Robustness of causal recommender systems against shilling attacks

Apara Venkat Kenny Zhang Ruoyi Zhu

STAT 566 Spring 2021
University of Washington

Abstract

The goal of a recommender system is to collect information about preferences of different users and suggest new items that the user will like. Traditionally, this has been framed as a prediction problem – what items will the user like? But there has been work approaching this as an intervention problem using a causal perspective – what will the rating be if the user is forced to use the item? Recommender systems face an issue where malicious people inject fake data in order to bias the recommendation results for their own gain. The vulnerability of traditional recommender systems to these *shilling attacks* is well documented. In this paper, we perform a simple test to show that the deconfounded causal recommender is significantly more robust to shilling attacks. We specifically show this for movie recommendations using the MovieLens dataset. However we believe that our observation extends to other scenarios as well. We conclude by providing directions for future work including developing better metrics to assess performance of causal recommenders and addressing important controversies surrounding the deconfounded model.

1 Introduction

The amount of data available to us is increasing at a faster rate than possible to digest and make sense of it. This is especially true when it comes to making decisions. We are often presented with innumerable choices. For example, searching for "notebook" on Amazon returns over 10,000 items and digital movie and music streaming services have given us instantaneous access to over a billion items. Given that making these choices is expensive in terms of time, money, and other resources, it is difficult to sift through the plethora of choices presented to us.

This is where recommender systems come into play. The goal of a recommender system is to collect information about preferences of different users and suggest new items that the user will like. Recommendation techniques have achieved great success in industrial applications. Traditional recommendation methods attempt to answer a predictive question – what item will the user like? In recent years, efforts have been made to expand the recommendation strategy beyond this. In particular, a causal perspective frames this as an intervention question about *potential outcomes*.

As a motivating example, consider the problem of movie recommendation. Classically, we want to predict which movies a user will like. Under a causal lens, we want to predict what the rating would be if a user is forced to watch a movie. This viewpoint gives us access to a completely new toolbox. In particular, it is immediate that people usually do not watch movies at random. They are more likely to watch a movie that they think they will like. So there are observed (demographic information like age, region, and gender) and unobserved confounders that affect both which movies the users are exposed to and their ratings. Consequently, we can conclude that the traditional recommendation model gives biased results, which the causal recommender attempts to correct.

Recommender systems face another problem. There are malicious attackers who seek to bias the recommendations for their own gain. This vulnerability is due to the open nature of recommender systems and their reliance on user-specified information [Mobasher et al., 2005]. It is well known that classical recommender systems are vulnerable to these kinds of *shilling attacks*. In this project, we want to study the robustness of causal recommender systems against the same shilling attacks. In particular, we focus our attention on the model developed by [Wang et al., 2020].

The rest of the report is organized as follows: in Section 2, we review existing literature on recommender systems and shilling attacks. In Section 3, we describe the dataset we use in our analysis. Section 4 introduces the causal recommender, followed by results from our analysis in Section 5. Finally, in Section 6, we conclude our discussion with ideas for future work.

2 Previous Work

Often, recommender systems have information about users, items, and ratings to work with. The system then automatically removes unwanted information before presenting relevant items to users. Classically, there are three distinct approaches to filter information – demographic filtering, content-based filtering, and collaborative filtering [Seyednezhad et al., 2018, Candillier et al., 2007, Adomavicius and Tuzhilin, 2005]. By "classical," we mean that these methods attempt to answer the predictive question, "what will the user like?" Demographic (or user-based) filtering recommends items based on users with similar attributes with the intuition that similar people will have similar preferences. Content-based filtering uses at-

tributes of items that users have liked in the past with the intuition that users have a specific taste that does not drastically change over time [Van Meteren and Van Someren, 2000]. Finally, collaborative filtering uses actual user ratings of items to recommend new items. Collaborative filtering is a very popular approach [Herlocker et al., 2004, Salter and Antonopoulos, 2006]. There are also hybrid methods that use a combination of the three approaches. In our project, we will concern ourselves with the collaborative filtering approach, specifically the latent factor model, as the baseline "classical recommender."

A lot of attention has been devoted to answering the causal question, "what would the user rating be if they had used the item?" This includes work surrounding the violation of missing-at-random assumption in recommendation algorithms [Marlin and Zemel, 2009, Bonner and Vasile, 2018, Hernández-Lobato et al., 2014]. While the violation of this assumption is a confounding bias, these methods do not take an explicit causal perspective. [Schnabel et al., 2016] adopt an explicit causal perspective and use inverse propensity weighting to address missingness. Finally, we turn our attention to [Wang et al., 2020], who take an explicit causal perspective and frame the recommendation question as a multiple causal inference problem. This allows them to use the deconfounder method [Wang and Blei, 2019]. In this project, we will concern ourselves with the deconfounded recommender as the "causal recommender" and will describe this model in detail in Section 4.

Attacks on recommender systems are mainly categorized as data poisoning attacks and profile pollution attacks. Data poisoning attacks uses fake user profiles with carefully crafted ratings injected to the recommender system with a goal to promote target item [Deng et al., 2016]. Profile pollution attack injects information into users' profiles to perturb results obtained from services using personalization algorithms [Xing et al., 2013]. We will primarily focus on the data poisoning attacks (also called shilling attacks). Shilling attack models are mainly of 5 types [Li Yang, 2017]: random attack, average attack, bandwagon attack, segmented attack, and sampling attack. This categorization of attack models is done based on the way in which items and ratings are chosen for each fake profile. Every attack aims to create fake profiles with ratings for specific items. When these fake profiles are fed into recommender systems, its behavior can be altered. We describe the attacks in more detail in Section 5.

There has been work done on detecting shilling attacks. Some particularly interesting ones include time series analysis [Zhou et al., 2018, Xu and Zhang, 2019] and k-means clustering [Bilge et al., 2014]. There are also ways to use neural networks to detect shilling attacks [Tong et al., 2018, Vivekanandan and Praveena, 2021]. People have also studied how to prevent shilling attacks [Chirita et al., 2005]. And there is literature on comparing the effectiveness of different traditional recommender systems against shilling attacks [Kaur and Goel, 2016, Si and Li, 2020]. These comparisons find that although collaborative filtering is significantly affected, it is the most robust classical recommender against shilling attacks.

Clearly causal recommender systems and developing recommender systems that are robust to shilling attacks have been studied extensively. However, since the causal recommender is relatively new, these bodies of work are largely independent of each other. To our knowledge,

there has been no previous work studying the robustness of causal recommender systems to shilling attacks.

3 Data

In this project, we performed analysis on the MovieLens Data [Grouplens, 1998]. The dataset contains 100,000 ratings on 1682 movies from 943 users. Each user rated 1 through 5 on at least 20 movies. We have demographic information about each user – age, gender, occupation, and zip code for the users. We also have information about the release date and genre of each movie. A movie could belong to multiple genre. We also have IMDb URLs for each movie that allows us to fetch meta information. The 100,000 ratings were split into training (80%) and test (20%) data.

To simulate shilling attacks, we simulate 50 fake users who rate 30 movies (on average) each. We chose 3 movies with the lowest average rating as the target movies the fake users promote. The remaining movies were chosen at random for each user and the rating was the average rating of that particular movie. We describe this in more detail in Section 5.

4 Model

In this section, we describe the deconfounded recommender as developed by [Wang et al., 2020]. In particular, we also explicitly derive the estimation routine.

4.1 Classical recommender

First, we will introduce the classical probabilistic matrix factorization. Following [Wang et al., 2020], let a_{ui} be the indicator whether user u has watched (been exposed/treated to) movie i. Using the potential outcome notation, let $y_{ui}(1)$ denote the rating that user u would have given movie i had they watched it. This is the observed rating. And let $y_{ui}(0)$ denote the rating that user u would have given movie i had they not watched it. We often "observe" $y_{ui}(0) = 0$.

Assuming there are U users and I movies, the observed data consists of $\{y_{ui}(a_{ui}) : a_{ui} = 1\}_{u=1,i=1}^{U,I}$. The goal of the recommender system is to estimate $y_{ui}(1)$ for user-movie pairs where $a_{ui} = 0$, and use these estimates to recommend movies. In the language of causal inference, this estimation corresponds to prediction of the counterfactuals under intervention.

In the probabilistic matrix factorization method, the prediction is made by the following

model, which [Wang et al., 2020] refer to as the *outcome model*:

$$y_{ui}(a) = \theta_u^\top \beta_i \cdot a + \epsilon_{ui}, \qquad \epsilon \overset{\text{iid}}{\sim} N(0, \sigma^2) \tag{1}$$

Here, θ_u and β_i are latent vectors that represent the user preferences and the movie attributes. Note that when $a = 1$, the potential rating is a linear combination of user preferences and movie attributes. And when $a = 0$, the potential rating is a mean-zero Gaussian. We first learn these latent features through MAP estimation using spherical Gaussian priors and use them to predict $y_{ui}(1)$ whenever $a_{ui} = 0$.

4.2 Deconfounded recommender

The problem with probabilistic matrix factorization is that it does not give us an unbiased estimate of the potential outcome $y_{ui}(1)$. This is because we need to assume ignorability to get unbiased estimates but we cannot assume ignorability. In other words, $\{\boldsymbol{y_u}(1), \boldsymbol{y_u}(0)\} \not\perp \boldsymbol{a_u}$ where $\boldsymbol{y_u} = [y_{u1}, \ldots, y_{uI}]$ and $\boldsymbol{a_u} = [a_{u1}, \ldots, a_{uI}]$. Intuitively this makes sense because people are more likely to watch a movie that they think they will like i.e., $\boldsymbol{y_u}(1)$ and $\boldsymbol{a_u}$ are dependent. Since we cannot assume ignorability, the model in Equation 1 gives us biased estimates. In order to debias these estimates, we will need to measure and control for *all* per-user confounders w_u. However, this is both untestable and practically infeasible [Holland et al., 1985].

To circumvent this issue, [Wang et al., 2020] treat this problem as a multiple causal inference problem and use results from [Wang and Blei, 2019] to obtain an unbiased estimate of the potential ratings. They develop a two-stage model that uses the exposures $\boldsymbol{a_u}$ as evidence for the unobserved confounders. The first stage is the *exposure model*, and the second stage is the *outcome model*.

4.2.1 Exposure model

The authors fit a Poisson factorization model [Gopalan et al., 2015] to the exposure data by assuming that the data come from the following distribution

$$\begin{aligned} a_{ui} \mid \pi_u, \lambda_i &\sim \text{Poisson}(\pi_u^\top \lambda_i) \\ \pi_u \overset{\text{iid}}{\sim} \text{Gamma}(c_1, c_2), \quad \lambda_i &\overset{\text{iid}}{\sim} \text{Gamma}(c_3, c_4) \end{aligned} \tag{2}$$

Here $\pi_u, \lambda_i \in \mathbb{R}^K$ capture the user preferences and item attributes respectively where K is the pre-determined dimension of the latent factors. Using the fitted model, we can reconstruct the exposure matrix $\widehat{a}$ as the posterior mean

$$\widehat{a}_{ui} = \mathbb{E}_{\text{PF}}\left[\pi_u^\top \lambda_i \mid \boldsymbol{a}\right] \tag{3}$$

where $\boldsymbol{a}$ is the observed exposures for all users. From *Lemma 3* of [Wang and Blei, 2019], $\widehat{a}$ acts as a substitute confounder by capturing the unobserved confounders.

Algorithm 1 MAP estimation for causal recommender using alternating least squares

Given: $\widehat{a}_{ui}$ for all user u and item i. All updates are restricted to $\Omega_{R_{u,i}}$.
procedure MAP
  for iteration $\leftarrow$ 1 to MAX_ITER **do**
    for each item i **do** ▷ update item matrix $\boldsymbol{\beta}$
      $\boldsymbol{y_i} \leftarrow$ all the ratings for item i
      $\widehat{\boldsymbol{a}}_{\boldsymbol{i}} \leftarrow$ all the estimated confounders for item i
      $\boldsymbol{\gamma} \leftarrow$ parameter vector
      $\beta_i \leftarrow$ is the i^{th} row of matrix $\boldsymbol{\beta}$
      $\beta_i \leftarrow \left(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \lambda \boldsymbol{I}\right)^{-1} (\boldsymbol{\theta}^\top \boldsymbol{y_i} - \boldsymbol{\theta}^\top (\boldsymbol{\gamma} \circ \widehat{\boldsymbol{a}}_{\boldsymbol{i}}))$ ▷ $\circ$: element-wise product
    end for
    for each user u **do** ▷ update user matrix $\boldsymbol{\theta}$
      $\boldsymbol{y_u} \leftarrow$ all the ratings for user u
      $\widehat{\boldsymbol{a}}_{\boldsymbol{u}} \leftarrow$ all the estimated confounders for user u
      $\theta_u \leftarrow$ is the u^{th} row of matrix $\boldsymbol{\theta}$
      $\theta_u \leftarrow \left(\boldsymbol{\beta}^\top \boldsymbol{\beta} + \lambda \boldsymbol{I}\right)^{-1} (\boldsymbol{\beta}^\top \boldsymbol{y_u} - \gamma_u \boldsymbol{\beta}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}})$
      $\gamma_u \leftarrow (\boldsymbol{y_u}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}} - \boldsymbol{\beta} \boldsymbol{\theta}_u^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}}) / (\widehat{\boldsymbol{a}}_{\boldsymbol{u}}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}})$
    end for
  end for
  return $\boldsymbol{\theta}, \boldsymbol{\beta}$
end procedure

4.2.2 Outcome model

Conditioned on the substitute confounders, we have the outcome model,

$$y_{ui}(a) = \theta_u^\top \beta_i \cdot a + \gamma_u \cdot \widehat{a}_{ui} + \epsilon_{ui}, \qquad \epsilon \overset{\text{iid}}{\sim} N(0, \sigma^2) \tag{4}$$

where γ_u tells us the contribution of the substitute confounder on the potential ratings. We infer the parameters $\theta_u, \beta_i, \gamma_u$ using MAP estimation, which we describe in Section 4.3. After learning these parameters, we predict the potential ratings $y_{ui}(1)$. *Theorem 8* of [Wang and Blei, 2019] state that the predicted rating will be an unbiased estimate.

4.3 Estimation algorithms

We now describe our implementation of the causal recommender. We use the Poisson factorization from [Wang et al., 2020] and derive our own MAP estimation and optimization procedure to implement the model. Let $\Omega_{R_{u,i}}$ be the set that ratings are available for user u and item i. We define the loss function we want to minimize as

$$\ell = \frac{1}{2} \left(\sum_{u=1}^{U} \sum_{i=1}^{I} (y_{ui} - \theta_u^\top \beta_i - \gamma_u \widehat{a}_{ui})^2_{(u,i) \in \Omega_{R_{u,i}}} + \lambda (\sum_u \|\theta_u\|^2 + \sum_i \|\beta_i\|^2) \right) \tag{5}$$

Algorithm 2 Alternating least squares algorithm for probabilistic matrix factorization

procedure ALS
  for iteration $\leftarrow$ 1 to MAX_ITER **do**
    for each item i **do** ▷ update item matrix $\boldsymbol{Q}$
      $r_i \leftarrow$ all the preferences for item i
      $q_i \leftarrow \left(\boldsymbol{P}^\top \boldsymbol{P} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{P}^\top r_i$
    end for
    for each user u **do** ▷ update user matrix $\boldsymbol{P}$
      $r_u \leftarrow$ all the preferences for user u
      $p_u \leftarrow \left(\boldsymbol{Q}^\top \boldsymbol{Q} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{Q}^\top r_u$
    end for
  end for
  return $\boldsymbol{P}, \boldsymbol{Q}$
end procedure

where $\widehat{a}_{ui}$ is estimated from the Poisson factorization exposure model and γ_u is the latent parameter adjusts the exposure for each user. Note that minimizing the loss corresponds to maximizing the log-likelihood of the potential ratings when assuming spherical Gaussian priors for the parameters.

The gradient with respect to parameters are

$$\nabla_{\beta_i} \ell = \sum_{u=1}^{U} (y_{ij} - \theta_u^\top \beta_i - \gamma_u \widehat{a}_{ui}) \theta_u^\top - \lambda \beta_i$$
$$\nabla_{\theta_u} \ell = \sum_{i=1}^{I} (y_{ij} - \theta_u^\top \beta_i - \gamma_u \widehat{a}_{ui}) \beta_j^\top - \lambda \theta_u$$
$$\nabla_{\gamma_u} \ell = \sum_{i=1}^{I} (y_{ij} - \theta_u^\top \beta_i - \gamma_u \widehat{a}_{ui}) \widehat{a}_{ui}$$

Setting the gradients to zero gives updates on each parameter in matrix form are

$$\beta_i \leftarrow \left(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \lambda \boldsymbol{I}\right)^{-1} (\boldsymbol{\theta}^\top \boldsymbol{y_i} - \boldsymbol{\theta}^\top (\boldsymbol{\gamma} \circ \widehat{\boldsymbol{a}}_{\boldsymbol{i}}))$$
$$\theta_u \leftarrow \left(\boldsymbol{\beta}^\top \boldsymbol{\beta} + \lambda \mathbf{I}\right)^{-1} (\boldsymbol{\beta}^\top \boldsymbol{y_u} - \gamma_u \boldsymbol{\beta}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}})$$
$$\gamma_u \leftarrow (\boldsymbol{y_u}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}} - \boldsymbol{\beta} \boldsymbol{\theta}_u^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}}) / (\widehat{\boldsymbol{a}}_{\boldsymbol{u}}^\top \widehat{\boldsymbol{a}}_{\boldsymbol{u}})$$

where $\circ$ denotes element-wise product. Algorithm 1 describes this procedure.

Algorithm 2 briefly describes the alternating least squares algorithm for the classical probabilistic matrix factorization (latent factor) model. We use the collaborative filtering implementation from `spark.ml` package in Python.

5 Results

In this section, we describe the attack profiles for the shilling attacks and discuss our results.

5.1 Attack Profiles

To implement shilling attacks, we need to create fake user profiles that resemble the real users and insert them into the data. There are various types of shilling attacks including random attack, average attack, bandwagon attack, and segment attack [Kumari and Bedi, 2017]. The major difference among these attacks is the composition of attack profiles.

Let I_S denote the selected item set, the set of items identified by the attacker to exploit the owned knowledge of system to maximize the effectiveness of the attack (for instance set of popular items). Let I_F denote the filler item set, the set of randomly selected items for which rating scores are assigned from normal distribution of ratings to make the attack imperceptible. Let I_ϕ denote the unrated item set, set of items without ratings in the fake user profile. And let I_T denote the target item set, set of target item(s) is to promote or demote. Then Shilling Profile (SP) is defined as

$$\mathcal{S} = (I_S, I_F, I_\phi, I_T)$$

The SP composition varies based on attack strategies. Figure 1 represents a basic structure of shilling profile taken from [Bilge et al., 2014].

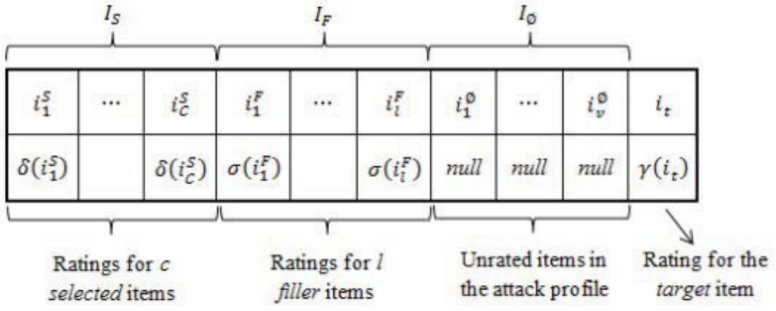

Figure 1: Example shilling profile taken from [Bilge et al., 2014]. I_S on the left denotes the selected item set. I_F denotes the filler items. I_ϕ denotes the items unrated by the attacker. And I_T denotes the targeted items.

As allued to in Section 3, the shilling attack that we implemented is average attack, which consists of 3 target items whose IDs are $[678, 235, 210]$ in our original data set. All three items have low average rating. The goal of the shilling attack is to promote these items.

To create attack profile, first we create 50 fake users ($\approx 5\%$ of the original size of our user base) and place these three target items in each of the fake user profile. Then we randomly choose the number of filler movies rated by each fake user weighted by the number of movies each genuine user has rated in our dataset. Then we randomly selected these filler movies. The fake user then rates each filler movie with the rounded average score of the movie from the original dataset. We assume that this information is available to the attacker (Amazon, IMDb, etc. publish the average ratings from users). In this attack, we set $I_S = \varphi$ i.e., the set of selected movies is empty.

5.2 Evaluation Metric

Let $R_u^{(k)}$ be the set of top k recommendations for each user u. For each promotion attack on item i, the value of a recommendation hit for user u denoted by $H_{ui}^{(k)} = \mathbb{I}(i \in R_u^{(k)})$. In other words, $H_{ui}^{(k)}$ is 1 if the movie i is in the top k recommendations, and 0 otherwise. We define *hit ratio* for movie i, $\mathcal{H}_i^{(k)}$, as the number of hits across all users in the test set divided by the total number of users in the test set. Then, hit ratio for the targeted item i over all users for k top recommendations is

$$\mathcal{H}_i^{(k)} = \frac{1}{U} \sum_{u=1}^{U} H_{ui}^{(k)} \tag{6}$$

Then, we define the *average hit ratio*, $\mathcal{H}^{(k)}$, over all the target movies I_T as

$$\mathcal{H}^{(k)} = \frac{1}{|I_T|} \sum_{i \in I_T} \mathcal{H}_i^{(k)} = \frac{1}{|I_T| \cdot U} \sum_{i \in I_T} \sum_{u=1}^{U} H_{ui}^{(k)} \tag{7}$$

Here, $|I_T|$ is the total number of target movies and U is the total number of users. This metric $\mathcal{H}^{(k)}$ corresponds to the average percentage of users to whom the target item will be recommended among top k recommendations after the attack.

As a final remark, [Wang et al., 2020] note that a metric like root mean squared error (of predicted potential ratings), when naïvely used, may lead to biased results unless it is computed on a truly randomized test set. This because we need to evaluate our performance against all potential outcomes to get unbiased errors. Unfortunately, the MovieLens dataset does not have a randomized test set. Hence we evaluate performance only using the average hit ratio.

5.3 Results

To briefly recap, we first use the original to compute the average hit ratio after fitting the classical and causal recommender. This corresponds to the metrics "before" the attack.

Then, we repeat this on the data containing the attack profiles. The average hit ratio here corresponds to "after" the attack.[1]

The average hit ratio corresponding to top 10 recommendations is shown in Table 1. As we can see, the hit ratio before the attack is very low initially for both models. However, after the attack, the classical recommender (latent factor model) has $\approx 5\%$ average hit ratio. In other words, on average, each targeted item appears in the top 10 recommendations for 4.7% of the users after the attack. In fact if we take a look at hit ratio separately for each item, item 210 in our original dataset has a significant hit ratio about 13% while the other two target items have hit ratio still quite low. This means the attack is partially successful on the classical recommender system in promoting one of the target item. Meanwhile the causal recommender is more robust towards the attack as item 210 only appears in about 1% (1 out of 100) users' top 10 recommendations with an average hit ratio of 0.35% for all target items. This means the attack is not very successful on the causal recommender.

	Before attack	**After attack**
Classical recommender	0.35%	4.7%
Causal recommender	0.00%	0.35%

Table 1: Average hit ratio (in %) among top $k = 10$ recommendations.

We further generalize the hit ratio to top k recommendations to different k to see the behaviour of the two recommendation algorithms. The results are shown in Figure 2. We see two trends on this plot. First, the causal model is robust against shilling attacks even if we take k to be large. The target items only get promoted minimally after the attack. Second, in the classical recommender, after the attack, item 210 appears in almost half of users' top 50 recommendations. Therefore, we can conclude that the causal model is unlikely to promote these low rating items.

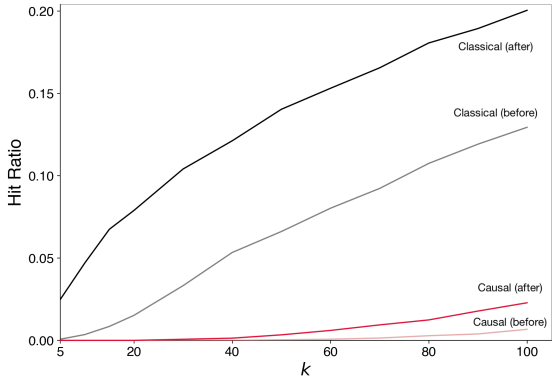

Figure 2: Hit ratio of classical and causal recommender systems before and after the shilling attacks

[1]Our code is available at https://github.com/AparaV/566-causal-recommender.

6 Conclusion and future directions

To conclude our study, we found that causal recommenders are more robust to shilling attacks than traditional latent factor models (in particular average attack that we have implemented). Also causal recommenders inherit the simplicity and interpretability of the classical recommender with its modification of the exposure phase. It can be a very good algorithm for building recommenders that need to be interpretable and robust against shilling attacks. We want to remark that we performed a very simple test on one dataset. Given more time, we would have liked to perform other kinds of shilling attacks and explore other datasets. This will allow us to more accurately characterize the sensitivity of causal recommenders against shilling attacks.

We have identified three areas for future study to address some limitations of our work. For causal recommenders, the traditional error metrics like mean squared error (MSE) on random split of train and test set are biased towards popular items and active users. Since we don't observe all potential outcomes, these metrics are biased for the causal recommender [Wang et al., 2020]. This makes it difficult to compare causal and classical recommenders. This is one area of interest that could be worked on – developing metrics for causal recommender to evaluate based on traditional data sets. For example, we can find clever ways to weight the metric based on $\widehat{a}_{ui}$ so that we obtain an unbiased estimate of the MSE to compare with classical recommenders.

Another area of improvement is utilizing or incorporating observed user and item covariates. For example, the MovieLens Data [Grouplens, 1998] contains simple demographic info for the users (age, gender, occupation, zip). The deconfounded recommender infers latent features for users and items that supposedly capture these observed covariates [Wang et al., 2020]. However, it may be of interest to explicitly incorporate these covariates for interpretability.

Finally, we want to acknowledge that there are controversies in the causal inference literature on whether the exposure model will capture all multi-treatment confounders. In particular, [Ogburn et al., 2019] commented that the latent variable methods to control for unmeasured confounding proposed in [Wang and Blei, 2019] "has foundational errors." They also provide counterexamples and arguments against the lemmas and theorem on which the deconfounded recommender rely upon. They state that no fact about the observed data alone can be informative about ignorability. While the causal recommender seems to be more robust to shilling attacks, [Ogburn et al., 2019] raise serious concerns. And this leaves us with an open can of worms.

References

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

[Bilge et al., 2014] Bilge, A., Ozdemir, Z., and Polat, H. (2014). A novel shilling attack detection method. *Procedia Computer Science*, 31:165–174.

[Bonner and Vasile, 2018] Bonner, S. and Vasile, F. (2018). Causal embeddings for recommendation. In *Proceedings of the 12th ACM conference on recommender systems*, pages 104–112.

[Candillier et al., 2007] Candillier, L., Meyer, F., and Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 548–562. Springer.

[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*, pages 67–74.

[Deng et al., 2016] Deng, Z.-J., Zhang, F., and Wang, S. P. (2016). Shilling attack detection in collaborative filtering recommender system by pca detection and perturbation. In *2016 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, pages 213–218. IEEE.

[Gopalan et al., 2015] Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical poisson factorization. In *UAI*, pages 326–335.

[Grouplens, 1998] Grouplens (1998). Movielens 100k dataset. data retrieved from Grouplens, https://grouplens.org/datasets/movielens/100k/.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

[Hernández-Lobato et al., 2014] Hernández-Lobato, J. M., Houlsby, N., and Ghahramani, Z. (2014). Probabilistic matrix factorization with non-random missing data. In *International Conference on Machine Learning*, pages 1512–1520. PMLR.

[Holland et al., 1985] Holland, P. W., Glymour, C., and Granger, C. (1985). Statistics and causal inference. *ETS Research Report Series*, 1985(2):i–72.

[Kaur and Goel, 2016] Kaur, P. and Goel, S. (2016). Shilling attack models in recommender system. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–5. IEEE.

[Kumari and Bedi, 2017] Kumari, T. and Bedi, P. (2017). A comprehensive study of shilling attacks in recommender systems. *International Journal of Computer Science Issues (IJCSI)*, 14(4):44.

[Li Yang, 2017] Li Yang, Wei Huang, X. N. (2017). Defending shilling attacks in recommender systems using soft co-clustering. *IET Information Security*.

[Marlin and Zemel, 2009] Marlin, B. M. and Zemel, R. S. (2009). Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12.

[Mobasher et al., 2005] Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. (2005). Effective attack models for shilling item-based collaborative filtering systems. In *Citeseer*.

[Ogburn et al., 2019] Ogburn, E. L., Shpitser, I., and Tchetgen, E. J. T. (2019). Comment on "blessings of multiple causes". *Journal of the American Statistical Association*, 114(528):1611–1615.

[Salter and Antonopoulos, 2006] Salter, J. and Antonopoulos, N. (2006). Cinemascreen recommender agent: combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41.

[Schnabel et al., 2016] Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., and Joachims, T. (2016). Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pages 1670–1679. PMLR.

[Seyednezhad et al., 2018] Seyednezhad, S., Cozart, K. N., Bowllan, J. A., and Smith, A. O. (2018). A review on recommendation systems: Context-aware to social-based. *arXiv preprint arXiv:1811.11866*.

[Si and Li, 2020] Si, M. and Li, Q. (2020). Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review*, 53(1):291–319.

[Tong et al., 2018] Tong, C., Yin, X., Li, J., Zhu, T., Lv, R., Sun, L., and Rodrigues, J. J. (2018). A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network. *The Computer Journal*, 61(7):949–958.

[Van Meteren and Van Someren, 2000] Van Meteren, R. and Van Someren, M. (2000). Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56.

[Vivekanandan and Praveena, 2021] Vivekanandan, K. and Praveena, N. (2021). Hybrid convolutional neural network (cnn) and long-short term memory (lstm) based deep learning model for detecting shilling attack in the social-aware network. *Journal of Ambient Intelligence and Humanized Computing*, 12:1197–1210.

[Wang and Blei, 2019] Wang, Y. and Blei, D. M. (2019). The blessings of multiple causes. *Journal of the American Statistical Association*, 114(528):1574–1596.

[Wang et al., 2020] Wang, Y., Liang, D., Charlin, L., and Blei, D. M. (2020). Causal inference for recommender systems. In *Fourteenth ACM Conference on Recommender Systems*, pages 426–431.

[Xing et al., 2013] Xing, X., Meng, W., Doozan, D., Snoeren, A. C., Feamster, N., and Lee, W. (2013). Take this personally: Pollution attacks on personalized services. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 671–686.

[Xu and Zhang, 2019] Xu, Y. and Zhang, F. (2019). Detecting shilling attacks in social recommender systems based on time series analysis and trust features. *Knowledge-Based Systems*, 178:25–47.

[Zhou et al., 2018] Zhou, W., Wen, J., Qu, Q., Zeng, J., and Cheng, T. (2018). Shilling attack detection for recommender systems based on credibility of group users and rating time series. *PloS one*, 13(5):e0196533.