Riedl, 2004, Patel et al., 2015, Mingdan and Li, 2020] involving users rating the target item with a large rating to promote it and the rest of the items (filler

items) with scores that mimic the rating distribution of the targeted user cluster. We show that this low-knowledge attack is enough to promote an item in a target cluster.

The main contributions are as follows:

- We provide a systematic study of data poisoning attacks targeted at a group of users and identify the factors contributing to the effectiveness of attacks in MF-based RS.
- We analyse the individual effects of attacks on latent feature matrices U and V and explore how they contribute to the propagation of targeted attacks in a MF-based RS. Studies investigating the role of latent feature matrices are new to the literature.
- We illustrate the findings with real-world datasets and show that a simple attack strategy using limited knowledge of user preferences suffices to target a specific user group precisely.

4.2 Related Work

The benefits of clustering users in RS are discussed in [Checco et al., 2017, Shams et al., 2021, Yanxiang et al., 2013, Mittal et al., 2010]. For example, [Checco et al., 2017] proposes enhancing privacy by assigning the users to clusters and providing cluster-wise recommendations rather than for each individual user and [Shams et al., 2021, Yanxiang et al., 2013] exploits user clusters for improving cold-start recommendation.

The impact of data poisoning attacks where fake users are injected in RS with carefully crafted user-item interaction has been studied extensively. For surveys on attack models and the robustness of RS algorithms, see [Lam and Riedl, 2004, Patel et al., 2015, Mobasher et al., 2007, Mingdan and Li, 2020]. Studies such as [Huang et al., 2021, Li et al., 2016, Fang et al., 2018, Fang et al., 2020] focus on modelling attacks specific to the type of RS, e.g. [Fang et al., 2020] studies attacks in MF-based RS by proposing to model fake users similar to true users who are influential to the recommendations, [Huang et al., 2021, Fang et al., 2018] propose data poisoning attacks for deep learning based RS and graph-based RS respectively.

Very few studies explore the impact of data poisoning attacks targeting specific user clusters in MF-based RS. Earliest work on the analysis of segment/targeted attack was studied by [Mobasher et al., 2005]. They demonstrated that the segment/targeted attack is effective against both user-based and item-based collaborative filtering RS and show

that such a segment-focused attack helps to shield fake profiles from suspicion because it has a limited impact on the whole user base while having a big impact on the targeted users who are likely to consume the item.

Other recent works exploring targeted attacks is probably [Hu et al., 2019, Christakopoulou and Banerjee, 2019]. In [Christakopoulou and Banerjee, 2019], Generative Adversarial Networks (GAN) generate fake users whose distribution is close to that of the true users. They assume that the adversary has knowledge of the recommender's model and algorithm and aims at adversarial training of RS to build robust recommendation models (for other adversarial robustness studies, see [Xu et al., 2020, Deldjoo et al., 2020]). In [Hu et al., 2019], they propose to promote an item to a group of users by injecting fake ratings and social connections in a factorisation-based social RS. They assume that fake users can quickly establish social connections with true users and model the attack as a bi-level optimisation problem.

In the above studies, attacks are modeled as an optimisation problem. They require a good deal of information about the RS to be carried out, i.e. a rather powerful adversary. However, observations from these studies motivate us to revisit the more straightforward average-attack strategy and study their effect on MF-based RS. For example, in [Fang et al., 2018], the attacker generates fake users using a graph-based RS, and the target RS uses MF. The differences between their attack and the existing standard attack strategies are not very large despite the high level of knowledge assumed by the adversary. Similar observations are made in [Lin et al., 2020], where the authors study data poisoning attacks on deep learning RS models. They generate fake profiles by randomly sampling a submatrix from real users who have rated sufficient items to generate fake users close in distribution to the real users. They compare their attack strategy to standard attack models such as a random attack, average attack etc, and find that the effectiveness of the attack is only marginally increased compared to the lower knowledge and simpler average-attack strategy.

4.3 User-Cluster Targeted Poisoning Attack

4.3.1 User Cluster Based Recommendation Model

Recall from Section 2.1.1 that a MF-based recommender factorises user-item rating matrix R approximately as $U^T V$, where matrices U and V have relatively small inner dimension

d. Each column in matrix U is a length d weight vector that captures a user's preferences, and each column in matrix V is a length d weight vector that captures the characteristics of one item. The predicted rating matrix given by the inner product $U^T V$ gives the individual predicted rating per user per item.

We depart from the usual setup by assuming that users belong to distinct groups. We have a set $\mathcal{G}$ of user groups, and each user belongs to one group $g \in \mathcal{G}$ such that users belonging to a group have similar preferences. In this study, we use k-means to cluster columns of matrix U so that users with similar preferences (i.e. weight vectors) are clustered together. However, other clustering algorithms might also be used, e.g. the approach in [Checco et al., 2017].

Let $\tilde{U}_g \in \mathbb{R}^{d \times 1}$ for $g = 1, \cdots, |\mathcal{G}|$ be the preference vector associated with a group of users and gather these features to form matrix $\tilde{U} \in \mathbb{R}^{d \times |\mathcal{G}|}$. Each column in $\tilde{U}$ is a length d weight vector that captures a group's preferences. For users belonging to a group g with weight vector $\tilde{U_g}$ and items with weight vectors in V, the predicted rating matrix is the inner product $\tilde{U_g}^T V$, i.e., users belonging to the same group have the same predicted rating value per item.

4.3.2 Attack Model

We assume that the adversary knows the cluster-wise mean rating for every item in the system. This is a reasonable assumption since such cluster-wise information may be inferred from online databases that publicly display the average user ratings of items (e.g., movie databases, online shopping databases, etc.).

Attackers can use general domain knowledge about the features of items in order to target users who prefer items of a particular type. For example, they may be able to spot products with genres that are comparable to the one they want to advertise. They can find possible targets by locating users who highly score these related items. In the RS, it is possible to then create fictitious profiles that closely resemble real ones by computing the mean ratings that this set of users provided for other items. Examples of such instances have been discussed in Section 2.3.1. Further, Facebook, Instagram and Amazon already offer targeted advertising/information sharing services where you can select any user segment based on preferences to the type of product you want to promote, age, location and other demographics [1].

[1] Amazon Web Services: [Katidis and christianbonzelet, 2022], Facebook: [Meta, 2023]

For the type of attacks that we focus on, there is a *target item* that the attacker is interested in promoting in a target cluster and a set of *filler items* that are rated to ensure that the fake users preferences share some similarity to that of the true users in the target cluster. The target item is given the maximum rating to try to promote it in the system and the filler items ensure that fake users can correctly enter the targeted group.

4.4 Study of Attack Effects on Feature Matrices U and V

In Section 2.1.1, we introduced the standard method for deriving matrices U and V from user-item ratings which involve minimizing the sum of squared errors for all observed ratings, expressed as

$$\min_{U,V} \sum_{(i,j)\in\mathcal{O}} (R_{i,j} - U_i^T V_j)^2 + \lambda \left(\sum_i \|U_i\|^2 + \sum_j \|V_j\|^2 \right) \tag{4.1}$$

After obtaining the preference vectors U_i and V_j for user i and item j respectively, we proceed to cluster these U_i values to obtain matrix $\tilde{U}$ of group vectors, as discussed in Section 4.3.1.

When fake users arrive and start rating items, we study changes in U, V by updating only one of the feature matrices in response to newly entered ratings and keeping the other constant. In particular, we have the following approaches to incorporate the new ratings:

1. *Fix V, update U:* RS holds V constant and updates U for all users after the attack. Keeping V constant, the cost function in equation 4.1, becomes for every user i, a convex function of V. The solution to this least squares problem is

$$U_i = \left(\sum_{j\in\mathcal{V}(i)} V_j V_j^T + \lambda I \right)^{-1} \sum_{j\in\mathcal{V}(i)} V_j R_{i,j} \tag{4.2}$$

   where $\mathcal{V}(i) = \{j : (i,j) \in \mathcal{O}\}$ is the set of items rated by user i. Then re-cluster the U_i values to obtain an updated matrix $\tilde{U}$ of group vectors.

2. *Fix U, update V:* RS holds U constant and updates V after the attack.

$$V_j = \left(\sum_{i \in \mathcal{U}(j)} U_i U_i^T + \lambda I \right)^{-1} \sum_{i \in \mathcal{U}(j)} U_i R_{i,j} \tag{4.3}$$

where $\mathcal{U}(j) = \{i : (i,j) \in \mathcal{O}\}$ is the set of users rating item j.

4.4.1 Fix V, Update U

It can be seen from equation 4.2 that user vector U_i depends on the item vector V_j and rating value $R_{i,j}$ of all items $j \in \mathcal{V}(i)$ rated by user i. For any existing true user i in RS, the addition of fake users neither changes the set $\mathcal{V}(i)$ of items rated nor $R_{i,j}$ and V_j of items in $\mathcal{V}(i)$. So the weight vector U_i is unchanged after an update provided V is kept constant.

Let U_f be the feature vector of a fake user f entering the RS. U_f is generated based on the set of items $\mathcal{V}(f)$ rated by the fake user and the ratings given to these. After generating U_i values for all users (now includes fake users as well), the cluster-feature weights $\tilde{U}$ are updated using k-means. We also warm start the k-means clustering with the initial centers before the attack instead of using a random cluster center initialization. This allows us to focus on regrouping due to adding fake users and avoid spurious group membership changes due to randomness in the initial conditions.

Impact of Attack on Target Cluster Weight Vector $\tilde{U}_t$

The new cluster center is the arithmetic mean of all user weights in the cluster. When fake users successfully enter the target cluster t by generating feature vectors U_f that are close to $\tilde{U}_t$, we update the cluster center to reflect the presence of these new users. The updated $\tilde{U}_t$ is the mean of the feature vectors of the existing true users and the newly added fake users. Therefore the predicted rating of target item j^* in the target cluster given by $\tilde{U}_t^T V_{j^*}$ changes. Thus, the effect of the attack on item j^*, in this case, is determined by the changes to the cluster center $\tilde{U}_t$ resulting from the inclusion of fake user weights. Based on this understanding, we expect the relative number of true users (n) and fake users (m) in the target cluster to play an essential role in the effectiveness of the attack. If $n >> m$, then the impact of fake users on the arithmetic mean $\tilde{U}_t$ is small.

Impact of Attack on Non-Target Items and Non-Target Clusters

The inner product $\tilde{U_t}^T V$ gives the predicted rating for all items in the target cluster. Specifically, we expect that, in the target cluster, items similar to j^* (i.e: item feature vectors V_j and V_{j^*} have high correlation) will show a more significant change in the predicted ratings. In contrast, other items will show a lower change in the predicted ratings.

Recall that any true user's feature vector remains unchanged after injecting fake users in RS. So we expect non-target cluster center $\tilde{U}_g$ for $g \in \mathcal{G} \setminus \{t\}$ to be unchanged if all fake users correctly enter target cluster t. Thus, keeping V constant will isolate the attack effects on the target cluster alone if all fake users enter the target cluster.

Surprisingly, the analysis shows us that the injection of fake ratings has no effect on the user vector U_i of all true users i in the RS (when V is kept constant). Also, the number of true users in target cluster t relative to the number of fake users determines the attack's effect on $\tilde{U}_t$.

It is V_{j^*} that changes in response to injected fake ratings. Further, the post-attack changes are influenced by the number of true ratings to item j^* from the target cluster rather than the total number of true users in the target cluster. We will study this in detail in the next section.

4.4.2 Fix U, Update V

It can be seen from equation 4.3 that a target item vector V_{j*} depends on user vector U_i and rating R_{i,j^*} of all users i in set $\mathcal{U}(j^*)$ who provide rating to the item j^*. V_{j^*} remains unaffected by the number of users in a cluster unless they also provide a rating to item j^*.

Let a fake user f enter RS targeting item j^*. This implies that the set $\mathcal{U}(j^*)$ of users now also contains the fake users. Following this, fake user feature vector U_f and rating R_{f,j^*} are also used to determine V_{j^*}. Therefore, target item vector V_{j^*} will show significant changes after the attack when many fake users enter RS.

Impact of Attack on Target Item Vector V_{j^*}

Let $\hat{V_{j^*}}$ be the updated V_{j^*} after attack. Given the k^{th} item-feature of V_{j^*} represented by $v_k^{j^*}$ and k^{th} user-feature of $\tilde{U}_t$ represented by u_k^t, the change in target item predicted

rating in the target cluster after the attack, for this setup, is

$$\tilde{U_t}^T \hat{V}_{j^*} - \tilde{U_t}^T V_{j^*} = \sum_{k=1}^{d} u_k^t (\hat{v}_k^{j^*} - v_k^{j^*})$$

The change in predicted rating will increase when $\hat{v}_k^{j^*} - v_k^{j^*} = \gamma u_k^t$ for a positive constant γ. This is because, the inner product will then become $\sum_{k=1}^{d} u_k^t (\hat{v}_k^{j^*} - v_k^{j^*}) = \sum_{k=1}^{d} \gamma (u_k^t)^2$. We can see that, as γ increases, product terms in the inner product become larger and positive, increasing the change in the predicted rating of the target item. Specifically, we define the following setup to study the transformation of V_{j^*}.

We noted from equation 4.3 that updating V_{j^*} after the attack requires the U_i values corresponding to all users who have provided ratings, i.e., true users and newly added m fake users. Since users in a group have similar preferences, we set the U_i values of true users to their corresponding cluster weight vector for this study. Given vector $U_f \in \mathbb{R}^{d \times 1}$ associated with fake user f capturing the fake user's preferences, we gather these vectors together to form matrix $X \in \mathbb{R}^{d \times m}$. To improve the predicted rating of the target item in the target cluster t, columns in the X matrix must be closer to $\tilde{U}_t$ than to other cluster feature vectors. To study the maximum changes to V_{j^*}, which corresponds to the maximum change in the predicted rating of the target item, we set U_f equal to $\tilde{U}_t$ in the analysis.

We re-write equation 4.3 such that after adding a block of m fake users, we can formulate updates to V_{j^*} recursively.

THEOREM 2 (*Proof in Appendix B, Section 7.2.3). Given $\hat{V_{j^*}}$ is the updated V_{j^*} after attack. Let a block of m fake users with feature vectors in $X \in \mathbb{R}^{d \times m}$ and target item fake ratings vector $y \in \mathcal{R}^{m \times 1}$ enter RS. Given $A^{-1} = \left(\sum_{i \in \mathcal{U}(j^*)} U_i U_i^T + \lambda I\right)^{-1}$ for set of all true users $\mathcal{U}(j^*)$ rating item j^* and $K = \left(I + X^T A^{-1} X\right)^{-1} \left(y - X^T V_{j^*}\right)$, we can write,*

$$\hat{V_{j^*}} - V_{j^*} = m \times K \times \left(A^{-1} \tilde{U}_t\right) \tag{4.4}$$

□

Analysis of Vector V_{j^*} Update Mechanism

Using equation 4.4, for any k^{th} item-feature of V_{j^*} given by $v_k^{j^*}$ and k^{th} user-feature of $\tilde{U}_t$ given by u_k^t, we can write the updated k^{th} item-feature $\hat{v}_k^{j^*}$ of $\hat{V}_{j^*}$ as

$$\hat{v}_k^{j^*} - v_k^{j^*} = m \times K \left(a_{kk} u_k^t + \sum_{i=1, i \neq k}^{d} a_{ki} u_i^t \right) \tag{4.5}$$

where a_{kk} represents the diagonal and a_{ki} for $\{i \mid i \neq k \text{ and } i \in 1, 2, \cdots, d\}$ represents non-diagonal elements in the k^{th} row of matrix A^{-1}. We can clearly observe from equation 4.5 that $\hat{v}_k^{j^*} - v_k^{j^*}$ is not a simple scaling of u_k^t due to the presence of the second summation term. Therefore the transformation to any feature v_k depends on the matrix A^{-1} and target cluster-feature vector $\tilde{U}_t$.

We know that A^{-1} is a Positive Definite (PD) matrix. [2] Thus all diagonal elements a_{kk} are positive.[3] From this we can say that the term $a_{kk}u_k^t$ always has the sign of u_k^t or in other words scales u_k^t. Equation 4.5 reduces to a simple scaling of u_k^t only when A^{-1} is a positive diagonal matrix causing the non-diagonal terms in any row of A^{-1} to be zero i.e. the summation term $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$ in equation 4.5 becomes zero giving $\hat{v}_k^{j^*} - v_k^{j^*} = m \times K \times a_{kk} \times u_k^t$.

Let's take a closer look at the possible behaviors of equation 4.5 for any k^{th} feature in item vector V_j. We know that m, K are positive constants [4]. So the overall sign and magnitude of the update is determined by the behaviour of term $a_{kk}u_k^t + \sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$. We noted how $a_{kk}u_k^t$ is a term always having the sign of u_k^t, effectively acting as a scaling of u_k^t. Depending on the sign and magnitude of the term $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$ relative to $a_{kk}u_k^t$, we can have the following behaviours:

- case 1: if $sgn(a_{kk}u_k^t) = sgn(\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t)$: then the update factor in equation 4.5 is similar to scaling u_k^t. i.e we could say $\hat{v}_k - v_k = \gamma u_k^t$ where γ is a positive constant.
- case 2: if $sgn(a_{kk}u_k^t) \neq sgn(\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t)$: then,
  - case 2a): $|a_{kk}u_k^t| > |\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$ then here too the update factor in equation 4.5 is similar to scaling of u_k^t. i.e. we could say $\hat{v}_k - v_k = \gamma u_k^t$ where γ is a positive constant.

[2] See Lemma 9 in Appendix B
[3] See Lemma 10 in Appendix B
[4] See Lemma 15 in Appendix B

- case 2b): $|a_{kk}u_k^t| < |\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$, then here the update factor is such that it shifts opposite to $sgn(u_k^t)$.
- case 2c): $|a_{kk}u_k^t| = |\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$, then $\hat{v}_k - v_k = m \times K \times 0 = 0$ implying $\hat{v}_k^{j^*} = v_k^{j^*}$, thus no change in the feature k after attack.

In particular, when target item true rating distribution possess certain structures, we have the following key observations regarding the relative signs and magnitudes of $a_{kk}u_k^t$ and $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$. Further we show that equation 4.5 is convergent under certain conditions.

1) $|a_{kk}u_k^t|$ reduces as the number of true ratings to target item increases:

If we look at the term $A^{-1} = (\sum_{i\in\mathcal{U}(j^*)} U_i U_i^T + \lambda I)^{-1}$, we can write using Sherman-Morrison-Woodbury identity [5] that for every new true user $\hat{i}$ with feature vector $U_{\hat{i}}$ providing ratings to item j^*, the inverse is updated such that

$$\hat{A}^{-1} = (A + U_{\hat{i}}U_{\hat{i}}^T)^{-1} = A^{-1} - \frac{A^{-1}U_{\hat{i}}U_{\hat{i}}^T A^{-1}}{1 + U_{\hat{i}}A^{-1}U_{\hat{i}}^T}$$

From lemma 11 [6], the diagonal elements of updated inverse $\hat{A}^{-1}$ is always less than or equal to the diagonal elements of A^{-1}. This implies that as more true user provides ratings, diagonal values of A^{-1} decrease. i.e., term $|a_{kk}u_k^t|$ reduces as the number of true ratings to the target item increases.

2) $sgn(a_{ki}u_i^t)$ relative to $sgn(a_{kk}u_k^t)$:

By lemma 17 [7], we have that if true ratings to item j^* come exclusively from target cluster users, each term $a_{ki}u_i^t$ in the summation has a sign opposite to $sgn(a_{kk}u_k^t)$. Conversely, if the source of true ratings is not exclusive to the target cluster, then the sign of each term $a_{ki}u_i^t$ in the summation does not depend exclusively on $sgn(a_{kk}u_k^t)$ and may not all be the same.

[5] See Definition 5 in Appendix B
[6] See Lemma 11 in Appendix B
[7] See Lemma 17 in Appendix B

3) Convergence of $A^{-1}\tilde{U}_t$

Suppose item j^* received a large number of true ratings from true users in RS. Will V_{j^*} prove difficult to change after the attack? From case $2c$, we can see that magnitude of updates to the item vector after an attack is reduced only if $A^{-1}\tilde{U}_t$ approaches zero vector. In particular, we have

THEOREM 3 (*Proof in Appendix B). Given A^{-1}, let N true ratings be received by target item such that A^{-1} updates to $\hat{A}^{-1}$. Then column vector $\hat{A}^{-1}\tilde{U}_t$ is guaranteed to converge to zero vector as N approaches infinity only if the N ratings come from the target cluster. For increasing true ratings from any non-target cluster, convergence to zero is not guaranteed.* □

Thus if fewer users from the target cluster provide ratings to item j^*, then irrespective of the number of ratings from non-target clusters, V_{j^*} changes to increase rating in target cluster. So we expect the effectiveness of the attack to depend on the number and cluster-wise distribution of true ratings to the target item. V_{j^*} is guaranteed to not shift the rating of item j^* in the target cluster only when the target cluster provides a large number of true ratings to the item j^*.

Impact of Attack on Non-Target Clusters and Non-Target Items:

From the analysis of equation 4.3, we expect that non-target items in a cluster will not be affected after an attack. This is because the target item's V_{j^*} is updated independent of V_j for all items $j \neq j^*$, thus changes to V_{j^*} after the attack does not affect any V_j. Attack effects leak to non-target item vectors only via changes to $\tilde{U}_t$ as discussed in Section 4.4.1. But, since $\tilde{U}$ is kept constant, post-attack changes are only isolated to the target item.

The predicted rating of target item j^* across all clusters is calculated by $\tilde{U}^T V_{j^*}$. Thus item vector V_{j^*} is common to all the cluster weight vectors in $\tilde{U}$. This implies that changes to the item vector affect all clusters. For example, suppose true users have $U = [1, -1]$ or $U = [-1, 1]$. i.e. the first set of users like items with $V = [1, 0]$ and dislike items with $V = [0, 1]$, but the second set of users are the opposite. Then an attack against the first set of true users keeping $U = [1, -1]$ constant to increase the rating of an item with $V = [0, 1]$ may be performed by shifting $V = [0, 1]$ to $V[1, 0]$. Then such a shift, while it increases the rating in users with $U = [1, -1]$, would decrease the rating in group $U = [-1, 1]$.

But based on observations from real-world datasets in Section 4.6, cluster weight vec-

tors are rarely so simple and opposite in preference values. In fact, with higher d dimensional feature space, the cluster preference vectors enjoy correlation. For example, in Table 4.1a, features $k = 0, 3, 4, 5, 6, 7$ have the same sign across all clusters. Similar observations can be made for the Goodreads dataset in Table 4.1b for features $k = 0, 2, 4, 6, 8, 9$. Thus these features of clusters are positively correlated. In Table 4.1b, feature $k = 1$ has a negative sign in cluster 2 but is positive in all non-target clusters. Thus this feature is negatively correlated with corresponding features of clusters $g = 0, 1, 3$. Similarly, many such features exist in both datasets, with opposite signs between clusters.

g/k	0	1	2	3	4	5	6	7	8	9
0	0.54	0.06	0.03	0.06	0.63	-0.43	0.29	-0.25	0.45	-0.01
1	0.45	-0.16	0.25	0.29	0.88	-0.09	0.29	-0.21	-0.22	-0.11
2	0.71	0.61	-0.13	0.08	0.63	-0.04	0.18	-0.01	-0.12	-0.13
3	0.30	-0.08	-0.46	0.20	0.81	-0.36	0.60	-0.04	-0.17	-0.34

(a) ML

g/k	0	1	2	3	4	5	6	7	8	9
0	-0.14	0.28	-0.18	0.26	-0.53	-0.17	-0.35	0.03	-0.15	-0.63
1	-0.24	0.47	-0.14	0.29	-0.08	-0.09	-0.48	0.04	-0.57	-0.26
2	-0.11	-0.22	-0.47	0.42	-0.53	-0.19	-0.47	0.26	-0.30	-0.10
3	-0.06	0.21	-0.16	-0.16	-0.61	0.11	-0.67	-0.10	-0.19	-0.14

(b) GR

Table 4.1: Cluster-Weight Values for $d = 10$ features against $|G| = 4$ clusters for Movielens (ML) and Goodreads (GR) datasets

We expect that in such correlated clusters, after attack, the effect leaks to non-target clusters as well. While the relative change in rating increases in the target cluster, for non-target clusters, the relative change in rating may increase or decrease after attack depending on their weight vector $\tilde{U}_g$ and correlation with updated target item vector $\hat{V}_{j*}$.

4.5 Experiment Evaluation Set-up

To illustrate the findings discussed in previous sections, we describe the experimental set-up and report the results in Section 4.6.

4.5.1 Datasets

We evaluate the effectiveness of the attack on the MovieLens dataset (943 users rating 1682 movies, contains 100000 ratings from 1-5), widely used in literature for evaluating RS under attack, and the Goodreads 10K dataset (53,424 people rating 10,000 books, 5.9M ratings from 1-5). We take a dense subset of the Goodreads dataset (since MF is computationally expensive for large datasets), obtained by selecting the top 1000 users who have provided the most ratings and the top 1682 items rated by these users. This provides us with 1000 users and 1682 items.

Synthetic User Generation

We evaluate by generating synthetic users using Movielens and Goodreads as a baseline where, by construction, the setup considered in this section is intentionally tightly controlled so that we can vary one aspect at a time and study its impact.

From the original R matrix of the two datasets considered, calculate U, V, and cluster U using k-means to get group membership of all users. For users in each group, calculate the empirical distribution of the ratings for each item. Using this mean and variance of the item ratings for a group g, generate the required number of users per group by drawing a vector of random ratings for items, i.e., for item j generate a random variable with the probability of each rating being given by the empirical distribution previously calculated for group g, item j. Notably, we generate ratings so that the proportion of ratings from group g for each item j matches that in the original Movielens and Goodreads dataset. For example, if an item j received ratings from 10% of the users from cluster g in the original dataset and if we generated 1000 users per cluster, then we ensure that the item j also receives ratings from 10% of users from cluster g (= 100 ratings) in this generated dataset. For a target item, the number and distribution of ratings per group are completely controlled while leaving the distribution of ratings for other items much the same as before. This also has the advantage that we can easily generate large numbers of new users cleanly and reproducibly.

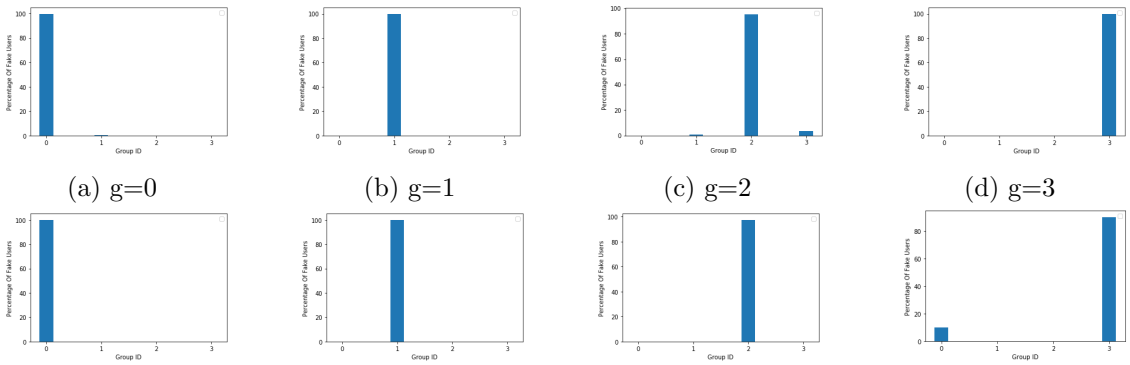
For the study, we generate 4 clusters (results are similar for any number of clusters chosen, but a cluster size of 4 gives a larger sample set of target items with required empirical mean in a target cluster) with latent feature space of dimension $d = 10$ and a number of items of 1682 for both datasets. Note that there must be at least 250 users per cluster. This is because too few users cause fewer ratings to be generated per cluster according to the set-up described. Lack of sufficient ratings per cluster causes the centers of the generated clusters to shift from the original centers. This may cause the membership of generated users to change. So unless mentioned otherwise, for all the simulation studies, we fix the cluster-wise population to be 250 per cluster. The results reported are the average of over 50 such datasets generated randomly as described using Movielens and Goodreads as the baseline.

4.5.2 Threat Model

Target Item

The target item is given the maximum rating to try to promote it in the system. Specifically, in the experiments, we randomly sample an item from a set of items with an empirical mean rating ≤ 3 and treat it as the target item. The number and distribution of ratings per cluster are completely controlled for the chosen target item, as detailed in Section 4.5.1. We thus evaluate the results for different target items based on the number of true ratings it has.

Filler Items



(a) g=0 (b) g=1 (c) g=2 (d) g=3

(e) g=0 (f) g=1 (g) g=2 (h) g=3

Figure 4.1: Plot showing the percentage of fake users entering per cluster when targeting each cluster g for ML and GR Datasets respectively using distinguisher filler items

The filler items are chosen to aid fake users in correctly entering the target cluster. We rate filler items such that the columns in the U matrix corresponding to the fake users are closer to those of users in the target cluster than those in others. Randomly choosing items may risk the fake user falling into a non-target cluster. So we choose those that are likely to be favored by the users of the target group. Thus we choose so-called distinguisher items [Shams et al., 2021] as filler items. Specifically, we choose distinguisher items identified in a cluster with i) mean rating very different and higher from other clusters and ii) the ratings tend to be consistent/reliable, i.e., the variance is small. These 'popular' items in the target cluster can make fake user profiles more realistic and representative of the target cluster user's preferences. Also, too low a number of filler items makes it difficult for fake users to enter the target cluster correctly. So, we fix the number of filler items to be the minimum number of items rated by true users in the generated data set. The ratings for

these filler items are sampled from the Gaussian distribution using the cluster-wise mean rating and a small standard deviation.

In Figure 4.1 we see the percentage of fake users entering the target cluster for the choice of distinguisher filler items. We can observe >= 90% of fake users correctly entering the target cluster. Thus the fake users simulate the preferences of target cluster users accurately.

4.5.3 Performance Metric

We use the change in the rating of the target item relative to the maximum deviation possible as the evaluation metric.

$$\text{Relative Change in Rating}_g = \frac{\mu_f(g,i) - \mu_o(g,i)}{|5 - \mu_o(g,i)|} \tag{4.6}$$

Where $\mu_f(u,i)$ is the predicted rating of target item i of a user in cluster g after the attack, $\mu_o(g,i)$ is the predicted rating of the target item i of a user in cluster g before the attack and 5 is the maximum rating that can be given to the target item in the datasets that we consider.

We note that it is common in the literature to measure the effectiveness of an attack using either Hit-Rate (HR@N), i.e., the fraction of normal users whose top-N recommendation lists contain the target item, or Prediction Shift (PS), i.e., the difference in the rating of the item before and after the attack indicating by how much the rating has increased after the attack. While the PS metric illustrates whether an attack has the intended effect of increasing rating, it does not measure the attack's power. Thus many studies use HR@N to measure the effectiveness of the attack.

However, in a cluster-based RS, the top N list is common to all users in a cluster. Further, the items in the top N list are RS dependent varying over many factors, such as the threshold rating considered above which it may be put in a recommended list, and the percentage of users who already interacted with that item in the cluster. Since the study considers different target items based on the number of true ratings they received, the standard Hit-Ratio is not an ideal evaluation method for us [8].

[8] However, we report results for a slightly modified hit-rate definition as additional results in Appendix B

Equation 4.6 gives the shift in predicted rating relative to the maximum deviation possible after an attack. It therefore directly measures the power of attack on clusters.

4.5.4 Visualising Results

In each study, we average the results over 50 generated datasets and attack each of these to allow us to study variability in the effectiveness of the attack. We then use the mean and standard deviation to visualize the relative change in the rating of the target item after the attack. We show the results when the adversary targets cluster 2. Results when targeting other clusters are similar and so not reported separately.

4.6 Performance Illustration with Data

Recalculate U, V for the generated user-item ratings in each iteration. This becomes the baseline U, V against which we measure the changes after the attack. Inject fake users and perform update approaches 1 and 2 respectively as discussed in Section 4.4. Recall we use a warm start approach when performing updates to U, V and for k-means clustering.

4.6.1 Fix V, update U

Before we proceed to show the results illustrating our findings, let us briefly recall our key observations of approach 1 from Section 4.4.1:

- Relative number of true and fake users plays a role in the effectiveness of the attack. Increasing the number of true users in the target cluster reduces changes to $\tilde{U}_t$ and subsequently the effect of targeted attacks.
- The effect of the attack is isolated to the target cluster when all fake users correctly enter the target cluster. Also, changes to $\tilde{U}_t$ result in 'leakage' of attack to non-target items. The effect of the attack is expected to be pronounced in items highly correlated to the target item.

We first illustrate the attack's impact when the target cluster's true user size is increased relative to the fake user size. We then proceed to illustrate the leakage effect of attack to non-target items and clusters.

Varying Ratio of True Users (n) to Fake Users (m) in Target Cluster:

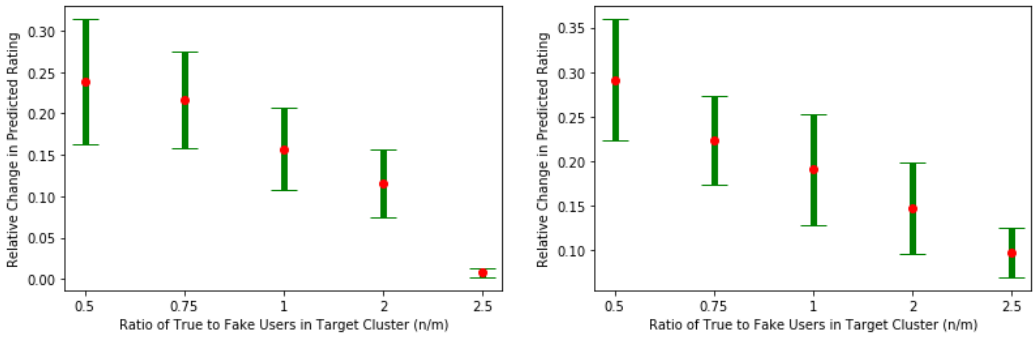
In this experiment, we compare the relative change in target item predicted rating in the target cluster for varying values of ratio $\frac{n}{m}$. We fix the number of target-item true ratings per cluster throughout the experiment to focus only on changes to $\tilde{U}_t$ due to cluster true user population size. This ensures that any noise due to rating distribution is avoided.

To illustrate, we fix a cluster-wise true user population of $250 - 250 - n - 250$ (i.e. 250 users per non-target cluster and n users in target cluster), and a target item rating distribution of $5 - 5 - 5 - 5$ (i.e. 5 ratings per cluster). Then, we vary the ratio of the number of true users to fake users ($\frac{n}{m}$) in the target cluster. We increase the values from $\frac{n}{m} = 0.5$ to 2.5 for both datasets. Figure 4.2 reports the mean and standard deviation of change in the target item's predicted rating when cluster 2 is targeted against the increasing ratio of $\frac{n}{m}$. As expected, it shows decreasing relative change in predicted rating as the ratio increase from 0.5 to 2.5 for both Movielens and Goodreads dataset. This is because when $\frac{n}{m} < 1$, the presence of a higher number of fake user weights compared to true user weights makes it easier for fake users to shift the cluster weight $\tilde{U}_t$. Thus, we conclude that as the number of true users in the target cluster increases relative to the number of fake users, it becomes harder to attack the item.

Effect of Attack on Non-Target Items and Non-Target Clusters

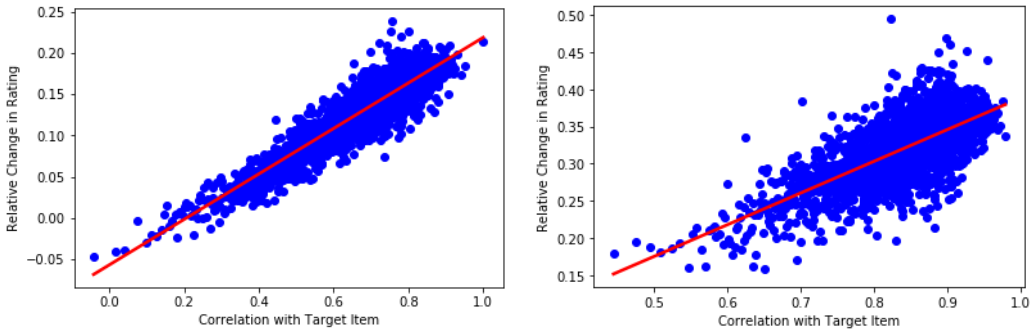
Figure 4.3, gives the correlation of each item with the target item v/s the change in the predicted rating of that item in the target cluster. We can see a linear relationship indicating that the effect of the attack on non-target items in the target cluster depends on the correlation between non-target item vector V_j and target item vector V_{j^*}.

The effect of the attack is not expected to be visible in non-target clusters since almost all fake users enter target cluster t. In Figure 4.4, we take the case of $\frac{n}{m} = 0.5$ and cluster-wise user distribution $250 - 250 - n - 250$ which gave the maximum change in predicted rating for the target item in Figure 4.2 and show the cluster wise change in the target item predicted rating when cluster 2 is targeted. As predicted, we can see that non-target clusters $0, 1, 3$ show negligible change in predicted rating after the attack for both datasets. The results are similar for any attack size or cluster population size. Keeping V constant results in the attack effects being isolated to the target cluster alone (when all fake users enter the target cluster).



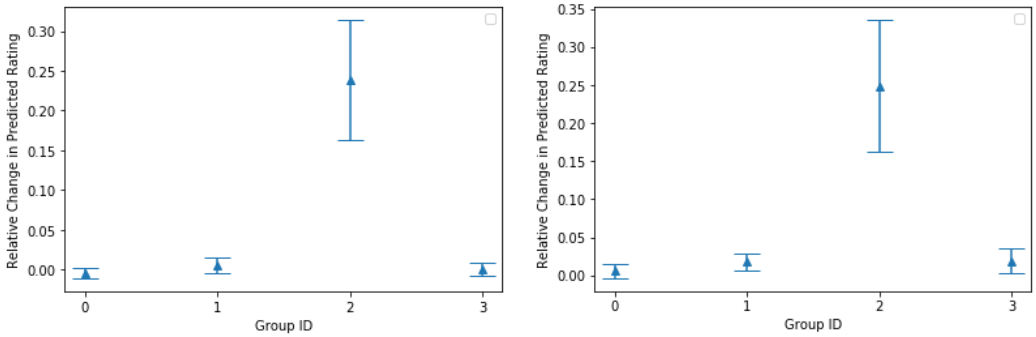
(a) Movielens (b) Goodreads

Figure 4.2: Plot comparing the change in predicted rating in target cluster against the increasing ratio of true users (n) to fake users (m) in the target cluster ($\frac{n}{m}$)



(a) Movielens (b) Goodreads

Figure 4.3: Plot comparing the change in the predicted rating of an item in the target cluster against correlation values between the target item and the other items



(a) Movielens (b) Goodreads

Figure 4.4: Plot comparing the change in the predicted rating of the target item across clusters when cluster 2 is targeted

4.6.2 Fix U, update V

Before we go ahead and show the results illustrating our findings, let us briefly recall our key observations of approach 2.

1. Increasing true ratings to target items from the target cluster guarantees a reduction of the effect of the attack in the target cluster while increasing true ratings from non-target clusters offers no guaranteed protection from attack.

2. Attacks against users in a target cluster may 'leak' and affect users in other clusters. The attack may increase or decrease change in predicted rating in other clusters depending on their correlation with the target item feature vector after the attack. This shows that the V vector propagates the effect of the attack to all non-target clusters. This contrasts the case when V is kept constant, causing the attack to be isolated to the cluster where the fake users are present.

3. Simply increasing the true user population (n) relative to the fake user population (m) does not reduce the change in the target item predicted rating in the target cluster after the attack. This is in contrast to when we update U (keeping V constant), where increasing n relative to m reduces the effect of the attack.

First, we show that increasing true ratings from the target cluster reduce the attack's impact as predicted. Then, we study the attack when the target item is rated by no users in the target cluster but by users in other clusters. For both experiments, we also illustrate the mechanism of how $a_{kk}u_k^t$ and $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$ of equation 4.5 work to result in the observed output. Finally, we show that increasing the number of true users in the target cluster does not reduce the impact of the attack on V_{j^*}.

Varying Ratio of True Ratings (N_t) to Fake Ratings (N_f) in Target Cluster

We consider attacks when the target item has received true ratings in the target cluster but no ratings in the non-targeted clusters. As discussed in previous sections, we expect that increasing true ratings from the target cluster will reduce the attack's impact. Particularly, we study the effect of attack with increasing true ratings N_t from the target cluster given N_f fake ratings. To illustrate, we increase values for $\frac{N_t}{N_f}$ from 0.05 to 2.5 for $N_f = 100$ fake ratings and treat it as the target item. We note that the results presented show a similar trend for any choice of increasing ratio.

Results:

Figure 4.5 reports mean and standard deviation plots comparing the change in the predicted rating of the target item in target cluster 2 after the attack versus the ratio $\frac{N_t}{N_f}$. It

can be seen from both Figures 4.5(a) and 4.5(b) that an item with $\frac{N_t}{N_f} = 0.05$ shows the largest mean shift in predicted rating after the attack compared to an item with $\frac{N_t}{N_f} = 2.5$ in the target cluster. i.e., it is harder to change the predicted rating of the target item after the attack as the number of true ratings N_t received by the target item increases in the target cluster.

Illustration of Update Mechanism:

Let us walk through the update mechanism here. Specifically, in this particular case, since all true ratings come exclusively from the target cluster, as discussed in Section 4.4.2, lemma 17 case (1) predicts equation 4.5 to be such that each term in $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$ always has a sign opposite to $sgn(a_{kk}u_k^t)$. Also, recall that as the true ratings N_t increase, diagonal term a_{kk} decreases.

For $\frac{N_t}{N_f} = 0.05$ i.e. when number of true ratings N_t is least, diagonal values a_{kk} are higher such that all features follow mechanism of case $2a : |a_{kk}u_k^t| > |\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$, thus updates to feature v_k is similar to scaling of u_k^t. Table 4.2a reports $\hat{v}_k - v_k$ update values for each feature k of the target item vector for both datasets. Let us focus on $|\hat{v}_k - v_k|$ corresponding to large $|u_k^t|$ in both datasets [9]. From the Table, we can see that $|\hat{v}_k - v_k|$ corresponding to $k = 0, 1, 4$ in ML and $k = 2, 3, 4, 6$ in GR result in large values contributing more to the inner product $\tilde{U}_t^T \hat{V}_{j^*}$ resulting in the higher relative change in rating values for $\frac{N_t}{N_f} = 0.05$.

As the true ratings N_t increases when ratio $\frac{N_t}{N_f}$ increases from 0.05 to 2.5, diagonal term a_{kk} decreases, bringing down $|a_{kk}u_k^t|$ causing the gap between the two terms $|a_{kk}u_k^t|$ and $|\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$ to reduce. Thus $|\hat{v}_k - v_k|$ will tend to smaller values. From table 4.2a, we note all feature's $|\hat{v}_k - v_k|$ decreasing to smaller values for $\frac{N_t}{N_f} = 2.5$. A similar reduction in magnitude can be observed for the Goodreads dataset. This translates as the effect of attack decreasing for increasing ratio as illustrated by Figure 4.5.

Leakage of Attack to Non-Target Clusters:

To see the effect of attack leakage to non-target clusters, Figure 4.6 further breaks down the change in rating by the user clusters when cluster 2 is targeted in Movielens and

[9] Recall with $\tilde{U}_t$ constant, change in rating after attack is given by ${\tilde{U}_t}^T \hat{V}_{j^*} - {\tilde{U}_t}^T V_{j^*} = \sum_{k=1}^{d} u_k^t(\hat{v}_k^{j^*} - v_k^{j^*})$. Note $\hat{v}_k^{j^*} - v_k^{j^*}$ corresponding to a large $|u_k^t|$ will contribute more to change in rating than $\hat{v}_k^{j^*} - v_k^{j^*}$ corresponding to a lower $|u_k^t|$ since such low $|u_k^t|$ will down-weight $\hat{v}_k^{j^*} - v_k^{j^*}$ values.

$\frac{N_t}{N_f}$	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9
ML : U_2	**0.71**	**0.61**	**-0.13**	**0.08**	**0.63**	**-0.04**	**0.18**	**-0.01**	**-0.12**	**-0.13**
0.05	1.15	0.99	-0.23	0.15	1.03	-0.07	0.29	-0.02	-0.20	-0.20
2.5	0.28	0.23	-0.05	0.04	0.28	-0.05	0.07	-0.03	-0.05	-0.06
GR : U_2	**-0.11**	**-0.22**	**-0.47**	**0.42**	**-0.53**	**-0.19**	**-0.47**	**0.26**	**-0.30**	**-0.10**
0.05	-0.21	-0.43	-0.90	0.81	-1.04	-0.37	-0.90	0.51	-0.60	-0.22
2.5	-0.05	-0.10	-0.22	0.20	-0.28	-0.07	-0.21	0.15	-0.13	-0.009

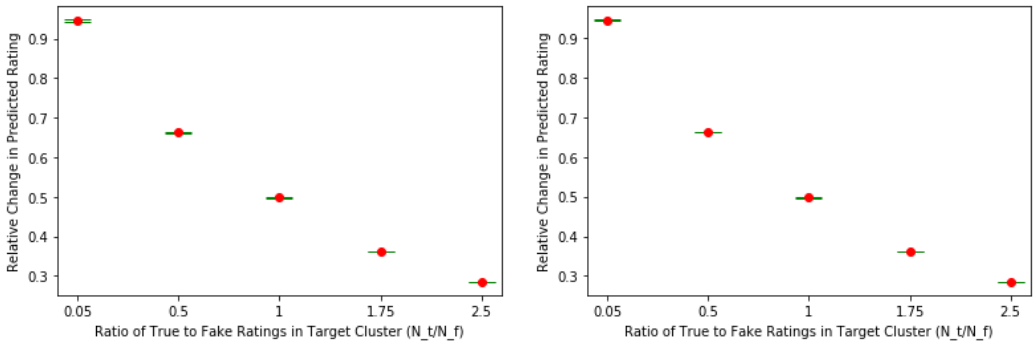
(a) True rating distribution $0 - 0 - N_t - 0$

$\frac{N_t}{N_f}$	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9
ML : U_2	**0.71**	**0.61**	**-0.13**	**0.08**	**0.63**	**-0.04**	**0.18**	**-0.01**	**-0.12**	**-0.13**
0.05	1.43	2.45	-0.35	-0.16	0.23	0.72	-0.47	0.45	-0.64	-0.11
2.5	1.34	2.45	-0.35	-0.16	0.12	0.80	-0.47	0.49	-0.71	-0.10
GR : U_2	**-0.11**	**-0.22**	**-0.47**	**0.42**	**-0.53**	**-0.19**	**-0.47**	**0.26**	**-0.30**	**-0.10**
0.05	-0.02	-1.84	-1.40	1.26	-0.75	-0.55	-0.45	1.08	-0.44	0.83
2.5	-0.01	-1.90	-1.37	1.21	-0.56	-0.49	-0.40	1.09	-0.47	1.06

(b) True rating distribution $N_t - N_t - 0 - N_t$

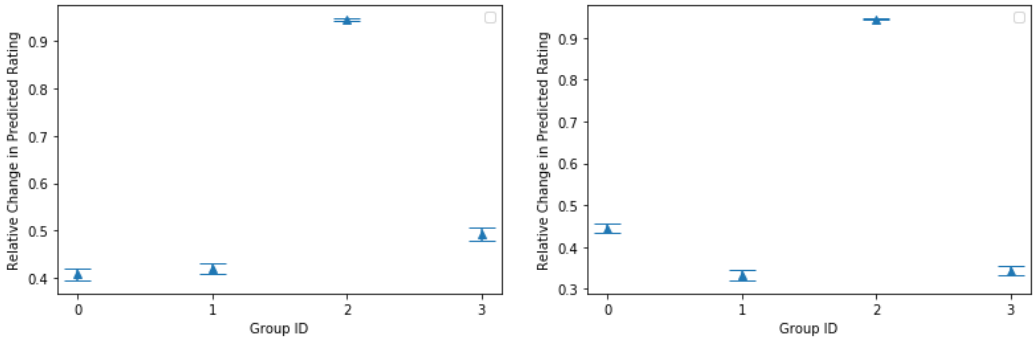
Table 4.2: Target cluster 2 feature vector and target item update vector $\hat{v_k^{j^*}} - v_k^{j^*}$ values for Movielens and Goodreads Dataset for $d = 10$ features

Goodreads dataset. It can be seen from Figure 4.6 that the non-target clusters also show a positive change in the predicted rating [10]. The attack, therefore, effectively fails to be focused on the target cluster.



(a) Movielens (b) Goodreads

Figure 4.5: Plot comparing the relative change in predicted rating in target cluster 2 against increasing ratio $\frac{N_t}{N_f}$ in the target cluster



(a) Movielens (b) Goodreads

Figure 4.6: Plot comparing the relative change in predicted rating across clusters when targeting cluster 2 with $\frac{N_t}{N_f} = 0.05$

[10] We note that there may be target items that, after transformation, result in leakage such that non-target clusters show a negative relative change in rating. But we do not find an item that may illustrate such behavior here. This is because, for our experiments, we choose randomly from a set of target items that has a reliable empirical mean ≤ 3. Such behavior may be found for a larger set of representative target items or in other datasets.

Varying Ratio of True Ratings (N_t) to Fake Ratings (N_f) in Non-Target Clusters

We now consider attacks when the target item has received true ratings in non-target clusters but no ratings in the targeted cluster. Particularly, we study the effect of attack with increasing true ratings N_t from each non-target cluster given N_f fake ratings. To illustrate, we increase values of $\frac{N_t}{N_f}$ per non-target cluster from 0.05 to 2.5 given $N_f = 100$ fake ratings. Since no true ratings come from the target cluster, by theorem 3, increasing total true ratings may not reduce the attack's impact in the target cluster.

Results:

Figure 4.7 reports the measured mean change in the target item's predicted rating in the targeted cluster against increasing $\frac{N_t}{N_f}$. It can be seen that the decreasing trend is much slower (almost negligible) compared to what was reported in Figure 4.5, signifying that even with a larger absolute number of true ratings compared to the previous case, the attack is effective in the target cluster. i.e., the large overall number of true ratings fails to protect users in the targeted cluster as presumed.

Illustration of Update Mechanism:

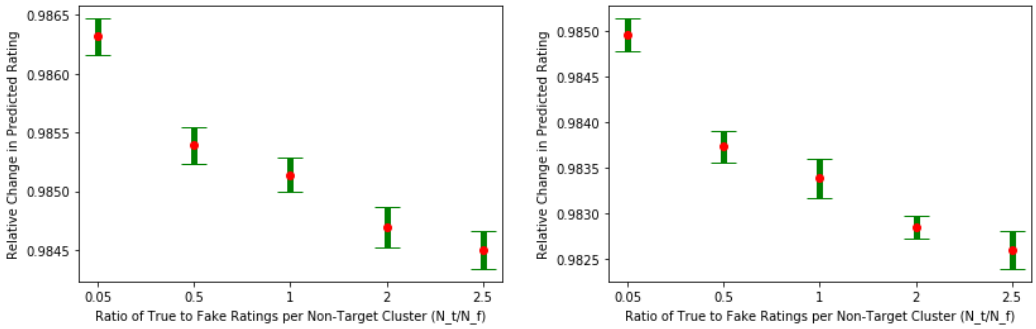
Let us look at the update mechanism here. Since true ratings are not exclusively from the target cluster, as predicted by lemma 17 case (2), equation 4.5 is such that terms $a_{ki}u_i^t$ in summation do not have the same signs. This reduces the overall magnitude of term $\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t$ compared to last section where all the terms in the summation had the same sign. Thus even though diagonal terms a_{kk} show a decreasing trend with increasing N_t, the behavior of summation terms may cause $|a_{kk}u_k^t| > |\sum_{i=1,i\neq k}^{d} a_{ki}u_i^t|$ even at higher ratios for some features, resulting in the reported greater relative change in predicted rating at these ratios. For example, note in Table 4.2b how feature $v_k^{j^*}$ especially corresponding to larger magnitude u_k^t ($k = 0, 1, 4$ in ML and $k = 2, 3, 4, 6$ in GR), result in large magnitude of $\hat{v}_k^{j^*}$ despite increasing ratio. This contrasts our observation from Table 4.2a.

Leakage of Attack to Non-Target Clusters:

Similar to in Figure 4.6, we expect to see leakage effect in non-target clusters. This is illustrated by Figure 4.8 which breaks down the change in rating by clusters when cluster

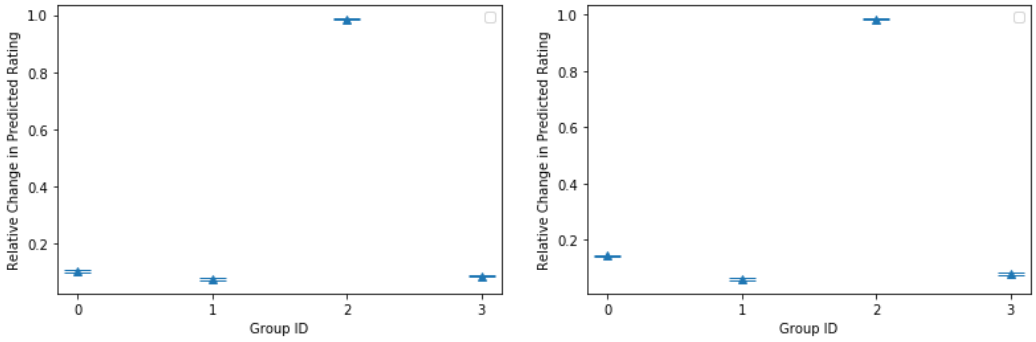
2 is targeted in Movielens and Goodreads datasets. You can see that the attack's impact on non-targeted clusters is lower here than suggested by Figure 4.6. This is because compared to the last section where all positively correlated features across clusters underwent case $2a$ mechanism, due to properties of A^{-1} here, $\hat{v}_k^{j^*} - v_k^{j^*}$ corresponding to many positively correlated cluster features (e.g.: $k = 3, 5, 6, 7$ in ML and $k = 9$ in GR) undergo case $2b$ mechanism causing a shift in directions dissimilar to not only u_k^t but all corresponding k^{th} features across non-target clusters $g \in \mathcal{G} \setminus t$.

Thus in these two datasets, the properties of U, V values are such that unless true ratings also come from the target cluster, it is tough to reduce the effect of attack within the target cluster.



(a) Movielens (b) Goodreads

Figure 4.7: Plot comparing the relative change in the predicted rating in target cluster 2 against increasing ratio $\frac{N_t}{N_f}$ per non-target cluster



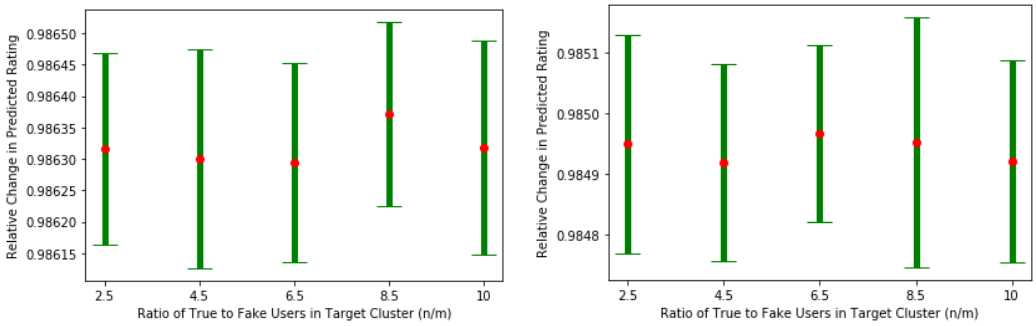
(a) Movielens (b) Goodreads

Figure 4.8: Plot comparing the relative change in predicted rating across clusters against ratio $\frac{N_t}{N_f} = 0.05$ per non-target cluster.

Varying Ratio of True Users (n) to Fake Users (m) in Target Cluster

In this section, we investigate the impact of the target cluster's true user size (n) relative to the fake user size (m) on the effectiveness of the attack.

For this experiment, we fix the distribution of ratings as $\frac{N_t}{N_f} = 0.05$ and $N_f = m = 100$ since it resulted in the maximum change in relative rating in Figure 4.5. Then we increase the ratio of $\frac{n}{m}$ from 2.5 to 10. Figure 4.9 reports the mean and standard deviation of change in the target item's predicted rating when cluster 2 is targeted against increasing values of $\frac{n}{m}$. We can see how the increasing target cluster size does not affect the attack's impact. The relative change in predicted rating after the attack remains almost the same as expected.



(a) (b)

Figure 4.9: Plot comparing the relative change in the predicted rating in target cluster 2 against increasing target cluster size n

4.6.3 Discussion

The objective of the targeted attack study on MF-based RS is to gain a better understanding of how attacks propagate and how the individual behaviors of feature matrices U and V contribute to this propagation. By holding one feature matrix constant and updating the other, we isolated the effects of attacks on each matrix and observed how these effects translate to the targeted promotion of items in RS. We can use the observations outlined to make updates to RS more robust.

We concluded that U_i values corresponding to all true users i in a RS are unaffected by the injection of fake ratings when only matrix U is updated to incorporate the new ratings, keeping V constant. The group feature vector $\tilde{U}_t$ of the target group t is affected due to the presence of fake users in the target group after the attack. This effect on $\tilde{U}_t$ decreases when the number of true users in the target cluster increases. Based on these observations, one possible defence mechanism to reduce the changes to a cluster centre might be introducing dummy users into any low populated cluster g with their U_i vectors set close to cluster centre $\tilde{U}_g$. This may help reduce the impact of an attack by making it harder for injected fake users to shift $\tilde{U}_g$ while not affecting the predicted ratings of the

cluster. However, adding such users may lead to a higher computational cost. Therefore, assigning weights to existing users based on their reliability could be a more efficient solution. Studies of this nature that introduce a weighing factor of trust, reputation, or reliability of users in RS have been explored in [O'Donovan and Smyth, 2005, Resnick and Sami, 2007].

Further, we noted a reduction in the impact of attacks when V remains constant. It is the target item vector V_{j*} that is more sensitive to the injection of fake ratings and serves to leak attack effects to all the clusters in RS when updated. Consequently, updates to latent feature matrices need not be performed frequently together. By updating matrix V less frequently and updating only the U matrix when new users enter, we can decelerate the propagation of attack effects across user clusters and allow monitoring for suspicious trends in item's ratings or feature vectors over time.

Another way to protect the columns of V matrix from possible attacks is by increasing the true ratings to items i.e. providing dummy ratings to items rated less frequently in a cluster might provide immunity against malicious alterations to their item vectors. Ratings for such items may also be sourced from trusted user communities, such as professional critics, trusted user representatives [Liu et al., 2011b, Shi et al., 2017b], or filter bots [Good et al., 1999].

Alternatively, prioritising strategies that detect outlier data and subsequently remove them from feature matrix re-training, or adopting robust matrix factorization methods that are insensitive to outliers can also potentially enhance the resilience of the feature vectors against attacks. For example, it is well known that the L2 norm minimization in equation 4.1 is sensitive to outliers in the data. In statistics, there are different strategies to make estimators more robust to outliers (and hence attacks). For instance, the conventional L2 norm in regularisation can be substituted with the L1 norm, in which the errors are not squared, so the impact of large errors is reduced. This was first suggested by [Croux and Filzmoser, 1998]. Further study of robust MF under L1 norm can be found in [Ke and Kanade, 2005, Eriksson and Hengel, 2010, Zheng et al., 2012, Wu et al., 2020]. Alternatively, [Xiong et al., 2011] proposes a constrained optimisation problem that excludes the outliers from the effort of low-rank approximation, subject to the assumption that the number of outliers is constrained. Similarly, [Mehta et al., 2007] study robust statistical methods, in particular M-estimators, to generate stable recommendations in the presence of noise and spam. Thus, a potential defence mechanism could be exploring

robust MF similar to the aforementioned methods.

For our analysis, we examined a single update step for either the U or V matrix when new users join the system. We anticipate that the analysis can be extended during the model's alternating updates until convergence.

As we discussed in Section 4.4.1, U_i relies on the item vector V_j and the rating value $R_{i,j}$ of all items $j \in \mathcal{V}(i)$ rated by user i. In the scenario of a single update, the U_i values for genuine users remain constant since V is fixed. However, when we continue alternating updates until convergence, subsequent updates to U_i involve the most recently updated V, potentially leading to changes in U_i for users. These changes are expected to be modest given that U_i depends on all items rated by user i, and since the target item j^* is just one item in the set, the alterations may not be substantial.

However, the update of V_{j^*} for a target item j^* can result in significant alterations in response to fake ratings because V_{j^*} relies on all the ratings received by item j^* as discussed in Section 4.4.2. Consequently, after convergence, both U_i and V_{j^*} are expected to undergo modification, with V_{j^*} experiencing the most notable changes comparatively.

4.7 Conclusion

This work studied the effect of user-cluster targeted data poisoning attacks on an MF-based RS by evaluating the changes after the attack on user and item feature matrices U, V. We analysed the mechanism of how U and V matrices change after the injection of fake ratings and how these changes help propagate targeted attacks. We also illustrated our findings using two real-world datasets. We further showed that the effectiveness of an attack on a target item's feature vector is influenced by the distribution of the number of ratings received by the target item. We find that items with fewer ratings in the target cluster are more susceptible to attack.

We conclude that a simple attack with limited knowledge of user preferences suffices to target a specific user cluster precisely. In the next chapter, we use these observations to research defensive approaches in MF-based RS that are simple yet effective and can be easily used in existing systems.

Chapter 5

Attack Detection Using Item Vector Shift in Matrix Factorisation Recommenders

5.1 Introduction

In the previous chapter, we examined the susceptibility of MF-based RS to targeted data poisoning attacks. Building upon the insights gained from the analysis, in this chapter, we investigate strategies for enhancing MF's resistance to such poisoning attacks.

Recall that in data poisoning attacks, an adversary creates fake profiles with carefully crafted ratings for items and attempts to target an item to increase/decrease the item's rating, thus making the item more/less likely to be recommended by the system [Lam and Riedl, 2004]. We assume that the attacker can only inject a limited number of fake users. Each fake user rates a limited number of items (including the target item and other non-target items called filler items) to evade suspicion. To detect these attacks, various approaches have been explored over the years. For example, exploiting features of attack profiles to distinguish "attack" profiles from "regular" profiles [Williams et al., 2007, Chirita et al., 2005], assigning reputation scores to users in RS [Resnick and Sami, 2007] and analysing rating distributions of items in time [Gao et al., 2015, Zhang et al., 2006] are some of the approaches explored to make RS robust.

Recall that in the typical paradigm of MF in RS, items are recommended based on the proximity of the item vectors to the user vectors in the latent space. The findings from

previous chapter indicates that when a targeted attack, such as a push attack, is launched on a particular item, it will cause a shift of the item vector in the low-dimensional space, impacting the item's recommendation. Success of the attack depends on how much it can shift the item vector distribution significantly. Based on this observation, we propose a detection approach that examines the shift in item vectors.

We know that the effect of fake ratings is largest in groups, while individual fake ratings have negligible effects, especially in small numbers [Mehta and Nejdl, 2009]. As a result, eliminating clusters of fake ratings rather than individual ones makes sense. If the feature vector of an item changes significantly after a block of ratings enters the RS, we can suspect that the ratings are fake and avoid using them to train the U, V matrices. In addition, many literary works assume that the target attack rating is always the highest value on a scale. An attacker can avoid detection based on such assumptions by simply providing a rating value one step lower [Williams et al., 2006]. The proposed detection method significantly improves the identification of these obscured attack tactics highlighting the importance of considering rating variability in shilling detection methods.

In this chapter, we present a novel detection method that is based on the deviation in the item vector and takes into account an item's overall preference information. With only $20-25$ true ratings per user cluster required during training, the experimental results achieve a superior detection precision over existing state of the art detection strategies.

5.2 Related Work

Various shilling detection methods have been proposed that defend by detecting and removing fake profiles in RS, focusing mainly on extracting the signatures of authentic user profiles. The generated profiles will deviate statistically from those of authentic users and several attributes for detecting these anomalies are examined in [Burke et al., 2006, Williams et al., 2007, Chirita et al., 2005]. But a major obstacle in user focussed detection method is that it can recognize the distinctive signature of only the known attack model. For newer or hybrid attack models, the detection methods may fall short. In [Mehta and Nejdl, 2009], authors exploit the similarity structure in shilling user profiles to separate them from normal user profiles using unsupervised dimensionality reduction methods. While the approach works well for attacks that show a good correlation among attack profiles, [Cheng and Hurley, 2009] discusses effective attack strategies dropping

the assumption of high similarities among malicious attack profiles. They demonstrate that dimensionality reduction-based detection methods cannot detect such low-diversity attacks accurately. Detecting shilling attacks in a model-free approach involves identifying abnormalities in the rating distribution of items as discussed in [Bhaumik et al., 2006, Zhang et al., 2006, O'Mahony et al., 2006, Yang et al., 2018]. For example, [Zhang et al., 2006] uses item anomaly detection based on sample average and sample entropy in a time series and only tested their method on dense items with at least 500 ratings. Authors in [O'Mahony et al., 2006, Yang et al., 2018] look at the deviation of target item ratings from predicted ratings to identify abnormal ratings. Other works such as [Mehta and Nejdl, 2008, Hidano and Kiyomoto, 2020, Resnick and Sami, 2007, Deldjoo et al., 2021] aim to build manipulation-resistant RS to limit the damage from injected fake ratings.

To the best of our knowledge, no previous work has investigated leveraging item preference vectors to detect attacks in MF-based RS. This detection approach provides the next step by performing well against obfuscated and unobfuscated attacks while requiring a little amount of training data.

5.3 Item Vector Shift Based Detection Model

5.3.1 Utilizing Item Vectors for Improved Anomaly Detection

Given a target item j^* and its item vector V_{j^*} based on initial rating information, we can recursively compute updates to V_{j^*} as we discussed in the previous chapter [1]. i.e. let $\hat{V}_{j^*}$ be the updated V_{j^*} after receiving a block of m new ratings, i.e. let a block of m users with feature vectors in $X \in \mathbb{R}^{d\times m}$ and target item ratings in vector $y \in \mathcal{R}^{m\times 1}$ enter RS. Given $A^{-1} = \left(\sum_{i\in\mathcal{U}(j^*)} U_i U_i^T + \lambda I\right)^{-1}$ where $\mathcal{U}(j^*)$ is the set of all true users who initially rated item j^*, we have,

$$\hat{V_{j^*}} - V_{j^*} = A^{-1}X\left(I + X^T A^{-1} X\right)^{-1}\left(y - X^T V_{j^*}\right) \qquad \text{(7.5 revisited)}$$

The updated position of the item vector in the latent space given by equation 7.5 depends on three factors: the initial user preference vectors of users who gave ratings (represented by matrix A), the feature values of users who provide the new ratings (represented by X), and the deviation factor (represented by $y - X^T V_{j^*}$).

[1] Proof of Theorem 2 in Appendix B, Section 7.2.3

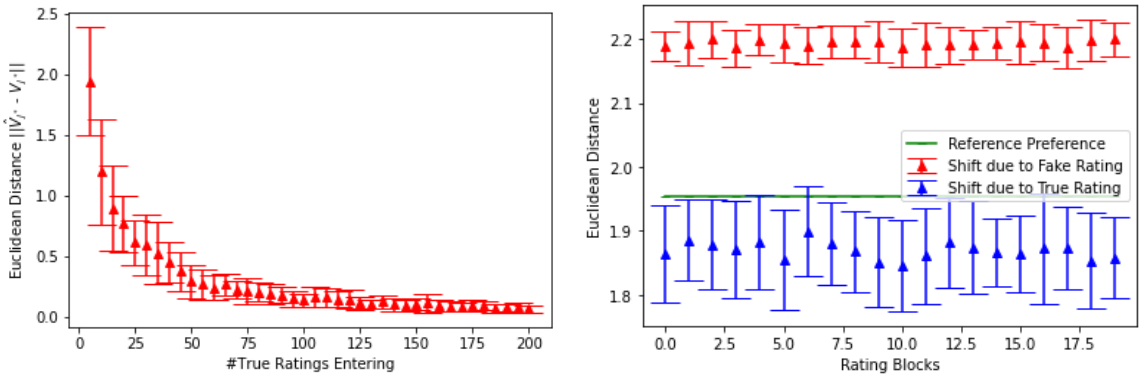
The term $y - X^T V_{j^*}$ measures how much of the new rating vector (y) for the target item can be explained by the predicted rating ($X^T V_{j^*}$), which helps to distinguish between fake and real ratings. If $X^T V_{j^*}$ can already explain y, then the term is zero and $\hat{V_{j^*}}$ is equal to V_{j^*}. However, a significant deviation is anticipated when compared to ratings from actual users because fake ratings are typically set to the maximum or nearly maximum value on the rating scale. If there is no significant deviation, the target item already has a high rating and is not worth further attacking. In addition to the deviation factor, the shift also depends on matrix X. The knowledge that the attacker has access to determines the feature vector values of attack profiles (X). Thus the shift to the item vector also depends on the information available to the attacker.

To illustrate how attacks may shift the target item vector, suppose true users have $U = [1, -1]$ or $U = [-1, 1]$. i.e. the first set of users like items with $V = [1, 0]$ and dislike items with $V = [0, 1]$, but the second set of users are the opposite. Then an attack against the first set of true users keeping $U = [1, -1]$ constant to increase the rating of an item with $V = [0, 1]$ may be performed by shifting $V = [0, 1]$ to $V[1, 0]$. Then such a shift, while it increases the rating in users with $U = [1, -1]$, would decrease the rating in group $U = [-1, 1]$. So the attack effects leaks to all clusters due to the changes to item vector V_{j^*}. The approach that is adopted in this work is to consider the consistency of the preference of user clusters to the target item vector.

Utilising item feature vectors for anomaly detection provides a distinct advantage over existing methods. Unlike other methods that rely solely on attack profile signatures or analysing item rating distributions, the item feature vectors capture the effect of both of these factors, promising a more robust and effective approach to detecting shilling attacks in RS.

5.3.2 Proposed Item Vector Based Detection (IVD) Method

Consider that the target item j^* initially received no ratings in the RS. Then, at random, we select a group and a user from that group to offer a rating. The iterative change in distance (mean and standard deviation) between the updated $\hat{V_{j^*}}$ and the last updated V_{j^*} after each block of 5 true ratings is depicted in Figure 5.1(a). We see that the item vector shifts less from its previous value in response to new ratings after roughly 100 in total true ratings, indicating that the item's preference has been established and that additional true ratings contribute little new preference information.



(a) Effect of Increasing True Ratings on V_{j^*} (b) V_{j^*} Deviation from True, Fake Ratings

Figure 5.1: Plot illustrating the IVD method in MovieLens 100k dataset

Given a target item j^*, we randomly select a cluster g as a reference in the d-dimensional vector space. We hypothesize that further real ratings will not significantly push the vector V_{j^*} away from cluster g, because true ratings rarely generate large changes in current preferences, as previously demonstrated. We suspect that the ratings are fraudulent if any block of new ratings leads the item vector to deviate significantly from reference cluster g. Significant deviations from the reference cluster may indicate that the new ratings are attempting to manipulate the recommendations by shifting the item vector into a different region of the vector space.

To demonstrate this concept, Figure 5.1(b) computes the distance ($D_{g,\hat{j}^*}$) between $\hat{V}_{j^*}$ to cluster g and compares it to the distance (D_{g,j^*}) between initial V_{j^*} and cluster g for each false (indicated by red plot) and true rating blocks [2] (indicated by blue plot). As we can see, each block of fake rating results in a larger shift away from the initial distance D_{g,j^*} (reference preference information: green line), indicating a shift in preferences, whereas each block of real rating results in shifts relatively close to the reference preference D_{g,j^*}.

This observation allows us to recognise and eliminate fake ratings from the RS. Note that the model can continue to recommend to suspected or flagged users. Thus, only suspected ratings are removed before the RS retrains the U, V to generate an updated prediction matrix.

[2] We select a block of 20 new ratings because attack sizes less than 1% do not lead to significant changes in the item vector, as we will elaborate later in Section 5.5.1.

Algorithm 3: Item Vector Based Detection Algorithm

Input: V_{j^*}, Cluster g, Reference Euclidean Distance $D_{g,j^*}(\tilde{U}_g, V_{j^*})$, Threshold $=th$

```
for block index n = 1, 2, 3.... do
    Find updated item vector V̂_{j*}  from eqn 7.5
    Calculate D_{g,ĵ*} between V̂_{j*} and Ũ_g;
    if Dg, ĵ* > D_{g,j*} + th then
        Remove the new block of ratings from the RS training process
    else
        Keep the new block of ratings for RS training process
```

5.4 Experiments

5.4.1 Datasets

We evaluate the effectiveness of the attack on the MovieLens dataset (943 users rating 1682 movies, contains 100000 ratings from 1-5) which is widely used across literature for evaluating recommender systems under attack. Additionally, we use larger Movielens dataset (6040 users, 3706 movies, contains 1 milion ratings from 1-5) and a Netflix dataset. We take a dense subset of the Netflix dataset by selecting the top 1000 items and the top 10000 users who rated these items.

5.4.2 Evaluation Setup

We assume that the datasets considered have no existing fake profiles, thus the attack profiles we add are the only fake profiles present. To a trained MF-RS, fake users are added which all target the same item which is selected at random. Fake users are generated using the well studied Average, Random and Target attack models. These new ratings were not part of the initial training data, and detection algorithms aim to identify and remove the fake ratings, (and sometimes the fake profiles themselves), to prevent their inclusion in the subsequent training update.

5.4.3 Attack Model

The attackers give high ratings to the target item to make it more visible, and a set of items are rated to create a fake user profile that mimics actual user behaviour. The rating

distribution of these filler items determines the type of attack. We report results for two standard attack profiles mentioned in the literature: Random Attack and Average Attack [Lam and Riedl, 2004]. From the discussion in Section 2.3.1, the *Random Attack* is a zero-knowledge attack where ratings to filler items in each rating profile are distributed around the overall mean of items and the *Average Attack* distributes the ratings for filler items in attack profiles around the mean for each item.

Additionally, we look at the sophisticated *Target-Cluster Attack* [Shams and Leith, 2022], where attackers target a specific user cluster [3] by sampling filler ratings from a Gaussian distribution using the mean rating of items in the RS. This is a reasonable assumption since such aggregate user preference information can often be obtained from publicly available sites as we discussed earlier in Section 4.3.2.

Target Item Ratings

We consider items with ground truth average empirical rating < 3.5 in all user clusters and with atleast $20 - 25$ true ratings per user cluster, and attack profiles assign a maximum rating value to promote it in the RS. To make the detection harder, we also report results when using the target-shifting obfuscation technique [Williams et al., 2006], which involves shifting the rating given to the target item from the maximum rating to a rating one step lower.

Additionally, we also evaluate the effect of the choice of filler items. Choosing filler items appropriately makes the generated profiles similar to the genuine profiles making it harder to spot fake users.

5.4.4 Baseline Detection Approach

In a series of experiments, the presented method's detection performance is compared with the following baselines. Here is a brief description of the methods used:

- PCA-based: This method involves transforming the user-item matrix into a lower-dimensional space using Principal Component Analysis (PCA). Each user profile is then represented by three principal components. By analyzing the proximity of profiles to the new space's origin, potential attackers can be identified as users exhibiting suspicious behavior. We fix the $r = 10\%$ i.e. the top 10% of the users given by PCA is suspected as fake users. [Mehta and Nejdl, 2009].

[3] We randomly choose $g = 2$ to target. Results are similar for any chosen g

- MPE-based: This method utilizes the mean prediction error (MPE) for individual items. It calculates the deviation between predicted ratings and actual ratings to identify anomalous rating behavior. If the item has significant MPE values, potential anomalies in the user ratings to the item can be detected [Yang et al., 2018]. We fix the threshold as 1.5

In Section 5.3.1, we discussed how IVD captures both the signatures of attack profiles and the deviation in item ratings, offering a more holistic approach to attack detection in RS. Several existing approaches discussed in Section 5.2 concentrate solely on one of these factors. They either utilize user profile features to discern fake profiles or concentrate solely on changes in item ratings. Among these, we omit comparisons against supervised approaches for fairness. Instead, we focus on unsupervised methods and select two baselines known for their high detection accuracy and stability against various filler and attack sizes. The baselines selected, each focus on factors corresponding to either user profiles or item deviation, to compare against IVD's performance. For instance, PCA assesses the overall user profiles, while MPE examines the deviation in target item ratings. This comparison allows us to understand how IVD overcomes the limitations of relying solely on a single detection factor and bridges the gap in these approaches.

5.4.5 Evaluation Metric

We evaluate detection performance using two metrics: detection rate and false alarm rate. The detection rate is calculated by dividing the number of times of detected attack ratings by the total number of times fake rating blocks were inserted. The false alarm rate, on the other hand, is determined by the number of true rating blocks predicted as anomalies divided by the total number of true rating blocks inserted. The reported results are an average over 50 randomly chosen target item from the set of potential target items. We set the threshold to $th = 0.07$ for IVD

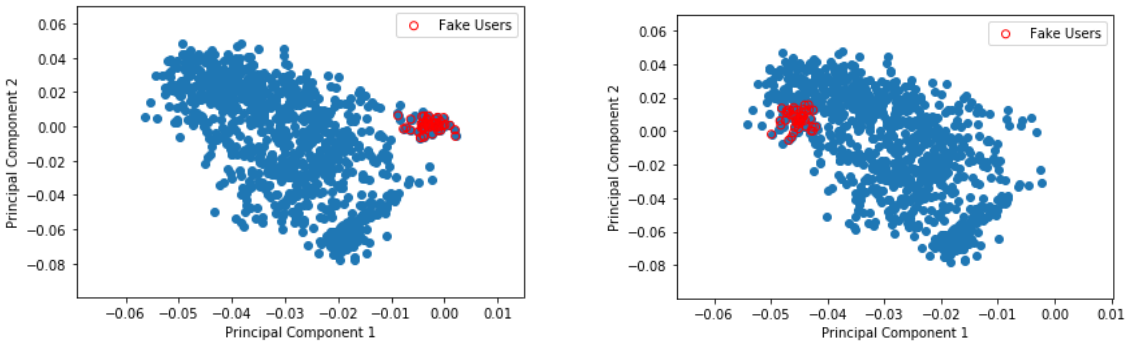
5.5 Results and Discussion

In this section, we compare the performance of IVD against two state of the art detection strategies that reports a high detection rate and very low false alarm rate, namely PCA method [Mehta and Nejdl, 2009] and MPE method [Yang et al., 2018]. Both methods are stable against attack and filler sizes with $> 90\%$ detection rates as we will see later.

But both approaches have their limitations. For example, PCA works by observing that the profiles of shillers are very similar. They interpret users as variables (i.e. the dimensions of the data are the users, and the observations are the item ratings), then we have data where a number of dimensions are very similar and PCA can find a set of variables (users) which are highly correlated. Thus an attack profile that is undetectable by the PCA detector must reduce the differences from genuine profiles. The key factor here is in filler selection. Choosing filler items appropriately can bring down the correlation among fake users making them similar to other true users in the RS. When filler items are selected randomly, they are likely to have smaller pairwise overlaps compared to genuine users. Genuine users are more likely to rate certain items more frequently than others. Based on this observation, study by [Cheng and Hurley, 2009] has shown that an attacker should choose filler items according to their overall popularity among the genuine user base. A simple and effective strategy to obfuscate attacks is to choose filler items with equal probability from the top x% of most popular items, rather than from the entire catalogue of items, to make detection harder.

In Figure 5.2, we plot the users in RS in the PC space. The coordinates here are the coefficients of each user in the 1st and 2nd principal component. Figure 5.2(a) shows how, due to their low PC scores, the fake users are focused around the origin in the PC space. To identify fake users, PCA method assumes the two coefficents of each user represent a point in space and sort the points in order of their distance from origin. The top r% eliminated users will contain fake users with more than 90% accuracy. The PC scores are difficult to distinguish when smaller subsets of popular item are used as filler items as can be observed from Figure 5.2(b) and thus fails to identify fake users. Figure 5.4(a) shows how the detection rate of PCA against standard average attack. The detection rate decreases as the selection of most popular items is reduced to smaller subsets, with 0% detection for $x \leq 20$. Although not demonstrated separately, we note that PCA shows no perceivable drop in detection accuracy in the face of target rating obfuscation as discussed in [Mehta and Nejdl, 2009]. This is because these are linear transformations, which are not very effective when linear dimensionality reduction is performed.

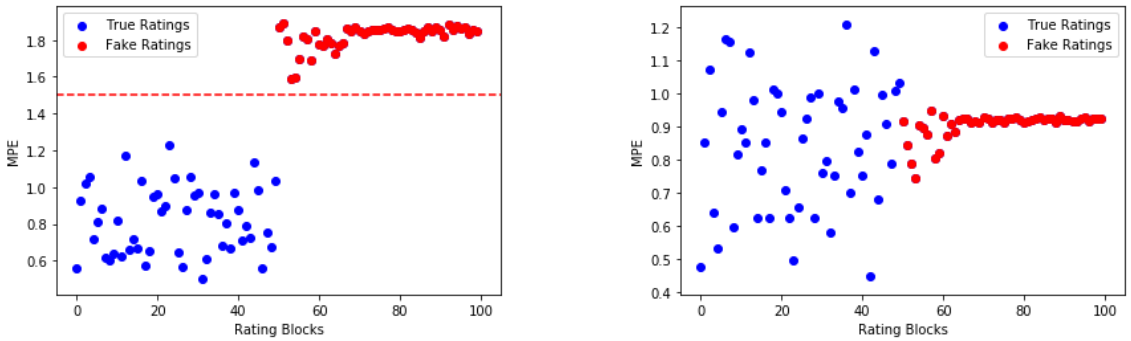
MPE, on the other hand, is more concerned with deviations in item predicted ratings than with user profiles. It is relatively resistant to filler selection strategies and produced good results in the studies. The capacity of MPE to identify tiny changes in item predicted rating deviation made it a desirable tool for attack detection.



(a) PC Space under Random Filler Choice

(b) PC Space Under top 20% Filler Choice

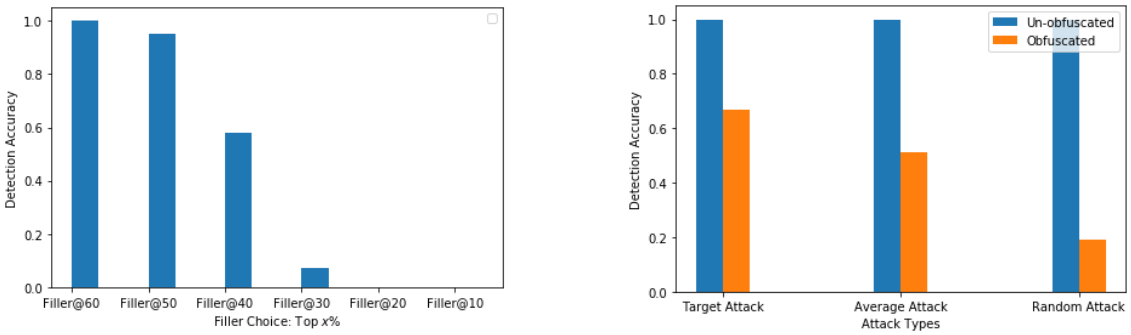
Figure 5.2: Clusters in 2D Space for Normal Users and Fake Users (Avg Attack) in ML-100k



(a) Target Item Rating: Maximum

(b) Target Item Rating: Obfuscated

Figure 5.3: Distribution of MPE for Genuine and Fake Rating Blocks in ML-100k



(a) PCA Performance Against Filler Choice Obfuscation

(b) MPE performance against Target Rating Obfuscation

Figure 5.4: Detection Accuracy for PCA and MPE in ML-100k

The assumption is that attackers just focus on the target item and rate it with the maximum or lowest rating many times in order to promote or demote the item to the recommendation list. The simple approach of obfuscating the target item rating from a maximum value to a value one step lower, on the other hand, leads the prediction error

to be similar to that of the genuine user base and fails to detect the fake activity.

The MPE for average attack with and without target rating obfuscation approach is plotted in the Figure 5.3. Figure 5.3(a) depicts how the fake ratings form a cluster away from true users due to their significant deviations. However, when target rating obfuscation is used, the MPE of false users becomes difficult to discern from that of legitimate users as is demonstrated in Figure 5.3(b). Figure 5.4(b) compares the performance of MPE against standard attack model and when target rating is obfuscated for a fixed filler choice of top 20%. We can see that the detection rate decreases when compared to 100% detection in un-obfuscated attacks. Thus, the choice of filler items has no effect on MPE, but the target rating obfuscation does since without obfuscation, MPE results in perfect detection for the $x = 20\%$ filler choice.

In the further sections, we compare how IVD performs under these obfuscation schemes to PCA and MPE approaches, and we give findings for three different datasets. We demonstrate that IVD performs more effectively than both techniques under these obfuscation strategies.

5.5.1 Effectiveness of Attack Size and Filler Size

In this experiment, we simulate an attack by introducing fake profiles alongside the regular user profiles in the recommender system. We examine how the performance of the IVD method is affected by varying attack and filler sizes using the ML-100k dataset. The findings from this experimentation hold true for larger datasets such as Movielens 1M and Netflix. The focus on the 100k ML dataset allows for direct comparison with previously reported results, as this dataset has been widely used to evaluate prior shilling detection methods.

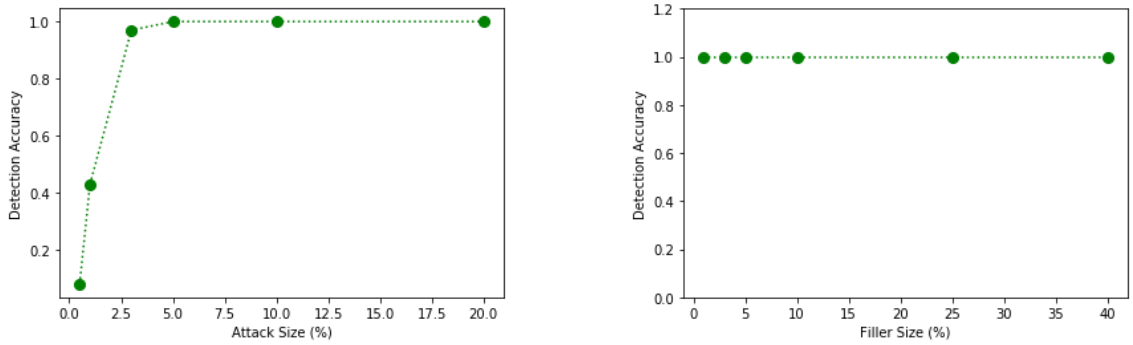
We consider diverse attack sizes (0.5%, 1%, 3%, 5%, 10%, 20%) and select the filler items randomly from the top 60% of most popular items in the RS. We fix the number of filler items to 10% of the total item count. In Figure 5.5(a), we observe that for attack size $< 1\%$, IVD fails to detect the presence of fake ratings. Detecting an attack's impact is challenging when attack sizes are small. This is because such low attack sizes are unable to alter the item vector sufficiently to enhance the target item's rating. As a result, the attack's effect on the target item is insignificant. Because IVD directly gauges attack power, small and weak attacks are difficult to detect.

In Figure 5.5(b), we consdider various filler sizes (1%, 3%, 5%, 10%, 25%, 40%) for

a fixed attack size of 5%. We can see that altering filler size has little effect on IVD performance.

Although we don't report separately, the PCA and MPE methods demonstrate almost flawless detection across diverse attack and filler sizes, corroborated by both the datasets and findings in [Mehta and Nejdl, 2009, Yang et al., 2018]. It's noteworthy that their detection accuracy is influenced by filler selection and rating obfuscation respectively, as elucidated in earlier sections.

Based on these findings, for the experiments, we fixed an attack size of 5% and a filler size of 10% for the rest of the analysis. Attack sizes below 1% result in considerably insignificant changes, rendering them uninteresting from the perspective of attack effects on targeted item.



(a) Detection Rate for varying attack sizes. Filler size=10%

(b) Detection Rate for varying filler sizes. Attack size=5%

Figure 5.5: Effect of Attack and Filler Size IVD for ML-100k

5.5.2 Effect of Choice of Filler Items

Recall the discussion that the PCA defense strategy relies on the assumption that attackers are more similar to each other than to the genuine user base. We choose filler items with equal probability from the top x% of the most popular items. We report the detection rate for the three methods against reducing the value of x from the top 60% to the top 10% of the most popular items and compare the performance. Table 5.1, 5.2 and 5.3 report the results for Average, Random and Target Attack respectively.

As expected, the PCA accuracy decreased significantly with increasing x% due to the increase in similarity among attack profiles and genuine profiles. As demonstrated earlier, detection becomes hopeless for x values less than 20% for all datasets. When filler items are selected from the popular items, their preferences become more diverse and closer to other genuine users who also rated these popular items and thus are less likely to form

a well-defined cluster. This leads to a reduced accuracy of PCA as it fails to effectively distinguish attackers from genuine users.

However, both the IVD and MPE strategies still demonstrate high accuracy in comparison to PCA as discussed previously. This is because these detection methods are not reliant on the correlation between fake profiles. Specifically, MPE focuses on the deviation from the point of view of item predicted ratings rather than user profiles. As a result, it is not significantly impacted by the choice of filler items, making it more robust against variations in filler selection. Similarly IVD focuses on the deviation of the item feature vector to new ratings. Fake user profiles have no direct impact on the item vector deviation calculation 4.4.2. Rather than working with the profiles directly, IVD employs user vector U of users who provide ratings to compute the updated item vector.

Defense Strategy/Dataset	Filler			
	filler@60	filler@40	filler@20	filler@10
IVD/ML-100k	**1**	**1**	**1**	**1**
MPE/ML-100k	1	1	1	1
PCA/ML-100k	1	0.58	0.0	0.0
IVD/ML-1M	**1**	**1**	**1**	**1**
MPE/ML-1M	0.96	0.97	0.95	0.94
PCA/ML-1M	0.96	0.31	0.0	0.0
IVD/Netflix	**1**	**1**	**1**	**1**
MPE/Netflix	1	1	1	1
PCA/Netflix	1	0.98	0.25	0.0

Table 5.1: Detection Rate for Different Defense Strategies and Datasets for **Average Attack** over top x% filler. Attack size=5%, Filler size=10%

Defense Strategy/Dataset	Filler			
	filler@60	filler@40	filler@20	filler@10
IVD/ML-100k	**0.84**	**0.85**	**0.97**	**1**
MPE/ML-100k	1	1	1	1
PCA/ML-100k	1	0.83	0.0	0.0
IVD/ML-1M	**0.87**	**1**	**1**	**1**
MPE/ML-1M	1	1	1	1
PCA/ML-1M	1	0.63	0.0	0.0
IVD/Netflix	**1**	**1**	**1**	**1**
MPE/Netflix	1	1	1	1
PCA/Netflix	1	1	0.43	0.0

Table 5.2: Detection Rate for Different Defense Strategies and Datasets for **Random Attack** over top x% filler. Attack size=5%, Filler size=10%

Defense Strategy/Dataset	Filler			
	filler@60	filler@40	filler@20	filler@10
IVD/ML-100k	**1**	**1**	**1**	**1**
MPE/ML-100k	1	1	1	1
PCA/ML-100k	0.93	0.58	0.0	0.0
IVD/ML-1M	**1**	**1**	**1**	**1**
MPE/ML-1M	0.80	0.89	0.83	0.88
PCA/ML-1M	0.94	0.25	0.0	0.0
IVD/Netflix	**1**	**1**	**1**	**1**
MPE/Netflix	1	1	1	1
PCA/Netflix	1	1	0.25	0.0

Table 5.3: Detection Rate for Different Defense Strategies and Datasets for **Target Attack** over top x% filler. Attack size=5%, Filler size=10%

5.5.3 Effect of Target Shifting Obfuscation

In this section, we will look at the effects of using target-shifting obfuscation in attack techniques. For the experiment, we fix the filler selection to the top 20% of the items in the datasets. The detection rates are shown in Table 5.4.

In terms of shilling detection rate, the IVD technique surpasses both MPE and PCA. As previously noted in the previous sections, the reduced performance of PCA is primarily attributable to the selection of filler items. In contrast, MPE performance suffers because, when fraudulent users use target rating obfuscation, their MPE values may closely mirror those of real users, making differentiation difficult as demonstrated in Section 5.5. However, because IVD takes into account both the user vector of raters and the rating deviation, it is more sensitive to changes from attacks even after target-rating obfuscation.

Defense Strategy/Dataset	Attack Type		
	Average	Random	Target
IVD/ML-100k	**0.86**	**0.65**	**0.90**
MPE/ML-100k	0.51	0.19	0.67
PCA/ML-100k	0.0	0.0	0.0
IVD/ML-1M	**0.86**	**0.60**	**0.83**
MPE/ML-1M	0.55	0.29	0.42
PCA/ML-1M	0.0	0.0	0.0
IVD/Netflix	**1**	**0.93**	**1**
MPE/Netflix	0.40	0.03	0.84
PCA/Netflix	0.22	0.40	0.25

Table 5.4: Results for Different Defense Strategies and Datasets under **Target Rating Obfuscation+Filler@20 Obfuscation**

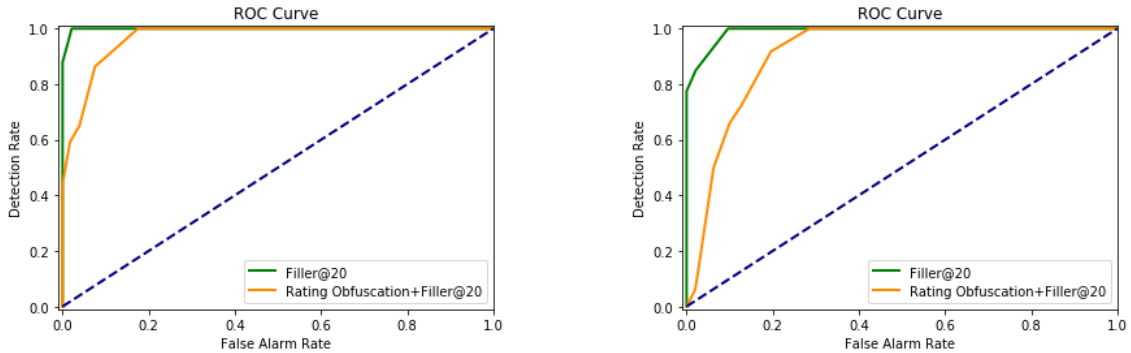
5.5.4 Reciever Operater Characterstics

The Receiver Operator Characteristic for the IVD method applied to two standard attack strategies: Average and Random is depicted in Figure 5.7. We simulate genuine ratings by drawing ratings from the empirical distribution of ratings for a group. When we need a rating for the target item that the user has not rated, pick a second user from the same group who has rated the item and merge the pair of user ratings.

IVD technique is applied to attack scenarios with fixed parameters: the top 20% popular items as filler options, a filler size of 10%, and an attack size of 5%. For each Average and Random attack case, we examine the ROC curve for the 1) maximum target item rating model and 2) target rating obfuscated model.

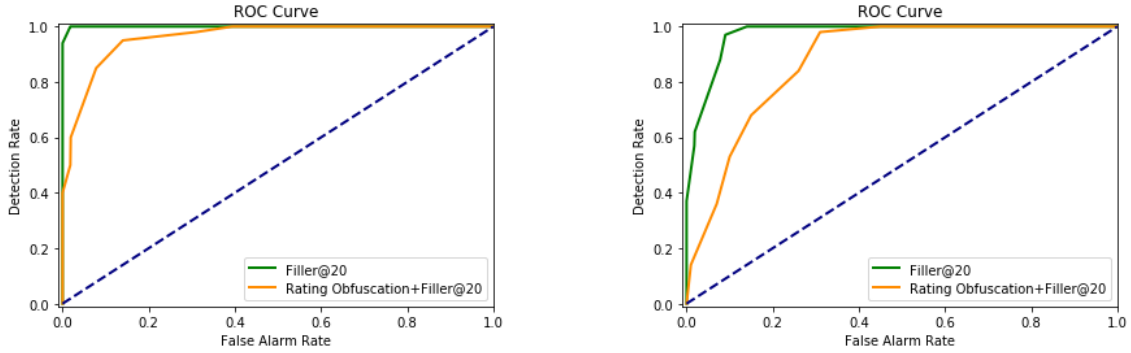
For the un-obfuscated Average and Random attacks, we can see that IVD obtains a 100% detection rate with a false-alarm rate of less than 10% for both datasets.

When we apply rating obfuscation to these attack strategies, we find that detection is achieved at 80% for the average attack and at 60% for the random attack for the same false alarm rate of less than 10%. For the obfuscated attacks in both datasets, a near perfect detection rate happens with a cost of approximately 35% false alarm rate. We can see that IVD performs better at detecting Obfuscated Average attacks than Random attacks. This is to be expected given that IVD directly gauges attack power. Obfuscated Random attacks are more difficult to detect since they are simpler and less effective at shifting the item vector.



(a) ML 100k: Average Attack (b) ML 100k: Random Attack

Figure 5.6: Reciever Operating Characterstics for ML100k Dataset



(a) ML 1M: Average Attack (b) ML 1M: Random Attack

Figure 5.7: Reciever Operating Characterstics for ML-1M Dataset

5.5.5 Discussion and Limitations

In the real-world deployment of RS, it's not always wise to delete user profiles suspected of being fake, as genuine users with specific interests may also be among them. IVD addresses this by excluding ratings from suspected profiles during training while still recommending items to them, ensuring user satisfaction without compromising system functionality.

In a large RS with numerous items, monitoring all items using IVD can be computationally expensive and unnecessary. Instead, vulnerable items (e.g., those with sudden spikes in rating activity, a high number of extreme rating values, new items, or items with few ratings in the RS) can be identified and given additional protection through IVD. Monitoring can cease for sets of vulnerable items once they've received many true ratings because it's difficult and costly for an attacker to change ratings for items with a substantial number of ratings, as seen in the previous chapter. Therefore, IVD can be used alongside time series analysis methods or other rating behaviour monitoring methods to reduce the detection scope. The following are the major limitations of the proposed method:

- Cold Items: The proposed method assumes that initial ratings for new items can be obtained in the RS. This is a reasonable assumption since, ratings for such items may be obtained from trusted sources, such as professional critics, reliable user representatives [Liu et al., 2011b, Shi et al., 2017b], or filter bots [Good et al., 1999]. We could also randomly elicit ratings from users in the system to reduce the risk of sampling fake ratings from false users already existing in RS. For instance,

Goodreads [4] and 'Voracious readers only' [5] give away free copies of new books to readers in exchange for honest ratings. At the same time, movie RS offer rewards for watching new movies and tv series [6]. We further assume that the randomly selected users are real and that the likelihood of all initial users and ratings being fake is low. We note that the RS does not have to wait for $20-25$ ratings per group before recommending the new item to existing system users. We simply propose that the RS avoid using unprompted ratings for the new item in the training process until it receives ratings from reliable sources upto the threshold.

- A reasonable choice of reference and threshold are crucial in determining the performance of the proposed approach. We demonstrated the detection in both a dense dataset and two complete datasets. IVD results in reasonable performance in all the cases. In practice, utilizing a single reference cluster as a broad assumption may yield insights and preliminary findings. However, in order to create a viable fake ratings detection mechanism, additional techniques such as dynamic references, weighted references, or models that learn references adaptively over time can be explored as part of future work. These methods may better capture the intricacies of user preferences, deal with potential changes, and accommodate a wide range of scenarios, resulting in increased detection accuracy.

5.5.6 Conclusions

We present a new approach for detecting shilling attacks that makes use of item preference vectors. The Item Vector Deviation (IVD) technique provides an unsupervised and attack model-free strategy that can be deployed directly to any target item, provided the RS has sufficient rating information about the item in question to compute its initial preference vector. Remarkably, as low as $20-25$ ratings per user cluster were sufficient to produce favorable outcomes across all examined datasets. The experiments confirm the sustained effectiveness of the proposed strategy, achieving high detection accuracy with low false alarms and even performs reasonably well against various obfuscated attack strategies.

[4] https://help.goodreads.com/s/announcements/a031H00000RKE8VQAX/giveaways-for-authors-frequently-asked-questions?ref=kdpnl

[5] https://voraciousreadersonly.com/

[6] https://www.amazon.in/b/ref=as_li_ss_tl?node=15697217031&ref_=dvm_crs_merch_in_ai_slashpv_P_watch&wint2_header&pf_rd_m=A1K21FY43GMZF8&pf_rd_s=merchandised-search-2&pf_rd_r=4N0T0PVXWA9S566NFGQV&pf_rd_t=101&pf_rd_p=51f6c505-5ad6-4b9c-b9b6-91450c336884&pf_rd_i=10882806031&linkCode=sl2&tag=thinkerviewsc-21&linkId=eb862e43198dfa2f8e5a415bb4db6f26&language=en_IN

Chapter 6

Conclusion

Our research focused on two key challenges in RS: the Cold Start Problem and Data Poisoning attacks within the user-clustering framework. We explored utilizing user clustering to address the Cold Start Problem, analyzed its impact on data poisoning attacks, and devised a detection method for robust recommendation systems.

By addressing the Cold Start issue in Chapter 3, we revisited the challenge of making recommendations for new users due to the absence of any historical preference data. We proposed a novel Cluster-based Bandit (CB) algorithm that achieves fast learning in cold-start users. CB suggests that for fast learning we want to initially ask the user to rate those items for which the information to distinguish between a group pair is the largest. Identifying such items called distinguisher items can quickly identify the correct cluster a user belongs to. Once the correct cluster is identified, smart recommendations for new users can be utilized by the collective wisdom of comparable users within the cluster.

Our experimental results demonstrated that for the standard Netflix dataset, < 10 items need to be rated in order to reliably distinguish between 16 user groups and $<$ 12 items to reliably distinguish between 32 groups. We demonstrate that the learning performance is fundamentally superior to that of the state-of-the-art decision-tree and that the group of a user is identified with much higher accuracy than with decision-tree without incurring higher regret or longer learning time. Thus even without extensive user data, our method improves the first user experience and offers good recommendations.

In Chapter 4, we examined the security ramifications of user clustering with regard to data poisoning threats. In order to modify recommendations and affect user behaviour, these malicious attacks entail the injection of skewed or false preference data. Our analysis

showed that MF could be vulnerable to targeted data poisoning attacks where attackers concentrate on particular user clusters. We specifically examined the process underlying how the user and item feature matrices U, V resulting from MF, alter following the injection of fake ratings. We demonstrated how these modifications aid in expanding the reach of targeted attacks by assessing the changes in these user and item feature matrices following the attack. Our findings lead us to the conclusion that the true user vectors in the U matrix of a RS remain unaffected by the introduction of fake ratings. On the other hand, the target item vector in the V matrix plays a pivotal role in propagating attacks throughout clusters, even in cases where those clusters are not the intended targets. Moreover, it is the target item vector that significantly contributes to altering the rating of the targeted item.

Hence, to mitigate the impact of attacks, it becomes crucial to explore methods that limit the modifications to the item vector after an attack. By focusing on strategies to contain changes in the item vector, we can effectively reduce the reach and influence of attacks.

Finally in Chapter 5, we re-visit the robusteness to attacks problem in a MF-based RS. Based on the conclusions from the last chapter, we presented an effective detection method for safeguarding user clusters in recommenders against data poisoning attacks. We presented a novel Item Vector Deviation (IVD) based detection method that is based on the deviation in the item feature vector following injection of set of new ratings. It monitors any user cluster's variation in preference to the targeted item vector that results from the shift in the target item vector following malicious ratings. IVD resulted in superior performance against Filler choice and Target rating-based Obfuscation techniques that brought down the performance of state-of-the-art PCA and MPE attack detection strategies. The proposed algorithm demonstrated promising results ($> 90\%$ accuracy with low false alarm rate for standard attack strategies) in detecting and mitigating attacks with just 20-25 true ratings needed per cluster to train the item vector to the preferences of these user segments.

6.1 Future Directions

The following are a number of problems that extend from the work done in this thesis:

6.1.1 Addressing the User Cold Start Problem by Leveraging User Clusters

In Chapter 3, our focus is on cold start problem specifically from explicit user rating interactions. Online systems also gather data from many sources to produce personalised content. Direct identification techniques, such as IP tracing or browser finger-printing are outside the scope of our current analysis. Implicit profiling effects due to Geo-location, for example, also affect personalised content. Thus investigating how explicit, implicit and other data collection techniques interact is another area for future research.

Further, extending the research to cold start *item* is challenging. This is because, when the set-up is reversed, we cluster items based on the ratings received by these items. When clustering the users, the earlier algorithm estimates the variance and the mean ratings of each item by the users in their respective clusters. The inverted configuration, however, introduces a difficulty. In most circumstances, an item may receive several ratings from the RS, but an ordinary user may only evaluate a few items in the RS. Due to the low amount of user evaluations for each item, mean and variance estimations for an item cluster may be imprecise and depends on the number of items in a cluster. A dense ratings matrix may be needed to increase confidence in results. Therefore, exploring of clustering items based on explicit ratings, along with its associated challenges and limitations, presents an interesting avenue for further investigation.

6.1.2 Evaluating Impact of User-Cluster Targeted Attacks in Matrix Factorisation Recommenders

In Chapter 4, the scope of our target attack impact study was specifically centered around the standard Matrix Factorization (MF) technique. The landscape of RS has changed over time, giving rise to a range of improvements and expansions to the fundamental MF approach. To enhance recommendation quality, these developments frequently combine Neural Networks [He et al., 2017, Huang et al., 2021] and Graph based Networks [Fang et al., 2018].

In particular, analyzing how these extensions of MF behave under targeted data poisoning attacks may offer insightful information about their weaknesses and advantages. It would be fascinating to determine whether these approaches display distinct patterns of susceptibility when compared to conventional MF and how attacks influence their em-

beddings. Additionally, understanding whether the modifications induced by attacks can be effectively mitigated or neutralized in these expanded techniques is a crucial aspect to explore.

6.1.3 Attack Detection Using Item Vector Shift in Matrix Factorisation Recommenders

In Chapter 5, we discussed a new detection method based on Item Vector Deviation (IVD) to make RS more robust. It is also worth noting that the IVD approach has a constraint. The strategy, in particular, may not be useful for new or cold items in the RS. This is because IVD relies on baseline preference data to measure variations and find anomalies. New items, by definition, lack historical preference data, which makes the IVD technique difficult to implement. Addressing this restriction and improving the IVD approach to successfully manage cold items is an appealing direction for future research. This approach would necessitate the techniques for addressing Item Cold Start to incorporate initial preference data for new items and provide a reliable baseline for deviation comparisons.

Undoubtedly, another intriguing extension of the current study may involve investigating the application of attack detection systems based on item embeddings to various Matrix Factorization methodologies discussed previously. The proposed detection method, which takes into account deviations in item vectors and overall preference information, has demonstrated promising results in protecting user clustering-based RS from data poisoning attacks. Extending this approach to advanced MF techniques like Neural Matrix Factorization, Graph based Neural Network frameworks and others would be a promising direction for future research.

Chapter 7

Appendices

7.1 Appendix A

7.1.1 Definitions

DEFINITION 1. *Given a Multi-Armed Bandit (MAB) problem of T time steps and n arms. At each time step, an arm i with expected reward μ_i is selected to play by a policy $\mathcal{A}$. The performance of the policy $\mathcal{A}$ is measured by its (cumulative) pseudo-regret at time T that is given by,*

$$R_T = \sum_{i=1}^{n} \mathbb{E}[I_T(i)]\Delta_i \tag{7.1}$$

where $\mu^* = \max_i \mu_i$ is the best arm with maximal mean reward over all arms, $\Delta_i = \mu^* - \mu_i$ the gap between arm i and the best arm and $I_T(i)$ denotes the number of times arm i was selected up to time T. □

DEFINITION 2. *The random variable $X \in \mathbb{R}$ is called σ^2-subgaussian if $\mathbb{E}[X] = 0$ and there exists a $\sigma > 0$ such that its moment generating function satisfies $\mathbb{E}[e^{cX}] \leq e^{c^2\sigma^2/2}$ for every $c \in \mathbb{R}$.* □

7.1.2 Theorems and Proofs

The goal is to use concentration inequalities to control sums of the form $Z_1 + \ldots + Z_n$ where each Z_n is a random variable related to the rating of some item at turn n.

Assumption 5. *We have the following data*

1. *Families* $\mathcal{Z} = \{Z_1(v_1), Z_2(v_2), \dots Z_n(v_n)\}$ *of real-valued random variables for* $i = 1, 2, \dots, n, \infty$.

2. *Real numbers* $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$ *such that each* σ_n^2 *is bounded.*

such that

(a) *For any* $n < \widetilde{n}$ *the random variables* $Z_n(v_n)$ *and* $Z_{\widetilde{n}}(v_{\widetilde{n}})$ *are independent.*

(b) *Each* $Z_i(v_i)$ *is* σ_i^2*-subgaussian with expectation zero.*

(c) *For any turn* $n \leq \widetilde{n}$ *the event that we rate an item* v_n *on turn* n *given by* $\{\tau(n) = v_n\}$ *is independent of* $Z_{\widetilde{n}}(v_{\widetilde{n}})$.

The above captures the idea of a sequence of items and rewards. i.e. at each turn i, we receive an instance of $\mathcal{Z}_i(v_i)$, the reward related to some item, and v_i is the index of that item. Property (a) says that the cost vector $\mathcal{Z}_i(v_i)$ are independent. Property (c) corresponds to how on any turn n we choose which item to present without seeing their rewards for that turn. For ease of notation, we write $Z_i(v_i)$ as Z_i in our discussions below.

THEOREM 4. For any $\varepsilon > 0$ and $\gamma_1, \gamma_2, \dots, \gamma_N \geq 0$, we have

$$P\left(\sum_{i=1}^{N} \gamma_i Z_i \geq \varepsilon\right) < \exp\left(\frac{-\varepsilon^2}{2\sum_{i=1}^{N} \gamma_i^2 \sigma_i^2}\right)$$
$$P\left(\sum_{i=1}^{N} \gamma_i Z_i \leq -\varepsilon\right) < \exp\left(\frac{-\varepsilon^2}{2\sum_{i=1}^{N} \gamma_i^2 \sigma_i^2}\right)$$

□

PROOF. Take exponentials and use the Markov inequality to get

$$P\left(\sum_{i=1}^{N} \gamma_i Z_i \geq \varepsilon\right) = P\left(\exp\left(\lambda \sum_{i=1}^{N} \gamma_i Z_i\right) \geq e^{\lambda\varepsilon}\right) \leq \frac{\mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^{N} \gamma_i Z_i\right)\right]}{e^{\lambda\varepsilon}} \tag{7.2}$$

Now take expectations to put the numerator of (7.2) in the form

$$\mathbb{E}\left[\exp\left(\lambda\gamma_N Z_N\right) \prod_{i=1}^{N-1} \exp\left(\lambda\gamma_i Z_i\right)\right] \tag{7.3}$$

To distribute the expectation over $\exp\left(\lambda\gamma_N Z_N\right)$ we must show it is independent of the other factors. Assumption 5(a) says each Z_i is independent of Z_N and so $\exp\left(\lambda\gamma_N Z_N\right)$ is independent of $\exp\left(\lambda\gamma_i Z_i\right)$. We conclude (7.3) equals

$$\mathbb{E}\left[\exp\left(\lambda\gamma_N Z_N\right)\right]\mathbb{E}\left[\prod_{i=1}^{N-1}\exp\left(\lambda\gamma_i Z_i\right)\right] \tag{7.4}$$

Assumption 5(b) says Z_N is σ_N^2-subgaussian. Hence $\mathbb{E}\left[\exp\left(\lambda\gamma_N Z_N\right)\right] \leq \exp\left(\frac{\lambda^2}{2}\gamma_N^2\sigma_N^2\right)$. Repeating the argument $N-1$ more times and applying the definition 2, equation (7.4) is at most

$$\leq \prod_{i=1}^{N}\exp\left(\frac{\lambda^2}{2}\gamma_i^2\sigma_i^2\right) = \exp\left(\frac{\lambda^2}{2}\sum_{i=1}^{N}\gamma_i^2\sigma_i^2\right)$$

Going back to equation (7.2), we get

$$P\left(\sum_{i=1}^{N}\gamma_i Z_i \geq \varepsilon\right) \leq \frac{\exp\left(\frac{\lambda^2}{2}\sum_{i=1}^{N}\gamma_i^2\sigma_i^2\right)}{e^{\lambda\varepsilon}} = \exp\left(\lambda\left(\frac{\lambda}{2}\sum_{i=1}^{N}\gamma_i^2\sigma_i^2 - \varepsilon\right)\right) = \exp\left(\lambda(a\lambda-\varepsilon)\right)$$

where $a = \frac{1}{2}\sum_{i=1}^{N}\gamma_i^2\sigma_i^2$.

Optimising the bound over λ, we get $\lambda = \varepsilon/2a$ and get the exponent as $\lambda(a\lambda-\varepsilon) = -\varepsilon^2/4a$. Hence, we get

$$P\left(\sum_{i=1}^{N}\gamma_i Z_i) \geq \varepsilon\right) \leq \exp\left(-\varepsilon^2/4a\right) = \exp\left(\frac{-\varepsilon^2}{2\sum_{i=1}^{N}\gamma_i^2\sigma_i^2}\right).$$

The proof for the second inequality is similar. □

To apply Theorem 4 to Algorithm we must define the objects in Assumption 5.

Definition 3. *We assume $g = 1$ is the correct cluster. We write $X_i(1, v_i)$ as the cost associated with rating the item v_i at turn i. For ease of notation, we write $X_i(1, v_i)$ as X_i.*

1. *$X_i(1, v_i)$ is iid with mean $\mu(1, v_i)$ and variance $\sigma(1, v_i)^2$*

2. *Set $Z_i(v_i) = X_i(1, v_i) - \mu(1, v_i)$*

3. Set $\sigma_i^2 = \sigma(1, v_i)^2$.

4. Define $\Gamma_{g,h\neq g}(v_i) = \frac{(\mu(g,v_i)-\mu(h,v_i))^2}{\max\{\sigma^2(g,v_i),\sigma^2(h,v_i)\}}$

□

Under Definition 3, $R_n(1,h), R_n(g,1)$ from equation (3.1) are

$$R_n(g,1) = \sum_{i=1}^{n-1} \frac{\alpha_i^n(1,h)}{\mu(g,v_i) - \mu(1,v_i)} Z_i$$
$$R_n(1,h) = \sum_{i=1}^{n-1} \alpha_i^n(1,h) \left(1 + \frac{Z_i}{\mu(1,v_i) - \mu(h,v_i)}\right)$$

Claim 1. *The provided definition of* Z_i *satisfies conditions (a-c) outlined in Assumption 5.*

Proof. Condition (a) follows from the independence assumption on the cost vectors $X_1, X_2, \cdots, X_N$. To prove (b), we have $Z_i = X_i(1,v_i) - \mu(1,v_i)$ which is $\sigma_i^2 = \sigma(1,v_i)^2$ -subgaussian with expectation zero by definition. To prove (c), note the decision to pull v_n on turn n depends on $X_1, \cdots, X_{n-1}$. Hence the event that we pull v_n on turn n is independent of $X_n, X_{n+1}, \cdots$ and in particular of $X_{\widetilde{n}}$. □

Lemma 5. For any $t > 0$, $g \neq 1$ and $n \in \mathbb{N}$, the weights $\alpha_i^n(g,1) = \frac{\Gamma_{g,1}(v_i)}{\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)}$ give the probability bounds

$$P(R_n(g,1) \geq t) \leq \exp\left(-\frac{t^2}{2}\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)\right)$$
$$P(R_n(g,1) \leq -t) \leq \exp\left(-\frac{t^2}{2}\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)\right)$$

□

Proof. By definition, we have $R_n(g,1) = \sum_{i=1}^{n-1} \frac{\alpha_i^n(g,1)}{\mu(g,v_i)-\mu(1,v_i)} Z_i = \sum_{i=1}^{n-1} \gamma_i Z_i$ for $\gamma_i = \frac{\alpha_i^n(g,1)}{\mu(g,v_i)-\mu(1,v_i)}$. Hence Theorem 4 says the event $\sum_{i=1}^{n-1}\gamma_i Z_i \geq t$ occurs with probability at most $\exp\left(\frac{-t^2}{2\sum_{i=1}^{n-1}\gamma_i^2\sigma_i^2}\right)$. The denominator simplifies to

$$\sum_{i=1}^{n-1}\frac{\alpha_i^n(g,1)^2\sigma(1,v_i)^2}{\left(\mu(g,v_i)-\mu(1,v_i)\right)^2} \leq \sum_{i=1}^{n-1}\frac{\alpha_i^n(g,1)^2\max\{\sigma(1,v_i)^2,\sigma(g,v_i)^2\}}{\left(\mu(g,v_i)-\mu(1,v_i)\right)^2}$$
$$= \sum_{i=1}^{n-1}\frac{\alpha_i^n(g,1)^2}{\Gamma_{g,1}(v_i)} = \sum_{i=1}^{n-1}\left(\frac{\Gamma_{g,1}(v_i)}{\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)}\right)^2\frac{1}{\Gamma_{g,1}(v_i)}$$
$$= \frac{1}{\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)}$$

So we can write the bound as

$$P(R_n(g,1)\geq t)\leq \exp\left(-\frac{t^2}{2}\sum_{j=1}^{n-1}\Gamma_{g,1}(v_j)\right)$$

This completes the proof. The proof of the second inequality is similar.

□

LEMMA 6. *For $t>0$, $g=1$ and $n\in\mathbb{N}$, the weights $\alpha_i^n(1,h)=\frac{\Gamma_{1,h}(v_i)}{\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)}$ give the probability bounds*

$$P(R_n(1,h)-1\geq t)\leq \exp\left(-\frac{t^2}{2}\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)\right)$$
$$P(R_n(1,h)-1\leq -t)\leq \exp\left(-\frac{t^2}{2}\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)\right)$$

□

PROOF. By defintion, we have $R_n(1,h)=\sum_{i=1}^{n-1}\alpha_i^n(1,h)\left(1+\frac{Z_i}{\mu(g,v_i)-\mu(1,v_i)}\right)=1+\sum_{i=1}^{n-1}\gamma_i Z_i$ for $\gamma_i=\frac{\alpha_i^n(g,1)}{\mu(1,v_i)-\mu(h,v_i)}$. i.e we have $R_n(1,h)-1=\sum_{i=1}^{n-1}\gamma_i Z_i$. Hence Theorem 4 says $R_n(1,h)-1=\sum_{i=1}^{n-1}\gamma_i Z_i\geq t$ occurs with probability at most $\exp\left(\frac{-t^2}{2\sum_{i=1}^{n-1}\gamma_i^2\sigma_i^2}\right)$. The denominator simplifies to

$$\sum_{i=1}^{n-1} \frac{\alpha_i^n(1,h)^2\sigma(1,v_i)^2}{\left(\mu(1,v_i)-\mu(h,v_i)\right)^2} \leq \sum_{i=1}^{n-1} \frac{\alpha_i^n(1,h)^2 \max\{\sigma(1,v_i)^2, \sigma(h,v_i)^2\}}{\left(\mu(1,v_i)-\mu(h,v_i)\right)^2}$$
$$= \sum_{i=1}^{n-1} \frac{\alpha_i^n(1,h)^2}{\Gamma_{1,h}(v_i)} = \sum_{i=1}^{n-1} \left(\frac{\Gamma_{1,h}(v_i)}{\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)} \right)^2 \frac{1}{\Gamma_{1,h}(v_i)}$$
$$= \sum_{i=1}^{n-1} \frac{\Gamma_{1,h}(v_i)}{\left(\sum_{j=1}^{n}\Gamma_{1,h}(v_j)\right)^2} = \frac{1}{\sum_{j=1}^{n-1}\Gamma_{1,h}(v_j)}$$

So we can write the bound as

$$P(R_n(1,h)-1 \geq t) \leq \exp\left(-\frac{t^2}{2} \sum_{j=1}^{n-1} \Gamma_{1,h}(v_j) \right)$$

This completes the proof. The proof of the second inequality is similar. □

7.2 Appendix B

7.2.1 Properties Of Positive Definite/ Semi-Definite Matrices

Definition 4. *A matrix M is called Positive Semi-Definite (PSD) if it is symmetric and its eigen values are non-negative. If the eigen values are positive, they are called Positive Definite (PD) Matrices. Equivalently, we can say that if the matrix M is symmetric and satisfies $x^T Mx >= 0$ for a non-zero vector x, then M is PSD and if M is symmetric and satisfies $x^T Mx > 0$ for a non-zero vector x, then M is PD.* □

Lemma 7. *Any $n \times n$ matrix of the form $M = BB^T$ is PSD for any $n \times m$ matrix B.* □

Lemma 8. *Let A be a PSD matrix, and B be a PD matrix. Then their sum $A + B$ is a PD Matrix.* □

Lemma 9. *Let M be a PSD matrix, then its inverse M^{-1} is also a PSD.* □

Lemma 10. *If the matrix M is PD, then all diagonal elements of M are positive.* □

7.2.2 Sherman-Morrison Formula

Given M is a square $n \times n$ matrix whose inverse matrix we know, for the simple case of a rank-1 perturbation to M, Sherman–Morrison formula provides a method to find the updated rank-1 change to the inverse.

DEFINITION 5. *Given M is a square $n \times n$ matrix whose inverse matrix we know, u and v are $n \times 1$ column vectors defining the perturbation to matrix M such that $u_i v_j^T$ is added to $M_{i,j}$, then we can find the inverse of the modified M by Sherman-Morrison formula*

$$(M + uv^T)^{-1} = M^{-1} - \frac{M^{-1}uv^T M^{-1}}{1 + v^T M^{-1} u}$$

□

For a generalised rank-k perturbation to M, the updated inverse is given by the following formula.

DEFINITION 6. *Given M is a square $n \times n$ matrix whose inverse matrix we know, U and V are $n \times k$ matrices defining rank k perturbation to matrix M, then we can find the inverse of the modified M*

$$(M + UV^T)^{-1} = M^{-1} - M^{-1}U(I_k + V^T M^{-1} U)^{-1} V^T M^{-1}$$

□

LEMMA 11. *if M^{-1} is a PSD matrix and $\hat{M} = M + uu^T$ for non-zero $u \in \mathbb{R}^{n \times 1}$, then diagonal elements of $\hat{M}^{-1}$ is less than or equal to the diagonal elements of matrix M^{-1}.* □

7.2.3 Theorems and Proofs

Deriving Recursive Updates to V_j

Proof of Theorem 2

PROOF. For simplicity, let Λ gather together the feature vectors U_i of all users i who rated target item j^* such that $\Lambda\Lambda^T = \sum_{i \in \mathcal{U}(j^*)} U_i U_i^T$ and let r be a column vector of ratings received by the target item from these true users. We can write equation 4.3 as,

$$V_{j^*} = (\Lambda\Lambda^T + \lambda I)^{-1} \Lambda r = A^{-1} \Lambda r$$

where $A = \Lambda\Lambda^T + \lambda I$. After the block of fake users enter, we write $\hat{U} = [\Lambda, X]$ and $\hat{U}\hat{U}^T = \Lambda\Lambda^T + XX^T$. Also $\hat{r} = \begin{bmatrix} r \\ y \end{bmatrix}$ and $\hat{A}^{-1} = (\hat{U}\hat{U}^T + \lambda I)^{-1} = (A + XX^T)^{-1}$. We now calculate the updated $\hat{V_{j^*}}$. We know that,

$\hat{V_{j^*}} = \hat{A}^{-1}\hat{U}\hat{r} = \hat{A}^{-1}(\Lambda r + Xy) = \left(A + XX^T\right)^{-1}(\Lambda r + Xy)$

Applying Sherman Morrison/Woodbury formula to the inverse (see Definition 5)

$= \left(A^{-1} - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TA^{-1}\right)(\Lambda r + Xy)$

$= A^{-1}\Lambda r - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TA^{-1}\Lambda r + A^{-1}Xy - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TA^{-1}Xy$

We have, $V_{j^*} = A^{-1}\Lambda r$. Substituting for $A^{-1}\Lambda r$ and rearranging, we get,

$= V_{j^*} - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TV_{j^*} + A^{-1}X\left(I - \left(I + X^TA^{-1}X\right)^{-1}X^TA^{-1}X\right)y$

Substituting $I = (I + X^TA^{-1}X)^{-1}(I + X^TA^{-1}X)$ inside third term, we get,

$= V_{j^*} - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TV_{j^*} +$

$A^{-1}X\left(\left(I + X^TA^{-1}X\right)^{-1}\left(I + X^TA^{-1}X\right) - \left(I + X^TA^{-1}X\right)^{-1}X^TA^{-1}X\right)y$

$= V_{j^*} - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TV_{j^*} + A^{-1}X\left(\left(I + X^TA^{-1}X\right)^{-1}\left(I + X^TA^{-1}X - X^TA^{-1}X\right)\right)y$

$= V_{j^*} - A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}X^TV_{j^*} + A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}y$

$= V_{j^*} + A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}\left(y - X^TV_{j^*}\right)$

□

Finally, we have,

$$\hat{V_{j^*}} - V_{j^*} = A^{-1}X\left(I + X^TA^{-1}X\right)^{-1}\left(y - X^TV_{j^*}\right) \tag{7.5}$$

Here we have three terms $a = A^{-1}X$, $b = \left(I + X^TA^{-1}X\right)^{-1}$ and $c = \left(y - X^TV_{j^*}\right)$. Using Lemma 6, 7, 8, 9, 10 (next section) we re-write equation 7.5 as

$$\hat{V_{j*}} - V_{j*} = m \times K \times \left(A^{-1}\tilde{U}_t\right)$$

where $K = bc$ and m, K are positive constants.

Re-writing Equation 7.5

If we look at the term $A^{-1} = (\Lambda\Lambda^T + \lambda I)^{-1}$ in equation 7.5, since $\Lambda\Lambda^T$ is a multiplication of a matrix with its transpose, it is positive semi-definite (PSD) matrix, and λI is a positive

definite (PD) matrix. [1] Then $(\Lambda\Lambda^T + \lambda I)$ is a PD matrix [2]. Since the inverse of a PD matrix is also PD, A^{-1} is a PD matrix. [3] Because A^{-1} is a PD matrix and since columns of X are identical to $\tilde{U}_t$, by lemma 12 we can say that term $X^T A^{-1} X$ in b results in an $\mathbb{R}^{n \times n}$ matrix with identical values for all the elements.

LEMMA 12. *Let $M = X^T A^{-1} X$ and $x_i, x_j \in \mathbb{R}^{d \times 1}$ be the i^{th} and j^{th} column in X. Given $x_i = x_j = \tilde{U}_t$ and A^{-1} is a PD matrix, then $(m)_{ij} = x_i^T A^{-1} x_j$ is a positive constant.* □

PROOF.

$$(m)_{ij} = x_i^T A^{-1} x_j = \tilde{U}_t^T A^{-1} \tilde{U}_t$$

We know by Definition 4 that pre-multiplying and post-multiplying a PD matrix by the same vector takes a quadratic form and always results in a positive number provided the multiplying vector is non-zero. □

So $X^T A^{-1} X$ results in a matrix with all elements having identical values. By lemma 13 the inverse of $I + X^T A^{-1} X$ results in a matrix with all diagonal elements having identical values and the non-diagonal elements having identical values.

LEMMA 13. *For any matrix $M \in \mathbb{R}^{n \times n}$ with identical elements, the inverse of $I + M$ results in a matrix with all diagonal elements having identical values and the non-diagonal elements having identical values.* □

PROOF. For any $M \in \mathbb{R}^{n \times n}$ with identical elements, we can write $(I + M)^{-1} = \left(I + ec^T\right)$, for $M = ec^T$ where $e = [1, 1, 1, \cdots, 1]^T$ and c is a column of M. Applying Sherman-Morrison formula, we get

$$(I + M)^{-1} = I - \frac{M}{1 + c^T e}$$

Note that the denominator is a positive constant. So we can say that the resulting matrix will have all diagonal elements the same and all non-diagonal values the same. □

Now we have term b. Next, we proceed to evaluate the term c. Term $c = y - X^T V_{j^*}$ says how much of target item fake rating vector y with all its value equal to the highest rating 5 can be explained by the existing weight vector V_{j^*}. If V_{j^*} before the attack can explain the vector y then term c will be a zero vector and equation 7.5 results in $\hat{V_{j^*}} = V_{j^*}$

[1] See Definition 4 and Lemma 7 in Appendix 7.2.1
[2] See Lemma 8 in Appendix 7.2.1
[3] See Lemma 9 in Appendix 7.2.1

as is to be expected i.e. there is no point in attacking an item that already has the maximum rating in the target cluster. By lemma 14, term c results in a column vector containing identical values in each row. In particular, we have,

LEMMA 14. *Given a column vector $y \in \mathbb{R}^{m\times 1}$ with constant values, all rows of column vector c result in an identical positive constant.* □

PROOF. We know that any i^{th} column in X can be written as $x_i = \tilde{U}_t$. Then we can write for any i^{th} row of term $c = y - X^T V_{j^*}$

$$y_i - x_i^T V_{j^*} = y_i - \tilde{U}_t^T V_{j^*}$$

$\tilde{U}_t^T V_{j^*}$ is the predicted rating of item j^* in cluster t before attack and is a constant for all users belonging to cluster t. Also $y_i = 5$ in our set-up, thus rows of $y - X^T V_{j^*}$ will result in the same non-zero positive value. □

Now that we have terms b, c, we proceed to find the values of the column vector resulting from the product of terms b and c.

LEMMA 15. *Given $m \times m$ matrix $b = (I + X^T A^{-1} X)^{-1}$ and $m \times 1$ matrix $c = y - X^T V_{j^*}$, the $m \times 1$ column vector resulting from their product has same value for all rows.* □

PROOF. We can write bc as

$$bc = b(y - X^T V_{j^*}) = by - b(X^T V_{j^*}) = by - b\left([\tilde{U}_t, \tilde{U}_t, \cdots, \tilde{U}_t]^T V_{j^*}\right)$$

We have $[\tilde{U}_t, \tilde{U}_t, \cdots, \tilde{U}_t]^T V_{j^*} = [\tilde{U}_t^T V_{j^*}, \tilde{U}_t^T V_{j^*}, \cdots, \tilde{U}_t^T V_{j^*}]^T$ So we can write

$$bc = by - b[\tilde{U}_t^T V_{j^*}, \tilde{U}_t^T V_{j^*}, \cdots, \tilde{U}_t^T V_{j^*}]^T$$

Using lemma 12, b is a matrix with all diagonal values identical and all non-diagonal values identical and we know $\tilde{U}_t^T V_{j^*}$ is a constant.

$bc = [K, K, \cdots, K]^T$ for positive constant K

□

Now we have $\hat{V}_{j^*} - V_{j^*} = A^{-1} X(bc)$ where $bc = [K, K, \cdots, K]^T$ for positive constant K. From lemma 16, computing the product $X(bc)$ results in a $\mathbb{R}^{d\times 1}$ vector with values as $m \times K \times \tilde{U}_t$.

LEMMA 16. *Given $X \in \mathbb{R}^{d\times m}$ with i^{th} column vector x_i as $\tilde{U}_t$ and $bc = [K, K, \cdots, K]^T$, we can write $X(bc) = m \times K \times \tilde{U}_t$.* □

PROOF.

We can write $X(bc)$ as,

$$X(bc) = [x_1, x_2, \cdots, x_m][K, K, \cdots, K]^T$$

Using $x_i = \tilde{U}_t$ where $\tilde{U}_t = [u_1^t, u_2^t, \cdots, u_d^t]^T$

$$= \begin{bmatrix} u_1^t & u_1^t & \cdots, & u_1^t \\ u_2^t & u_2^t & \cdots, & u_2^t \\ \vdots & & & \\ u_d^t & u_d^t & \cdots, & u_d^t \end{bmatrix} \begin{bmatrix} K \\ K \\ \vdots \\ K \end{bmatrix} = \begin{bmatrix} u_1^t K + u_1^t K + \cdots + u_1^t K \\ u_2^t K + u_2^t K + \cdots + u_2^t K \\ \vdots \\ u_d^t K + u_d^t K + \cdots + u_d^t K \end{bmatrix}$$

$$= [mKu_1^t, mKu_2^t, \cdots, mKu_d^t]^T = m \times K \times \tilde{U}_t$$

□

Equation 7.5 can now be written as $\hat{V_{j^*}} - V_{j^*} = m \times K \times \left(A^{-1}\tilde{U}_t\right)$

Proof of Lemma 17

DEFINITION 7. *Any matrix $M \in \mathbb{R}^{d\times n}$ with identical columns will satisfy $MM^T = nmm^T$ where m is a column of M. Similarly we can write $\sum_{i\in\mathcal{U}(j)} U_i U_i^T = n\tilde{U}_g\tilde{U}_g^T$ if the set of n users in $\mathcal{U}(j)$ belong to same cluster g with weight vector $\tilde{U}_g$* □

LEMMA 17. *Given a target item j^* such that $A^{-1} = \left(\sum_{i\in\mathcal{U}(j^*)} U_i U_i^T + \lambda I\right)^{-1}$ where $\mathcal{U}(j^*)$ is the set of n users who rated item j^*. Let a block of fake users with feature vectors in $X \in \mathbb{R}^{d\times m}$ enter to increase the predicted rating of item j^* in a cluster t with $X = [\tilde{U}_t, \tilde{U}_t, \cdots, \tilde{U}_t]$ where $\tilde{U}_t = [u_1^t, u_2^t, \cdots, u_d^t]^T$, then*

1) If the n users belong to target cluster t, all terms $a_{ki}u_i^t$ of any row k of matrix $A^{-1}\tilde{U}_t$ (equation 4.5) will have the same sign and the sign will be opposite to $sgn(a_{kk}u_k^t)$.

2) If the n users are not exclusive to the target cluster, all terms $a_{ki}u_i^t$ of any row k of the matrix $A^{-1}\tilde{U}_t$ (equation 4.5) need not have the same sign and the sign does not depend exclusively on $sgn(a_{kk}u_k^t)$. □

PROOF. Let target item be rated by all users belonging to target cluster. By Definition 7,

we have,

$$A^{-1} = \left(\sum_{i \in \mathcal{U}(j^*)} U_i U_i^T + \lambda I \right)^{-1} = (\lambda I + n\tilde{U}_t \tilde{U}_t^T)^{-1} = \frac{1}{n} \left(\frac{\lambda}{n} I + \tilde{U}_t \tilde{U}_t^T \right)^{-1}$$

To find A^{-1}, we apply Sherman-Morrison formula (see Definition 5) to the inverse term

$$\begin{aligned} \left(\frac{\lambda}{n} I + \tilde{U}_t \tilde{U}_t^T \right)^{-1} &= \frac{n}{\lambda} I - \frac{\frac{n}{\lambda} I \tilde{U}_t \tilde{U}_t^T \frac{n}{\lambda} I}{1 + \tilde{U}_t^T \frac{n}{\lambda} I \tilde{U}_t} = \frac{n}{\lambda} I - \frac{\left(\frac{n}{\lambda}\right)^2 \tilde{U}_t \tilde{U}_t^T}{1 + \frac{n}{\lambda} \tilde{U}_t^T \tilde{U}_t} = \frac{n}{\lambda} I - \frac{\frac{n}{\lambda} \tilde{U}_t \tilde{U}_t^T}{\frac{\lambda}{n} + \tilde{U}_t^T \tilde{U}_t} \\ &= \frac{n}{\lambda} I - \frac{\frac{n}{\lambda} \tilde{U}_t \tilde{U}_t^T}{C} \text{ where denominator } C = \frac{\lambda}{n} + \tilde{U}_t^T \tilde{U}_t \text{ is a positive constant} \end{aligned}$$

Then we can write A^{-1} as

$$A^{-1} = \frac{1}{n} \left(\frac{\lambda}{n} I + \tilde{U}_t \tilde{U}_t^T \right)^{-1} = \frac{1}{\lambda} \left(I - \frac{\tilde{U}_t \tilde{U}_t^T}{C} \right)$$

Case 1: Now that we have matrix A^{-1} for the case when true users come exclusively from the target cluster, we calculate $A^{-1}\tilde{U}_t$,

$$A^{-1}\tilde{U}_t = \frac{1}{\lambda} \begin{bmatrix} 1 - \frac{(u_1^t)^2}{C} & -\frac{u_1^t u_2^t}{C} & \cdots & -\frac{u_1^t u_d^t}{C} \\ -\frac{u_2^t u_1^t}{C} & 1 - \frac{(u_2^t)^2}{C} & \cdots & -\frac{u_2^t u_d^t}{C} \\ \vdots & & & \\ -\frac{u_d^t u_1^t}{C} & -\frac{u_d^t u_2^t}{C} & \cdots & 1 - \frac{(u_d^t)^2}{C} \end{bmatrix} \begin{bmatrix} u_1^t \\ u_2^t \\ \vdots \\ u_d^t \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} u_1^t \left(1 - \frac{(u_1^t)^2}{C}\right) + \sum_{i=1, i \neq 1}^{d} -u_1^t \frac{(u_i^t)^2}{C} \\ u_2^t \left(1 - \frac{(u_2^t)^2}{C}\right) + \sum_{i=1, i \neq 2}^{d} -u_2^t \frac{(u_i^t)^2}{C} \\ \vdots \\ u_d^t \left(1 - \frac{(u_d^t)^2}{C}\right) + \sum_{i=1, i \neq d}^{d} -u_d^t \frac{(u_i^t)^2}{C} \end{bmatrix}$$

Recall equation 4.5 that gives the k^{th} row of $A^{-1}\tilde{U}_t$. Here the k^{th} row $a_{kk}u_k^t + \sum_{i=1, i \neq k}^{d} a_{ki}u_i^t = u_k^t \left(1 - \frac{(u_k^t)^2}{C}\right) + \sum_{i=1, i \neq k}^{d} -u_k^t \frac{(u_k^t)^2}{C}$. Note that the first term $u_k^t \left(1 - \frac{(u_k^t)^2}{C}\right)$ has the sign of u_k^t since $\frac{(u_k^t)^2}{C} < 1$. Consider the terms in the summation i.e. $a_{ki}u_i^t = -u_k^t \frac{(u_i^t)^2}{C}$. Here $\frac{(u_i^t)^2}{C}$ is positive because the numerator is a square and the denominator is a positive constant. So each term $-u_k^t \frac{(u_i^t)^2}{C}$ has sign $-sgn(u_k^t)$ or in other words opposite to the sign of the first term. Therefore, for case 1, all terms $a_{ki}u_i^t$ of any row k of the matrix $A^{-1}\tilde{U}_t$ (equation 4.5) has the same sign and the sign is opposite to $sgn(a_{kk}u_k^t)$.

Case 2: Following similar steps, $A^{-1}\tilde{U}_t$ when true ratings come exclusively from any

cluster $g \neq t$ and for $C = \frac{\lambda}{n} + \tilde{U}_g^T \tilde{U}_g$ is given by.

$$\hat{A}^{-1}\tilde{U}_t = \frac{1}{\lambda}\begin{bmatrix} 1-\frac{(u_1^g)^2}{C} & -\frac{u_1^g u_2^g}{C} & \cdots & -\frac{u_1^g u_d^g}{C} \\ -\frac{u_2^g u_1^g}{C} & 1-\frac{(u_2^g)^2}{C} & \cdots & -\frac{u_2^g u_d^g}{C} \\ \vdots & & & \\ -\frac{u_d^g u_1^g}{C} & -\frac{u_d^g u_2^g}{C} & \cdots & 1-\frac{(u_d^g)^2}{C} \end{bmatrix}\begin{bmatrix} u_1^t \\ u_2^t \\ \vdots \\ u_d^t \end{bmatrix} = \frac{1}{\lambda}\begin{bmatrix} u_1^t\left(1-\frac{(u_1^g)^2}{C}\right) + \sum_{i=1,i\neq 1}^{d} -u_i^t\frac{u_1^g u_i^g}{C} \\ u_2^t\left(1-\frac{(u_2^g)^2}{C}\right) + \sum_{i=1,i\neq 2}^{d} -u_i^t\frac{u_2^g u_i^g}{C} \\ \vdots \\ u_d^t\left(1-\frac{(u_d^g)^2}{C}\right) + \sum_{i=1,i\neq d}^{d} -u_i^t\frac{u_2^g u_i^g}{C} \end{bmatrix}$$

Similar to previous case, first term in k^{th} row of $A^{-1}\tilde{U}_t$ has the sign of u_k^t. The terms in summation $a_{ki}u_i^g = -u_i^t\frac{u_k^g u_i^g}{C}$, has its sign depend on $sgn(u_i^t u_k^g u_i^g)$. So unlike case 1, term $a_{ki}u_i^g$ may have any sign.

When true ratings come from a combination of clusters:

Consider a simple extension where n_1 users of target cluster t and n_2 users of cluster g provide ratings. Given $n_1 + n_2 = n$, we have,

$$\begin{aligned} A^{-1} &= (\lambda I + n_1\tilde{U}_t\tilde{U}_t^T + n_2\tilde{U}_g\tilde{U}_g^T)^{-1} = (M + n_2\tilde{U}_g\tilde{U}_g^T)^{-1} \\ &= M^{-1} - \frac{M^{-1}\tilde{U}_g\tilde{U}_g^T M^{-1}}{\frac{1}{n_2} + \tilde{U}_g^T M^{-1}\tilde{U}_g} \text{ where } M^{-1} = \left(\lambda I + n_1\tilde{U}_t\tilde{U}_t^T\right)^{-1} \text{ (given in case 1)} \end{aligned}$$

Caclulating $A^{-1}\tilde{U}_t$

$$A^{-1}\tilde{U}_t = M^{-1}\tilde{U}_t - \frac{M^{-1}\tilde{U}_g\tilde{U}_g^T M^{-1}\tilde{U}_t}{\frac{1}{n_2} + \tilde{U}_g^T M^{-1}\tilde{U}_g} = M^{-1}(\tilde{U}_t - \beta\tilde{U}_g) \text{ where constant } \beta = \frac{\tilde{U}_g^T M^{-1}\tilde{U}_t}{\frac{1}{n_2} + \tilde{U}_g^T M^{-1}\tilde{U}_g}$$

$$= \frac{1}{\lambda}\begin{bmatrix} (u_1^t - \beta u_1^g)\left(1-\frac{(u_1^t)^2}{C}\right) + \sum_{i=1,i\neq 1}^{d} -\frac{-u_1^t u_i^t}{C}(u_i^t - \beta u_i^g) \\ (u_2^t - \beta u_2^g)\left(1-\frac{(u_2^t)^2}{C}\right) + \sum_{i=1,i\neq 2}^{d} -\frac{-u_2^t u_i^t}{C}(u_i^t - \beta u_i^g) \\ \vdots \\ (u_d^t - \beta u_d^g)\left(1-\frac{(u_d^t)^2}{C}\right) + \sum_{i=1,i\neq d}^{d} -\frac{-u_d^t u_i^t}{C}(u_i^t - \beta u_i^g) \end{bmatrix}$$

For any row k in the resultant matrix, while the first term will have $sgn(u_k^t - \beta u_k^g)$, each term in the summation given by $\frac{-u_k^t u_i^t}{C}(u_i^t - \beta u_i^g)$ has its sign depending on many factors. Similarly, we can extend the analysis to true ratings coming from any combination of clusters by finding the inverse recursively per cluster rating block using the Sherman-Morrison update.

Thus with the exception of case 1, for any combination of cluster-wise distribution of true ratings, all terms $a_{ki}u_i^t$ of any row k of the matrix $A^{-1}\tilde{U}_t$ (equation 4.5) need not

have the same sign and the sign does not depend exclusively on $sgn(a_{kk}u_k^t)$. □

Proof of Theorem 3

PROOF. Given A^{-1}, let N users from target cluster provide ratings. Then for $\epsilon > 0$ and $N \geq \frac{1}{\epsilon}$, we update the inverse by Sherman-Morrison formula (Definition 5)

$$\hat{A}^{-1} = (A + N\tilde{U}_t\tilde{U}_t^T)^{-1} = A^{-1} - \frac{A^{-1}\tilde{U}_t\tilde{U}_t^T A^{-1}}{\frac{1}{N} + \tilde{U}_t^T A^{-1}\tilde{U}_t} = A^{-1} - \frac{NA^{-1}\tilde{U}_t\tilde{U}_t^T A^{-1}}{1 + N\tilde{U}_t^T A^{-1}\tilde{U}_t}$$

Now that we have $\hat{A}^{-1}$, we calculate $\hat{A}^{-1}\tilde{U}_t$ of equation 4.4.

$$\hat{A}^{-1}\tilde{U}_t = \left(A^{-1} - \frac{NA^{-1}\tilde{U}_t\tilde{U}_t^T A^{-1}}{1 + N\tilde{U}_t^T A^{-1}\tilde{U}_t}\right)\tilde{U}_t = \left(\frac{A^{-1}\tilde{U}_t}{1 + N\tilde{U}_t^T A^{-1}\tilde{U}_t}\right)$$

We see that as $\epsilon \to 0$, $N \to \infty$, causing $\hat{A}^{-1}\tilde{U}_t$ to converge to zero vector.

Now suppose, instead of the target cluster, users from any non-target cluster g provide the N ratings; then we have

$$\begin{aligned}\hat{A}^{-1}\tilde{U}_t &\leq \left(A^{-1} - \frac{A^{-1}\tilde{U}_g\tilde{U}_g^T A^{-1}}{\epsilon + \tilde{U}_g^T A^{-1}\tilde{U}_g}\right)\tilde{U}_t = \left(A^{-1}\tilde{U}_t - \frac{A^{-1}\tilde{U}_g(\tilde{U}_g^T A^{-1}\tilde{U}_t)}{\epsilon + \tilde{U}_g^T A^{-1}\tilde{U}_g}\right) \\ &= \left(\epsilon A^{-1}\tilde{U}_t + C_1 A^{-1}\tilde{U}_t - C_2 A^{-1}\tilde{U}_g\right)\end{aligned}$$

where constants $C_1 = \frac{\tilde{U}_g^T A^{-1}\tilde{U}_g}{\epsilon + \tilde{U}_g^T A^{-1}\tilde{U}_g}$ and $C_2 = \frac{\tilde{U}_g^T A^{-1}\tilde{U}_t}{\epsilon + \tilde{U}_g^T A^{-1}\tilde{U}_g}$. As C_1, C_2 are non-zero constants with $sgn(C_1)$ as positive and $sgn(C_2)$ depending on $sgn(\tilde{U}_g^T A^{-1}\tilde{U}_t)$, $\hat{A}^{-1}\tilde{U}_t$ might not converge to a zero vector. □

7.2.4 Additional Experimental Results

We look at a slightly modified hit-rate definition, defining an item to belong to the recommended list if it has a predicted rating >= 4. If the target item after the attack satisfies this condition, we consider it a 'Hit.' i.e, it may be more likely for the target item to be now recommended to users of the target cluster. We measure the Average Hits after the attack for the target item and show the results when the adversary targets cluster 2. Results when targeting other clusters are similar and so are not reported separately.

Fix V, Update U

From table 7.1, Average-Hits over all the target items calculated is 0 for both datasets for increasing ratios of $\frac{n}{m}$, indicating that the attack is not able to push the rating high enough to be included in the recommended list of items. i.e, the results of the attack here, do not increase the predicted rating to ≥ 4. This supports our results in the main paper (figure 4.2), where the maximum mean relative change in rating observed for $\frac{n}{m} = 0.5$ was < 0.30 i.e. for both datasets, predicted rating increases only up to 30% of maximum deviation possible.

$\frac{n}{m}$/Avg Hits	Before	After
0.5	0	0
1.2	0	0
2.0	0	0

(a) Movielens

$\frac{n}{m}$/Avg Hits	Before	After
0.5	0	0
1.2	0	0
2.0	0	0

(b) Goodreads

Table 7.1: Average Hits for Movielens dataset and Goodreads dataset reported when target cluster is $t = 2$ for the cluster-wise distribution of true users $250 - 250 - n - 250$

Fix U, Update V

Varying Ratio of Target Item True Ratings (N_t) to Fake Ratings (N_f) in Target Cluster

Table 7.2 reports the average hits for the target item for increasing N_t from the target cluster given N_f fake ratings. We can see how the average hit is highest for $\frac{N_t}{N_f} = 0.05, 0.5, 1$ and drops to zero afterward. This is in agreement with the observations from Figure 4.5, where $\frac{N_t}{N_f} = 0.05, 0.5, 1$ showed a significant mean shift in the predicted rating $>= 0.5$ after the attack compared to the mean shift in the predicted rating < 0.4 for $\frac{N_t}{N_f} > 1$. Therefore, the attack increases the predicted rating to ≥ 4, raising the chances of being recommended to users of the target cluster, for decreasing $\frac{N_t}{N_f}$.

$\frac{N_t}{N_f}$/Avg Hits	Before	After
0.05	0	1
0.5	0	1
1	0	1
1.75	0	0
2.5	0	0

(a) Movielens

$\frac{N_t}{N_f}$/Avg Hits	Before	After
0.05	0	1
0.5	0	1
1	0	1
1.75	0	0
2.5	0	0

(b) Goodreads

Table 7.2: Average Hits for Movielens dataset and Goodreads dataset reported when target cluster is $t = 2$ for the rating distribution of type $0 - 0 - N_t - 0$.

Varying Ratio of Target Item True Ratings (N_t) to Fake Ratings (N_f) in Non-Target Clusters

Table 7.3 reports the average hits for target item for each dataset. We can see how the average hit remains high for increasing $\frac{N_t}{N_f}$. This agrees with the observations from Figure 4.7, where increasing $\frac{N_t}{N_f}$ showed larger mean shift in predicted rating of > 0.95 after the attack increasing the predicted rating to ≥ 4, thus raising the chances of being recommended to users of target cluster.

$\frac{N_t}{N_f}$/Avg Hits	Before	After
0.05	0	1
0.5	0	1
1	0	1
1.75	0	1
2.5	0	1

(a) Movielens

$\frac{N_t}{N_f}$/Avg Hits	Before	After
0.05	0	1
0.5	0	1
1	0	1
1.75	0	1
2.5	0	1

(b) Goodreads

Table 7.3: Average Hits for Movielens dataset and Goodreads dataset reported when target cluster is $t = 2$ for the rating distribution of type $N_t - N_t - 0 - N_t$.

Bibliography

[Amatriain et al., 2009] Amatriain, X., Lathia, N., Pujol, J., Kwak, H., and Oliver, N. (2009). The wisdom of the few a collaborative filtering approach based on expert opinions from the web. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 532–539.

[Angwin and Parris Jr., 2016] Angwin, J. and Parris Jr., T. (2016). Facebook lets advertisers exclude users by race. In *ProPublica*.

[Barjasteh et al., 2016] Barjasteh, I., Forsati, R., Ross, D., Esfahanian, A.-H., and Radha, H. (2016). Cold-start recommendation with provable guarantees: A decoupled approach. In *IEEE Transactions on Knowledge and Data Engineering*, volume 28, pages 1–1.

[Bates, 2013] Bates, D. (2013). Samsung ordered to pay 340,000 dollars after it paid people to write negative online reviews about htc phones. In *Daily Mail*.

[Bhaumik et al., 2006] Bhaumik, R., Williams, C., Mobasher, B., and Burke, R. (2006). Securing collaborative filtering against malicious attacks through anomaly detection. In *AAAI Conference on Artificial Intelligence*.

[Bindra, 2021] Bindra, C. (2021). Building a privacy-first future for web advertising. In *Google Ads*.

[Bubeck et al., 2013] Bubeck, S., Perchet, V., and Rigollet, P. (2013). Bounded regret in stochastic multi-armed bandits. In *Journal of Machine Learning Research*, volume 30.

[Burke et al., 2006] Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. (2006). Detecting profile injection attacks in collaborative recommender systems. In *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, volume 2006, pages 23–23.

[Burke et al., 2005] Burke, R., Mobasher, B., Zabicki, R., and Bhaumik, R. (2005). Identifying attack models for secure recommendation. In *Beyond Personalization*, volume 2005.

[Canamares et al., 2019] Canamares, R., Redondo, M., and Castells, P. (2019). Multi-armed recommender system bandit ensembles. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19.

[Chang et al., 2015] Chang, S., Harper, F. M., and Terveen, L. (2015). Using groups of items for preference elicitation in recommender systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW, New York, NY, USA. Association for Computing Machinery.

[Checco et al., 2017] Checco, A., Bianchi, G., and Leith, D. J. (2017). Blc: Private matrix factorization recommenders via automatic group learning. In *ACM Trans. Priv. Secur.*, New York, NY, USA. Association for Computing Machinery.

[Chen et al., 2018] Chen, R., Hua, Q., Chang, Y., Wang, B., Zhang, L., and Kong, X. (2018). A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. In *IEEE Access*, volume 6, pages 64301–64320.

[Cheng and Hurley, 2009] Cheng, Z. and Hurley, N. (2009). Effective diverse and obfuscated attacks on model-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 141–148.

[Chirita et al., 2005] Chirita, P.-A., Nejdl, W., and Zamfir, C. (2005). Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, WIDM '05, page 67–74, New York, NY, USA. Association for Computing Machinery.

[Choi, 2014] Choi, Y. S. (2014). Content type based adaptation in collaborative recommendation. In *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, New York, NY, USA. Association for Computing Machinery.

[Christakopoulou and Banerjee, 2019] Christakopoulou, K. and Banerjee, A. (2019). Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Con-*

ference on Recommender Systems, RecSys '19, page 322–330, New York, NY, USA. Association for Computing Machinery.

[Croux and Filzmoser, 1998] Croux, C. and Filzmoser, P. (1998). Robust factorization of a data matrix. In Payne, R. and Green, P., editors, *COMPSTAT*, pages 245–250, Heidelberg. Physica-Verlag HD.

[Datta et al., 2014] Datta, A., Tschantz, M. C., and Datta, A. (2014). Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. In *arXiv preprint arXiv:1408.6491*.

[Deldjoo et al., 2020] Deldjoo, Y., Di Noia, T., and Merra, F. A. (2020). *Adversarial Machine Learning in Recommender Systems (AML-RecSys)*, page 869–872. Association for Computing Machinery, New York, NY, USA.

[Deldjoo et al., 2021] Deldjoo, Y., Noia, T. D., and Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. In *ACM Computing Surveys (CSUR)*, pages 1–38. ACM New York, NY, USA.

[Dwoskin and Timberg, 2018] Dwoskin, E. and Timberg, C. (2018). How merchants use facebook to flood amazon with fake reviews. In *The Washington Post*.

[Elahi et al., 2018] Elahi, M., Braunhofer, M., Gurbanov, T., and Ricci, F. (2018). User preference elicitation, rating sparsity and cold start. In *Collaborative Recommendations*, pages 253–294.

[Elahi et al., 2016] Elahi, M., Ricci, F., and Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. In *Computer Science Review*, volume 20, pages 29–50.

[Epasto et al., 2021] Epasto, A., Muñoz Medina, A., Avery, S., Bai, Y., Busa-Fekete, R., Carey, C., Gao, Y., Guthrie, D., Ghosh, S., Ioannidis, J., Jiao, J., Lacki, J., Lee, J., Mauser, A., Milch, B., Mirrokni, V., Ravichandran, D., Shi, W., Spero, M., Sun, Y., Syed, U., Vassilvitskii, S., and Wang, S. (2021). Clustering for private interest-based advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 2802–2810, New York, NY, USA. Association for Computing Machinery.

[Eriksson and Hengel, 2010] Eriksson, A. and Hengel, A. (2010). Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l1 norm. pages 771 – 778.

[Fang et al., 2020] Fang, M., Gong, N. Z., and Liu, J. (2020). *Influence Function Based Data Poisoning Attacks to Top-N Recommender Systems*, page 3019–3025. Association for Computing Machinery, New York, NY, USA.

[Fang et al., 2018] Fang, M., Yang, G., Gong, N. Z., and Liu, J. (2018). Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM.

[Felício et al., 2017] Felício, C. Z., Paixão, K. V., Barcelos, C. A., and Preux, P. (2017). A multi-armed bandit model selection for cold-start user recommendation. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP, New York, NY, USA. Association for Computing Machinery.

[Frankowski et al., 2006] Frankowski, D., Cosley, D., Sen, S., Terveen, L., and Riedl, J. (2006). You are what you say: Privacy risks of public mentions. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 565–572.

[Galozy and Nowaczyk, 2023] Galozy, A. and Nowaczyk, S. (2023). Information-gathering in latent bandits. In *Know.-Based Syst.*, NLD. Elsevier Science Publishers B. V.

[Gao et al., 2023] Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., and Li, Y. (2023). A survey of graph neural networks for recommender systems: Challenges, methods, and directions. In *ACM Trans. Recomm. Syst.*, New York, NY, USA. Association for Computing Machinery.

[Gao et al., 2015] Gao, M., Tian, R., Wen, J., Xiong, Q., Ling, B., and Yang, L. (2015). Item anomaly detection based on dynamic partition for time series in recommender systems. In *PLOS ONE*, pages 1–22. Public Library of Science.

[Geyik et al., 2015] Geyik, S. C., Dasdan, A., and Lee, K.-C. (2015). User clustering in online advertising via topic models. In *arXiv preprint arXiv:1501.06595*.

[Golbandi et al., 2011] Golbandi, N., Koren, Y., and Lempel, R. (2011). Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the Fourth*

ACM International Conference on Web Search and Data Mining, WSDM, New York, NY, USA. Association for Computing Machinery.

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. In *Commun. ACM*, page 61–70, New York, NY, USA. Association for Computing Machinery.

[Gomez-Uribe, 2016] Gomez-Uribe, C. (2016). A global approach to recommendations. In *Netflix News*.

[Gomez-Uribe and Hunt, 2016] Gomez-Uribe, C. A. and Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. In *ACM Trans. Manage. Inf. Syst.*, New York, NY, USA. Association for Computing Machinery.

[Good et al., 1999] Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 439–446.

[Guarino, 2018] Guarino, B. (2018). Anti vaccine reviewers target childrens books on amazon. In *The Washington Post*.

[Harpale and Yang, 2008] Harpale, A. S. and Yang, Y. (2008). Personalized active learning for collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, New York, NY, USA. Association for Computing Machinery.

[Hawashin et al., 2018] Hawashin, B., Mansour, A., Kanan, T., and Fotouhi, F. (2018). An efficient cold start solution based on group interests for recommender systems. In *Proceedings of the First International Conference on Data Science, E-Learning and Information Systems*, New York, NY, USA. Association for Computing Machinery.

[He et al., 2021] He, S., Hollenbeck, B., and Proserpio, D. (2021). The market for fake reviews. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, EC '21, page 588.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182.

[Heidari et al., 2022] Heidari, N., Moradi, P., and Koochari, A. (2022). An attention-based deep learning method for solving the cold-start and sparsity issues of recommender systems. In *Knowledge-Based Systems*, volume 256.

[Hidano and Kiyomoto, 2020] Hidano, S. and Kiyomoto, S. (2020). Recommender systems robust to data poisoning using trim learning. In *ICISSP*, pages 721–724.

[Hofmann, 2004] Hofmann, T. (2004). Hofmann, t.: Latent semantic models for collaborative filtering. acm trans. inf. syst. (tois) 22(1), 89-115. In *ACM Trans. Inf. Syst.*, volume 22, pages 89–115.

[Hong et al., 2020] Hong, J., Kveton, B., Zaheer, M., Chow, Y., Ahmed, A., and Boutilier, C. (2020). Latent bandits revisited. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13423–13433. Curran Associates, Inc.

[Hu et al., 2019] Hu, R., Guo, Y., Pan, M., and Gong, Y. (2019). Targeted poisoning attacks on social recommender systems. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

[Huang et al., 2021] Huang, H., Mu, J., Gong, N. Z., Li, Q., Liu, B., and Xu, M. (2021). Data poisoning attacks to deep learning based recommender systems. In *Proceedings 2021 Network and Distributed System Security Symposium*. Internet Society.

[Katidis and christianbonzelet, 2022] Katidis, P. I. and christianbonzelet (2022). Use machine learning to target your customers based on their interest in a product or product attribute. In *Amazon Web Services (AWS)*.

[Ke and Kanade, 2005] Ke, Q. and Kanade, T. (2005). Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. volume 1, pages 739 – 746 vol. 1.

[Ko et al., 2022] Ko, H., Lee, S., Park, Y., and Choi, A. (2022). A survey of recommendation systems: Recommendation models, techniques, and application fields. In *Electronics*.

[Koren, 2010] Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. In *ACM Trans. Knowl. Discov. Data*, New York, NY, USA. Association for Computing Machinery.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. In *Computer*, volume 42, pages 30–37.

[Lam and Riedl, 2004] Lam, S. K. and Riedl, J. (2004). Shilling recommender systems for fun and profit. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, page 393–402, New York, NY, USA. Association for Computing Machinery.

[Lam et al., 2008] Lam, X. N., Vu, T., Le, T. D., and Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, ICUIMC, New York, NY, USA. Association for Computing Machinery.

[Lee et al., 2012] Lee, J., Sun, M., and Lebanon, G. (2012). A comparative study of collaborative filtering algorithms. In *arXiv preprint arXiv:1205.3193*.

[Li et al., 2016] Li, B., Wang, Y., Singh, A., and Vorobeychik, Y. (2016). Data poisoning attacks on factorization-based collaborative filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1893–1901, Red Hook, NY, USA. Curran Associates Inc.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news inproceedings recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW 2010, New York, NY, USA. Association for Computing Machinery.

[Lin et al., 2020] Lin, C., Chen, S., Li, H., Xiao, Y., Li, L., and Yang, Q. (2020). Attacking recommender systems with augmented user profiles. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 855–864.

[Liu et al., 2011a] Liu, N. N., Meng, X., Liu, C., and Yang, Q. (2011a). Wisdom of the better few: Cold start recommendation via representative based rating elicitation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys, pages 37–44, New York, NY, USA. Association for Computing Machinery.

[Liu et al., 2011b] Liu, N. N., Meng, X., Liu, C., and Yang, Q. (2011b). Wisdom of the better few: Cold start recommendation via representative based rating elicitation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 37–44, New York, NY, USA. Association for Computing Machinery.

[Lops et al., 2011] Lops, P., de Gemmis, M., and Semeraro, G. (2011). *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105.

[Mehta et al., 2007] Mehta, B., Hofmann, T., and Nejdl, W. (2007). Robust collaborative filtering. pages 49–56.

[Mehta and Nejdl, 2008] Mehta, B. and Nejdl, W. (2008). Attack resistant collaborative filtering. In *The 31st Annual International ACM SIGIR Conference (SIGIR)*.

[Mehta and Nejdl, 2009] Mehta, B. and Nejdl, W. (2009). Unsupervised strategies for shilling detection and robust collaborative filtering. In *User Model. User-Adapt. Interact.*, volume 19, pages 65–97.

[Meta, 2023] Meta (2023). Ad targeting: How to find people most likely to respond to your ad. In *Meta Ads*.

[Mingdan and Li, 2020] Mingdan, S. and Li, Q. (2020). Shilling attacks against collaborative recommender systems: a review. In *Artificial Intelligence Review*, volume 53.

[Mittal et al., 2010] Mittal, N., Nayak, R., Govil, M., and Jain, K. (2010). Recommender system framework using clustering and collaborative filtering. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 555–558.

[Mobasher et al., 2007] Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. (2007). Toward trustworthy recommender systems. In *ACM Transactions on Internet Technology*, volume 7, pages 23–es.

[Mobasher et al., 2005] Mobasher, B., Burke, R., Williams, C., and Bhaumik, R. (2005). Analysis and detection of segment-focused attacks against collaborative recommendation. In *Proceedings of the 7th International Conference on Knowledge Discovery on the Web: Advances in Web Mining and Web Usage Analysis*, volume 4198 of *WebKDD'05*, page 96–118.

[Narayanan and Shmatikov, 2006] Narayanan, A. and Shmatikov, V. (2006). How to break anonymity of the netflix prize dataset. In *arXiv preprint cs/0610105*.

[Nguyen et al., 2014] Nguyen, H. T., Mary, J., and Preux, P. (2014). Cold-start problems in recommendation systems via contextual-bandit algorithms. In *arXiv preprint arXiv:1405.7544*.

[Odić et al., 2013] Odić, A., Tkalčič, M., Tasič, J. F., and Košir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. In *Interacting with Computers*, pages 74–90.

[O'Donovan and Smyth, 2005] O'Donovan, J. and Smyth, B. (2005). Trust no one: Evaluating trust-based filtering for recommenders. pages 1663–1665.

[O'Mahony et al., 2006] O'Mahony, M. P., Hurley, N. J., and Silvestre, G. C. (2006). Detecting noise in recommender system databases. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, page 109–115, New York, NY, USA. Association for Computing Machinery.

[O'Mahony et al., 2002] O'Mahony, M., Hurley, N., and Silvestre, G. (2002). Promoting recommendations: An attack on collaborative filtering. In *Database and Expert Systems Applications: 13th International Conference, DEXA 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings 13*, pages 494–503. Springer.

[Park and Chu, 2009] Park, S.-T. and Chu, W. (2009). Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys 2009, New York, NY, USA. Association for Computing Machinery.

[Patel et al., 2015] Patel, K., AmitThakkar, Shah, C., and Makvana, K. (2015). A state of art survey on shilling attack in collaborative filtering based recommendation system. In *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1*, pages 377–385. Springer.

[Pérez-Marcos and Batista, 2018] Pérez-Marcos, J. and Batista, V. (2018). Recommender system based on collaborative filtering for spotify's users. In *Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection-15th International Conference, PAAMS 2017 15*, pages 214–220. Springer.

[Rajapakse and Leith, 2022] Rajapakse, D. C. and Leith, D. (2022). Fast and accurate user cold-start learning using monte carlo tree search. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 350–359, New York, NY, USA. Association for Computing Machinery.

[Rashid et al., 2002] Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., and Riedl, J. (2002). Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, New York, NY, USA. Association for Computing Machinery.

[Rashid et al., 2008] Rashid, A. M., Karypis, G., and Riedl, J. (2008). Learning preferences of new users in recommender systems: An information theoretic approach. In *SIGKDD Explor. Newsl.*, volume 10, New York, NY, USA. Association for Computing Machinery.

[Ravichandran and Vassilvitskii, 2009] Ravichandran, D. and Vassilvitskii, S. (2009). Evaluation of cohort algorithms for the floc api. In *Google Research and Ads*.

[Resnick and Sami, 2007] Resnick, P. and Sami, R. (2007). The influence limiter: Provably manipulation-resistant recommender systems. In *RecSys'07: Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 25–32.

[Resnick and Varian, 1997] Resnick, P. and Varian, H. R. (1997). Recommender systems. In *Commun. ACM*, volume 40, page 56–58, New York, NY, USA. Association for Computing Machinery.

[Rodriguez, 2017] Rodriguez, A. (2017). Netflix divides its 93 million users around the world into 1,300 "taste communities". In *Quartz*.

[Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, EC '00, page 158–167, New York, NY, USA. Association for Computing Machinery.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW, New York, NY, USA. Association for Computing Machinery.

[Shams et al., 2021] Shams, S., Anderson, D., and Leith, D. (2021). *Cluster-Based Bandits: Fast Cold-Start for Recommender System New Users*, page 1613–1616. Association for Computing Machinery, New York, NY, USA.

[Shams and Leith, 2022] Shams, S. and Leith, D. J. (2022). Improving resistance of matrix factorization recommenders to data poisoning attacks. In *2022 Cyber Research Conference - Ireland (Cyber-RCI)*, pages 1–4.

[Shi et al., 2017a] Shi, L., Zhao, W. X., and Shen, Y.-D. (2017a). Local representative-based matrix factorization for cold-start recommendation. In *ACM Trans. Inf. Syst.*, New York, NY, USA. Association for Computing Machinery.

[Shi et al., 2017b] Shi, L., Zhao, W. X., and Shen, Y.-D. (2017b). Local representative-based matrix factorization for cold-start recommendation. In *ACM Trans. Inf. Syst.*, volume 36, New York, NY, USA. Association for Computing Machinery.

[Shi et al., 2014] Shi, Y., Larson, M., and Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. In *ACM Comput. Surv.*, volume 47, New York, NY, USA. Association for Computing Machinery.

[Simonetti, 2022] Simonetti, I. (2022). Amazon fake reviews lawsuit. In *The New York Times*.

[Smith and Linden, 2017] Smith, B. and Linden, G. (2017). Two decades of recommender systems at amazon.com. In *IEEE Internet Computing*.

[Son, 2016] Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. In *Inf. Syst.*, volume 58, pages 87–104.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. (2009). A survey of collaborative filtering techniques. In *Adv. Artificial Intellegence*, volume 2009.

[Sun et al., 2013] Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., and Zha, H. (2013). Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM, New York, NY, USA. Association for Computing Machinery.

[Ungar and Foster, 1998] Ungar, L. H. and Foster, D. P. (1998). Clustering methods for collaborative filtering. In *AAAI Conference on Artificial Intelligence*.

[Williams et al., 2007] Williams, C., Mobasher, B., and Burke, R. (2007). Defending recommender systems: Detection of profile injection attacks. In *Service Oriented Computing and Applications*, volume 1, pages 157–170.

[Williams et al., 2006] Williams, C., Mobasher, B., Burke, R., Sandvig, J. J., and Bhaumik, R. (2006). Detection of obfuscated attacks in collaborative recommender systems. In *Proceedings of the ECAI*, volume 6.

[Wu et al., 2021] Wu, C., Lian, D., Ge, Y., Zhu, Z., Chen, E., and Yuan, S. (2021). *Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training*, page 1074–1083. Association for Computing Machinery, New York, NY, USA.

[Wu et al., 2020] Wu, D., Lu, G., and Xu, Z. (2020). Robust and accurate representation learning for high-dimensional and sparse matrices in recommender systems. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pages 489–496.

[Xie and Phoha, 2001] Xie, Y. and Phoha, V. V. (2001). Web user clustering from access log using belief function. In *Proceedings of the 1st International Conference on Knowledge Capture*, K-CAP '01, page 202–208, New York, NY, USA. Association for Computing Machinery.

[Xiong et al., 2011] Xiong, L., Chen, X., and Schneider, J. (2011). Direct robust matrix factorization for anomaly detection. In *2011 IEEE 11th International Conference on Data Mining*, pages 844–853.

[Xu et al., 2020] Xu, Y., Chen, L., Xie, F., Hu, W., Zhu, J., Chen, C., and Zheng, Z. (2020). Directional adversarial training for recommender systems. In *ECAI*, pages 553–560.

[Xue et al., 2005] Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., and Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121.

[Xue et al., 2017] Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., and Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th*

International Joint Conference on Artificial Intelligence, IJCAI'17, page 3203–3209. AAAI Press.

[Yang et al., 2018] Yang, Z., Sun, Q., and Zhang, B. (2018). Evaluating prediction error for anomaly detection by exploiting matrix factorization in rating systems. In *IEEE Access*, volume 6, pages 50014–50029.

[Yanxiang et al., 2013] Yanxiang, L., Deke, G., Fei, C., and Honghui, C. (2013). User-based clustering with top-n recommendation on cold-start problem. In *2013 Third International Conference on Intelligent System Design and Engineering Applications*, pages 1585–1589.

[Zhang et al., 2006] Zhang, S., Chakrabarti, A., Ford, J., and Makedon, F. (2006). Attack detection in time series for recommender systems. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 809–814.

[Zheng et al., 2012] Zheng, Y., Liu, G., Sugimoto, S., Yan, S., and Okutomi, M. (2012). Practical low-rank matrix approximation under robust l1-norm. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1417.

[Zhou et al., 2011] Zhou, K., Yang, S.-H., and Zha, H. (2011). Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2011, New York, NY, USA. Association for Computing Machinery.