Model Inversion Attack on Item-Item Collaborative Filtering

Sreekar Dhaduvai, Dr. Sahebi Shaghayegh

Abstract

Advertisements displayed online might seem benign at a first glance. But why exactly was a specific advertisement shown to you? If an advertisement from an E-commerce company was displayed at a random website you visited, chances are, it will feel very relevant to you. It is because, companies use a recommender system which looks at your purchase history and advertises an item which they think you will likely purchase. To common public, this feels harmless and maybe even helpful. In this report, we estimate if a cyber attacker has knowledge of items recommended to you, can they retrieve your purchase history.

Introduction

Amazon utilizes an item-item collaborative filtering recommender [1] which recommends you items that share a similarity with the items in your purchase history. If Amazon observes that many people who have purchased an Iphone are also purchasing an Apple Watch, a new user who has just purchased an Iphone will be recommended an Apple Watch. This recommendation shows up in many places like the User's Amazon Homepage, as an advertisement on their Facebook newsfeed or on any random website they might visit. It is fairly easy for a human to infer the purchase history of an Iphone given that their target was shown an Apple Watch advertisement, but here we try to alleviate the manual task of the attacker by employing machine learning models. We tested our approach against two types of attack algorithms – Item-Item Collaborative Filtering and Ranking Factorization Machine to see which one could aid the attacker the most. The results show that item-item collaborative filtering is more efficient in guessing items in a target's purchase history and ranking matrix factorization is more efficient in predicting ratings of unknown (user, item) pairs.

Related Work

In “Item-Based Collaborative Filtering Recommendation Algorithms” [1], the procedure used by our victim model is discussed. It employs a data discovery technique wherein, the known (user, item, rating) pairs are used to model an estimated rating for an unknown (user, item) pair by calculating the similarity between items in the database. This similarity is calculated using various measures like cosine similarity, jaccard similarity and pearson similarity.

“Matrix Factorization Techniques for Recommender Systems” [2], outlines the technique of extracting latent user features and latent item features from the data matrix using singular value decomposition. Every user vector and item vector have a bias term associated with them which are all optimized when we try to optimize the prediction loss equation. This technique is useful in modelling complex relationships between several variables in the dataset. This technique was implemented in the Netflix Competition and got very good results.

In the paper titled “Badvertisements: Stealthy Click-Fraud with Unwitting Accessories” [3], the author clearly illustrates that a webpage administrator can incorporate malicious javascript code in their website to display an ally’s ad at the cost of an adversary’s advertising budget. When an unsuspecting user visits their webpage, a CGI script is used to display ads of the web-admin’s desire, but a stealthy click operation is performed on the adversary’s ad without it actually being displayed. “Spidering” is a technique used by Google to crawl webpages and make sure that it is not listing any sites which violate its terms of service. To counter this policing, the stealthy web-admin incorporates a dual-personality page where the webpage with malicious javascript is shown only if they are sure that it’s an unwitting user visting the page. If it could be a policing attempt, the good webpage is displayed. The paper also talks about how any web-admin can do this with only menial knowledge of page-preprocessing and it also points out how personalized advertisements displayed on a webpage can be downloaded after they are rendered for a specific user.

In “Browsing Pattern Analysis: What user browsing Patterns Indicate” [4], they built a tool which can track the behavior exhibited by a person in their online

environment and place them in a respective demographic depending on which, relevant advertisements and search-engine queries are recommended to them. The user on whom on whom they tested this tool seemed to frequent websites pertaining to “entertainment” a lot and so the corresponding advertisements were made to them.
A person might actively participate a lot on a forum where they exercise a lot of their natural writing prowess using their actual credentials. But on other online platforms they might express their views under anonymity thinking that it can’t be traced back to them. But in “You Are What You Say: Privacy Risks of Public Mentions” [5], they talk about how a machine learning model could be built using TF-IDF word vectors of the samples signed with actual credentials and then use it to associate the anonymous mentions to their actual authors.

Everything is interconnected through web these days. A Youtube video you watch on your mobile device can influence the ads displayed to you on your computer. In “Cross-Device Tracking: Measurement and Disclosures” [6], they built a tool to verify that a website can have the capability to track a user’s movements across various devices. If a user has logged into a website with proper credentials using their computer and the computer sooner or later happens to be under the same WiFi as their other electronic devices like phone, laptop etc, the website now has the knowledge that many devices belong in the possession of this user.

“You Might Also Like: Privacy Risks of Collaborative Filtering” [7], talks about the most frequent kind of attacks witnessed by a recommender system. Shilling Attack refers to the attack where a malicious actor creates fake profiles in a large quantity and uses them to rate certain sets of items very highly by these fake profiles so that one item of interest to the malicious actor is recommended frequently to people who buy one other naturally popular item.

In “Collaborative Filtering for Implicit Feedback Datasets” [9], a factorization model is discussed which utilizes both explicit feedback like ratings and implicit feedback like user’s name and geolocation to learn latent representations for more efficient item recommendation. Rating prediction for unobserved (user, item) pairs is calculated using a weighted combination of user and item latent factors.

When black box access is given to a machine learning model, there can be an information leakage about the training records used for that model [10]. Several adversarial models are trained using a portion of the training data and the input which is given to the black box model, is also given to these adversarial models and the variations in responses generated by them is used to infer the presence of a training record for the actual black box model.

Phishers like to maximize the yield on their target by getting as much information on them possible. The browser cache is rife with information on what kinds of services they use. Armed with this knowledge, the phisher personalizes an attack making sure it maximizes the chances of success on their victim [11].

Attack Scenarios

There are several ways in which an attacker could get information about items recommended to a target. Modern digitalization has left cyber attackers many ways to exploit their targets. In this report, we will be mostly focusing on a scenario where the attacker has access only to a target's product recommendations and no other information. This can be achieved by the scenario listed below.

1) As described in [3], a web developer can easily snoop the personalized ads displayed to a user using a little javascript code and store the ads for malicious use. A sample javascript code that can accomplish that task is –

```
(function () {
  function print_ads() {
    document.write( "<iframe src=url of ad server>" );
  }

  print_ads();
})()
```

When the victim visits the website and personalized ads are rendered to them, the javascript code snippet is stealthily made to run and the ads are noted alongside the associated username. This way, a website can indulge in mass surveillance and gather advertisements displayed to thousands of people. Most of the times, those advertisements will be product related which are customized based on the user's

interaction with E-Commerce giants like Amazon and Ebay. Hence, a target's product recommendations are successfully added to the attacker's knowledgebase.

2) Modern advertising agencies also make use of browsing patterns to advertise relevant items to their users. Cyber-attackers can easily track the websites frequented by a user by sending you an unscrupulous link which when clicked allows them to indulge in history sniffing [11]. An attacker should be able to replicate an advertising system which generates advertisements to a target using similar logic. If any of these ads correspond to items of the company we are targeting, the attacker can add them to their knowledge base.

3) Many third-party websites record the IP addresses of the users logging into their website alongside their credentials [6]. If the target uses their work computer to visit a website, and then takes this device to their home, the IP address used by the router at target's home is used to form a connection between all of the target's devices and the attacker now has access to this vital information like the number of target's devices and their types. If the same login credentials are being used by devices spread across various networks, then these devices are also connected to a target's profile and the browsing behavior displayed across all of these devices can act as auxiliary knowledge to the attacker [4].

4) In [5], we see how everything you ever say even anonymously can be traced back to you. A model can be made to learn your writing patterns and then can be used to classify newly seen snippets of text as belonging to you. If a target user has been leaving anonymous comments on any kind of posts, it is possible to associate them to that target and then sentiment analysis can be performed on them to see target's behavioral aspects and their inclinations. This information can act as implicit feedback to train the attack model through implicit feedback.

Victim Recommender System

We are targeting Item-Item collaborative filtering recommender system [1] in this report. Large E-commerce giants like Amazon use this type of recommender system because their database has a greater number of users than items. Other forms of recommender systems cannot scale to such a high volume of (user, item) interaction pairs. But an item-item collaborative filtering model overcomes this scale problem very efficiently. Items rated similarly by various users are similar to each other. That is the main principle behind this type of recommender system. We target item-item collaborative filtering recommender systems which are built using cosine similarity.

$$\mathrm{CS}(i,j) = \frac{\sum_{u \in U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U_i} r_{ui}^2} \sqrt{\sum_{u \in U_j} r_{uj}^2}}$$

where similarity between 2 items i and j is calculated using the above formula. 'U_i' refers to the set of users who rated item i and 'U_{ij}' refers to the set of users who have rated both item i and j.

Personalized Item Recommendations are generated in a ranked list for a user where the score of each item is calculated using -

$$y_{uj} = \frac{\sum_{i \in I_u} \mathrm{SIM}(i,j) r_{ui}}{\sum_{i \in I_u} \mathrm{SIM}(i,j)}$$

Where SIM(i, j) = CS(i,j) and 'r_{ui}' is the rating given by user 'u' to item 'i'.

Dataset Description

For our experiments, we used the Amazon Movies and TV Shows Dataset. We preprocessed the dataset to remove all users who had less than 20 items in their purchase history. After preprocessing, the dataset characteristics were as follows –
Number of records in the dataset: 1339355
Number of Unique Users in the dataset: 27747
Number of Unique Items in the dataset: 108613

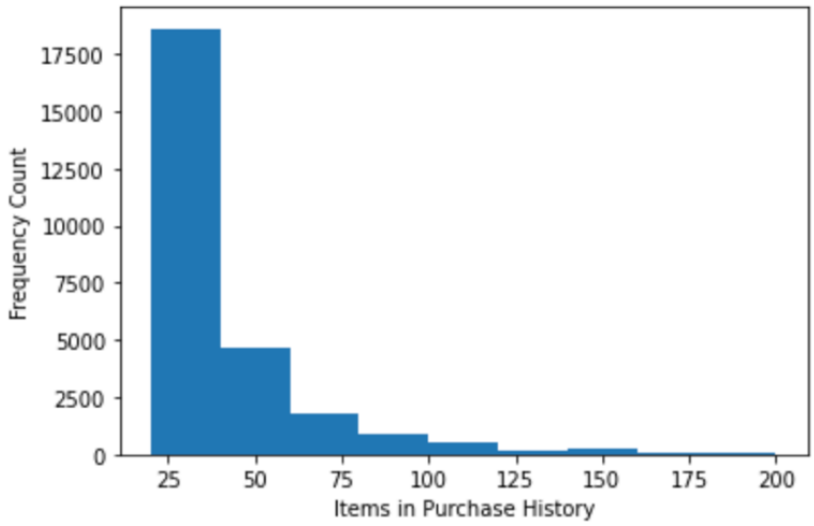


Fig 1(Items in Purchase History X Frequency Count)

The above figure shows that most of the users have between 25 to 40 items in their purchase history and the distribution follows a step-like decrease from there on.

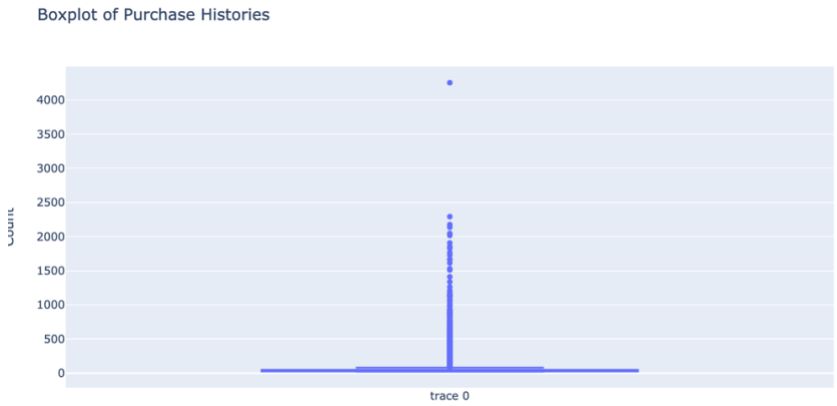


Fig 2

This boxplot shows that while 90% of the users have below 500 items in their purchase history, there are some users who are outliers with the items in their purchase histories being in the thousands with the highest outlier having 4024 items in their purchase history.

Metrics:

Precision: If 'pk' is a vector of the k items recommended to a user and 'a' is the set of items in original purchase history for that user. The "precision at k" for this user is defined as

$P(k)=|a \cap pk| / k$

In precision, we care that every move we make is a good move. For every item wrongly recommended, the precision is affected.

Recall: In Recall, we only care about the percentage of items correctly recommended to a user. The number of recommendations(k) can be infinite and still the recall won't be affected.

$R(k)=|a \cap pk| / |a|$

Hits: It is the number of items that could be successfully identified to be a part of the user's history.

$Hits(k) = |a \cap pk|$

Normalized Discounted Cumilative Gain (NDCG):

NDCG is a measure of ranking quality of a recommender. An ideal recommender would recommend most relevant items to a user at the top(with highest rank). Relevance of an item is the importance it holds to the user – in terms of how likely they are to purchase it, etc. In our attack model, as we already have the ratings of user's original purchase history, we treat the original rating of an item as its relevance.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

If an item with high relevance is recommended last, we penalize the recommender with respect to its index in the recommendation vector.

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

Every vector of recommendations is normalized by a Normalization Factor which is the ideal-NDCG (denominator). We push items with highest relevance at the top of the recommendations (ideal case) and calculate the DCG of this ideal recommendation to get IDCG.

RMSE (Root Mean Squared Error):

RMSE is a classic metric used in Regression Analysis. Suppose y and y^ are 2 vectors of length N where y contains the actual ratings and y^ contains the predicted ratings, then RMSE is calculated as –

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}.$$

Experiment Setup

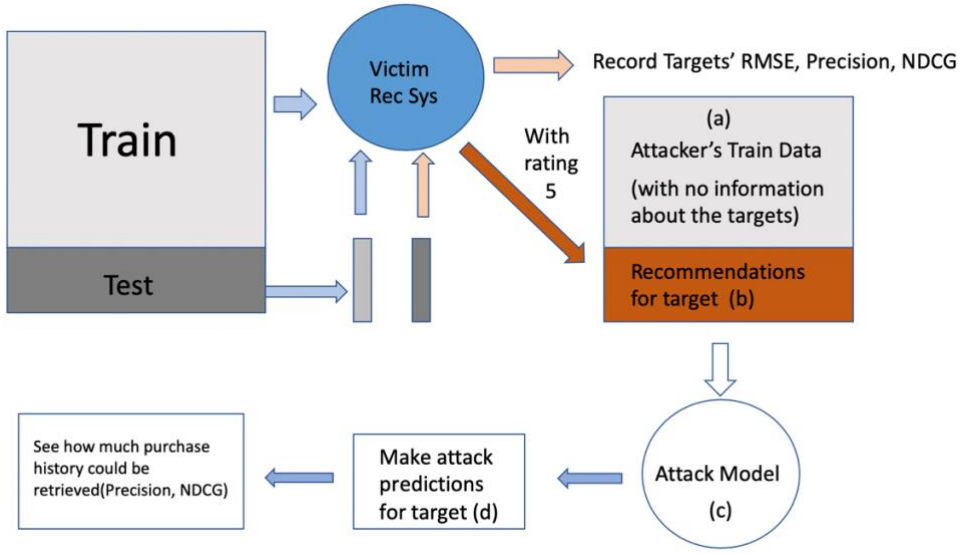


Fig 3: Project Workflow

Figure 3 broadly illustrates the approach we followed to answer a set of research questions which focus around maximizing an attacker's gain i.e. retrieval of target's purchase history –

RQ1) How much training data should the attacker have access to?
RQ2) How many users should the attacker target?
RQ3) How many recommendations for the targets should the attacker observe?
RQ4) How many attack predictions should the attacker make for a target?
RQ5) Which attack model is more efficient?

Approach to the Problem Statement

We are going to split all the users in the dataset into 99% train and 1% test. The entire purchase histories of all train users and a small portion of test users' purchase histories is used to train an item-item collaborative filtering model. The rest of the portion of test users is used to record the efficiency of this soon-to-be victim recommender system on each test user by using the RMSE metric.

This victim recommender system is made to recommend a few items to each test user. These recommendations are equivalent to what our attacker can gain on a target using any of the attack scenarios discussed earlier. Using these item recommendations for the target and train data available for other users via the publicly available datasets, the attacker is going to create an attack model and try to predict/recommend the purchase history of the target. This is a novel approach which was not tried out by anyone earlier.

To answer the above questions, the 5 variables were given the following values –
1) Train Data Percentages – [1, 0.8, 0.6, 0.4, 0.2]
2) Targets – [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3) Observed Recommendations – [5, 10, 15, 20, 30, 45, 60, 90, 120, 150, 200, 300]
4) Attack Recommendations – [5, 10, 15, 20, 30, 45, 60, 90, 120, 150, 200, 300]
5) Models – [Item-Item collaborative filtering, Ranking Matrix Factorization]

Attack Model

Experiments were performed by using two types of attack models –
1) Item-Item Collaborative Filtering
2) Ranking Factorization Machine

Item-Item Collaborative Filtering

As the victim recommender system, we are targeting is built using item-item collaborative filtering [1] which utilizes cosine similarity, our first attack model is also a recommender system which follows the same principle [1]. This way, we can exploit the similarity between purchase histories and recommended items to the maximum extent.

Ranking Factorization Machine

A Ranking Factorization Recommender [2] learns latent factors for each user and item and uses them to rank recommended items according to the likelihood of observing those (user, item) pairs. This is commonly desired when performing collaborative filtering for implicit feedback datasets [9] or datasets with explicit ratings for which ranking prediction is desired.

The (user, item) pair is scored by using the following function

$$\text{score}(i,j) = \mu + w_i + w_j + \mathbf{a}^T\mathbf{x}_i + \mathbf{b}^T\mathbf{y}_j + \mathbf{u}_i^T\mathbf{v}_j$$

where μ is a global bias term, w_i is the weight term for user i, w_j is the weight term for item j, $\mathbf{x}_i$ and $\mathbf{y}_j$ are respectively the user and item side feature vectors, and **a** and **b** are respectively the weight vectors for those side features. The user latent factors are given by $\mathbf{u}_i$ and the item latent factors are denoted by $\mathbf{v}_j$.

Training the model:
Stochastic Gradient Descent is used in training the model by trying to optimize the following objective function

$$\min_{\mathbf{w},\mathbf{a},\mathbf{b},\mathbf{V},\mathbf{U}} \frac{1}{|\mathcal{D}|} \sum_{(i,j,r_{ij})\in\mathcal{D}} \mathcal{L}(\text{score}(i,j), r_{ij}) + \lambda_1(\|\mathbf{w}\|_2^2 + ||\mathbf{a}||_2^2 + ||\mathbf{b}||_2^2) + \lambda_2 \left(\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2\right)$$

where D is the observation dataset, r_{ij} is the rating that user i gave to item j, $\mathbf{U}=(\mathbf{u}_1, \mathbf{u}_2,...)$ denotes the user's latent factors and $V=(v_1,v_2,...)$ denotes the item latent factors. The loss function $L(\hat{y}, y)$ is $(\hat{y} - y)_2$ by default. $\lambda 1$ denotes the linear regularization parameter and $\lambda 2$ is the regularization parameter. When the algorithm is run on a dataset, stochastic gradient descent is used to adjust the regularizing parameters and user, item latent vectors to minimize the loss function.

Algorithm1 – Retrieving Items in Purchase History

```
for t in targets:
  users_set <- select_10_pairs_of_t_users
  for user in users_set:
    user_items = get_user_items(dataset)
    for percentage in train_data_percentages:
      if percentage == 1:
          num_of_folds = 1
      else:
          num_of_folds = 5
      while num_of_folds > 0:
        num_of_folds <- num_of_folds - 1
        attacker_data <- percentage * train_dataset
        for k_o in observed_recommendations:
          recs <- victim_model.recommend(k_o)
          attacker_data.append(recs)
          for attack_model in models:
            attack_model <- attack_model.create(attacker_data)
            for k_a in attack_predictions:
              pred_items = attack_model.recommend(k_a)
              precision, recall, ndcg = get_metrics(user_items, pred_items)
              victim_model_rmse <- victim_model.evaluate(user)
              make_log(t, attacker_data, user, len(user_items), k_o, k_a, precision,
                      ndcg, recall, victim_model_rmse)
```

Every recommendation for a target observed by an attacker is added to their knowledge base with rating 5. When the attacker is given less than 100% of training data, 5-fold cross validation is performed to remove any bias because of a special portion of a dataset selected. The above algorithm was run once with item-item

collaborative filtering as the attack model and then again with Ranking Matrix Factorization as the attack model. So different users were targeted by both the models.

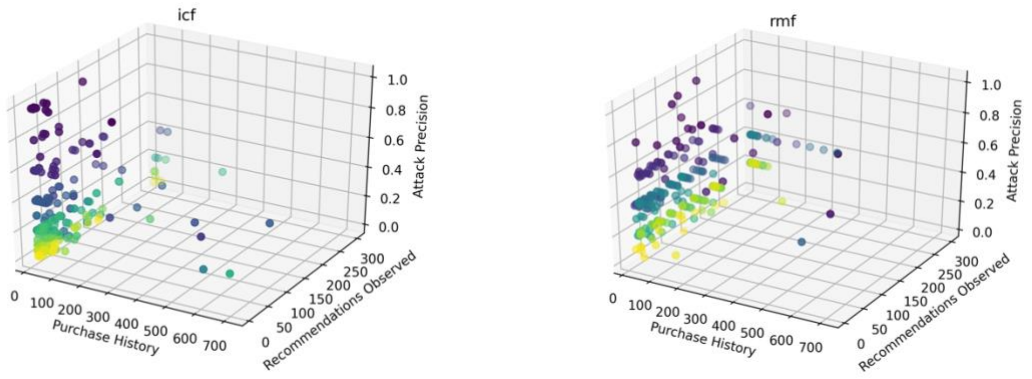
Results

The collected data helped us to estimate a few useful parameters for the attacker.

Each of the dots in the upcoming 3D graphs denotes a unique user. Item-Item collaborative filtering will be denoted by ICF and Ranking Matrix Factorization will be denoted by RMF.

Items in Purchase History

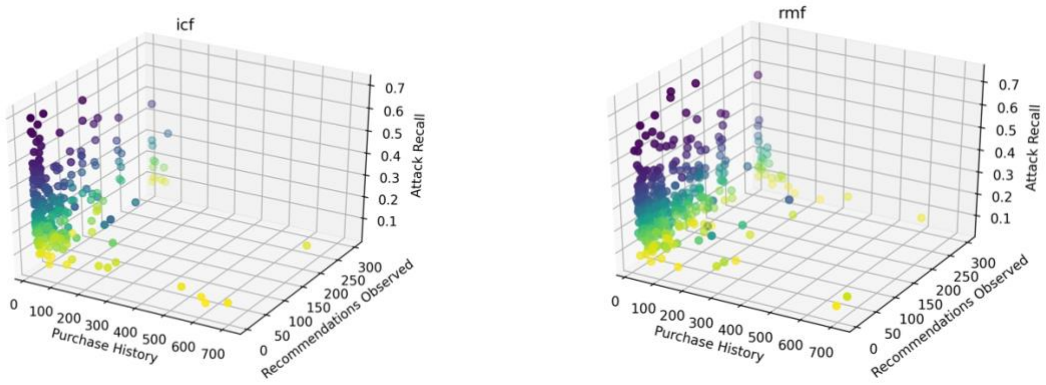
(Purchase History X Recommendations Observed X Precision) -



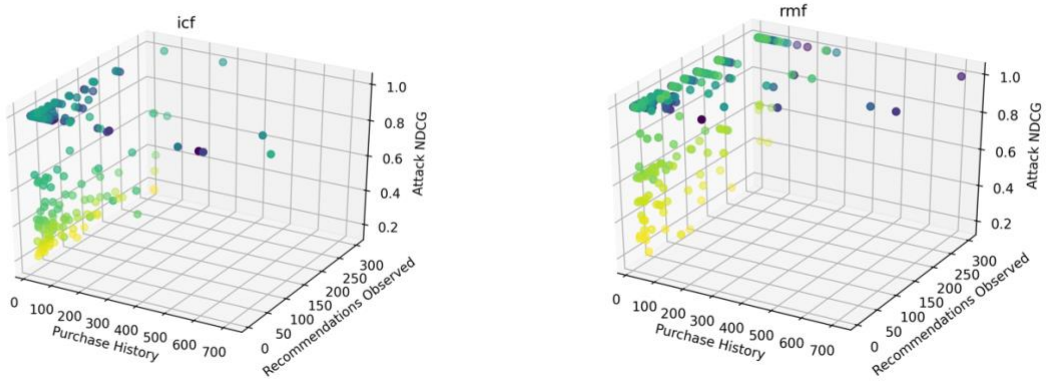
Both the algorithms performed the most efficiently when the targets had fewer number of items in their purchase history and when fewer recommendations were observed for them. According to intuition, observing as many recommendations as possible for a target should benefit the attacker, as it will broaden their understanding of the target's inclinations. The Victim Recommender System generates a ranked list of items for each user with items appearing earlier on being the most relevant and the items appearing later on being the least relevant. As we added all the recommendations with equal importance (rating = 5) to the attacker's knowledge base and as these algorithms are trained on explicit feedback (ratings metric), this generated noise for the attack-models and their performance deteriorated as more recommendations were added. The algorithms best understood

a user's preferences only by looking at the top 50 – 150 recommendations made to them. This speaks to the claim that recommender systems have an ideal spot for information gathering. Getting too much information beyond this point can be detrimental.

(Purchase History X Recommendations Observed X Recall) –

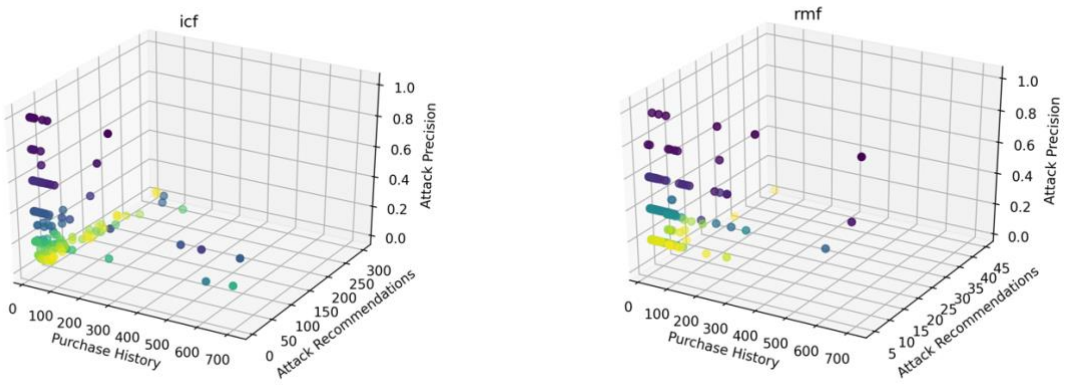


(Purchase History X Recommendations Observed X NDCG) –



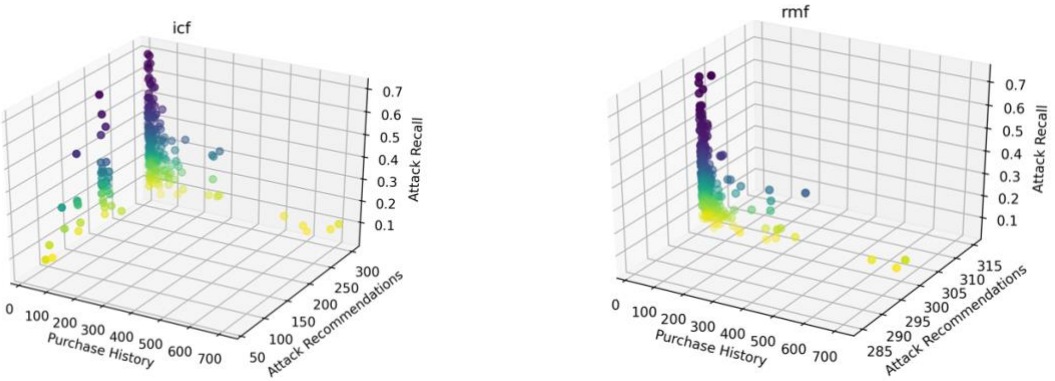
Users who experienced at least a little precision in these 2-variable correlations, can be seen to have received much higher Recall and NDCG.

(Purchase History X Attack Recommendations X Precision) –



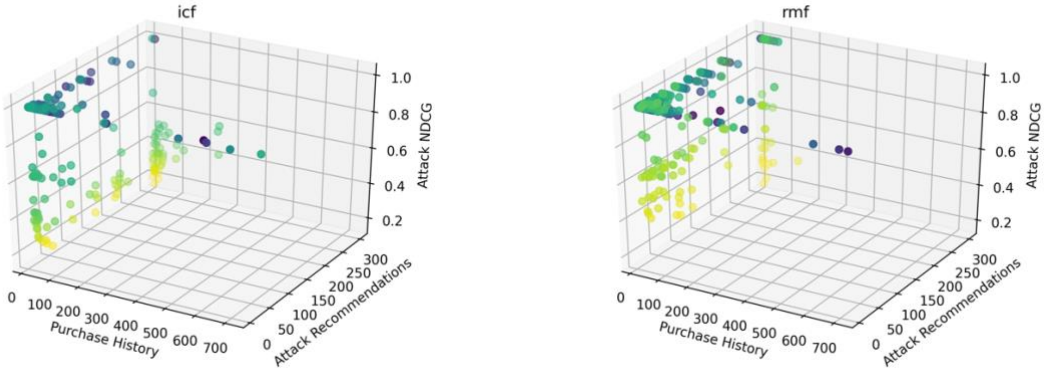
Users with few items in their purchase history can be easily targeted. Precision for them is greater when the number of attack recommendations is less, as a greater number of incorrect attacks/guesses will only negatively affect the precision.

Recall (Purchase History X Attack Recommendations) -



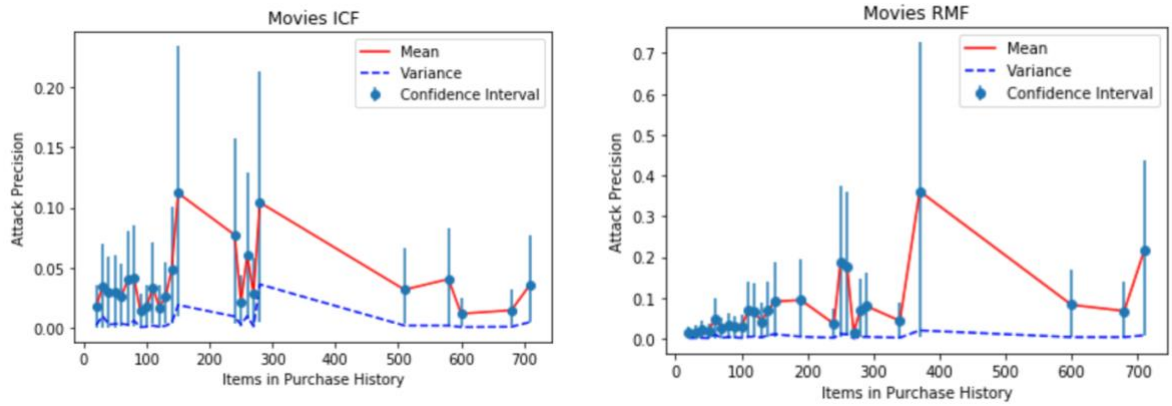
For recall, some users who were attacked by ICF, maxed out on item-hits after the attack model was only half-way through its mission (150 attack recommendations). Keeping on attacking them beyond this point had no effect. Most of the ICF users acquired maximum recall at maximum attack recommendations (300) but ALL of the RMF users acquired maximum recall at 300 attack recommendations.

(Purchase History X Attack Recommendations X NDCG) –



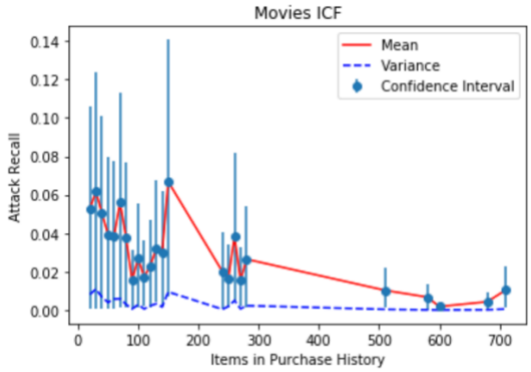
Every user who got even a little precision, received good NDCG and some users who didn't get any precision received some NDCG as the number of attack recommendations grew. This is because precision was penalized by the number of attack recommendations but NDCG can be achieved even if an item was retrieved at the last attack recommendation.

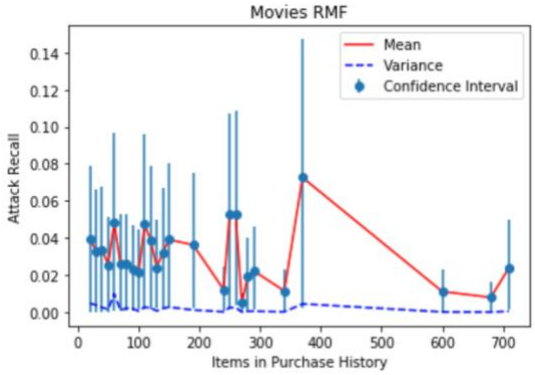
(Purchase History X Attack Precision)



There were abundent number of targets with 0-100 items in their purchase history. That is why the mean of that group is about 2 – 4% but the confidence interval is very thin which suggests that the mean is accurate. Although both the graphs represent high mean precision for users who had lots of items in their purchase history, the confidence interval tells us that their mean got affected by a few outliers.

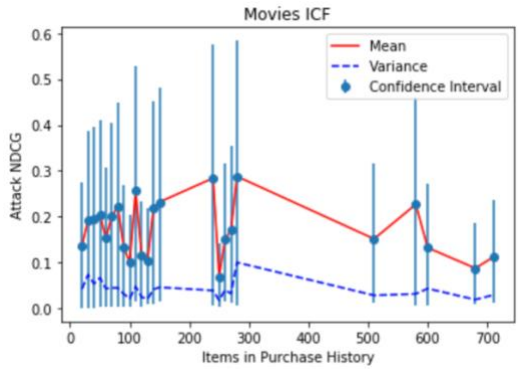
(Purchase History X Attack Recall)

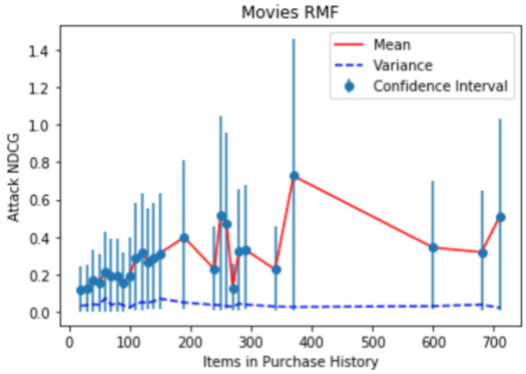




We can surely say that ICF received high Recall for targets with 200-300 items in their purchase history as the Confidence Interval is low for that group and RMF received high recall for targets who had less than 100 items in their purchase histories.

(Purchase History X Attack NDCG)

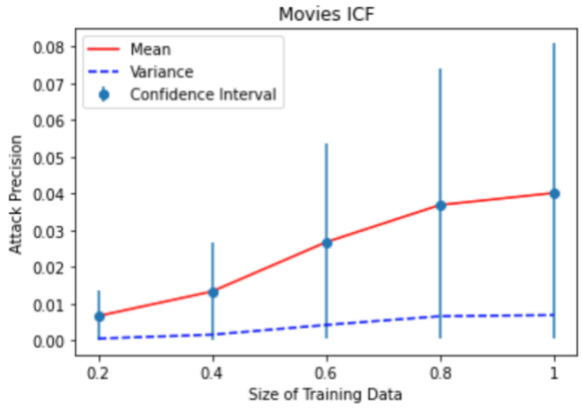


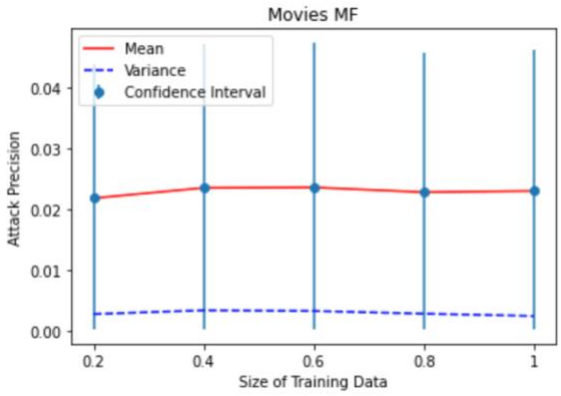


RMF had a verifiable decent NDCG performance on users with less items in their purchase history.

Training Data

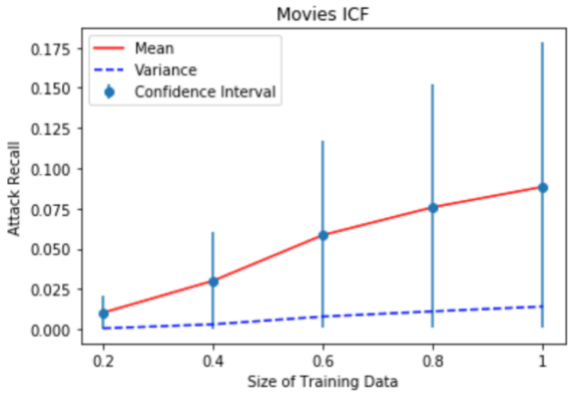
Precision -

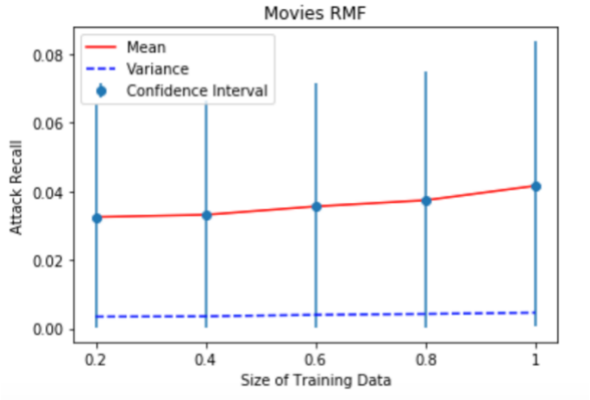




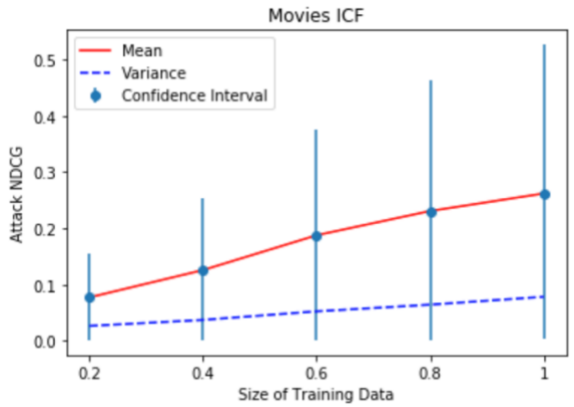
ICF suffered a lot in the absence of training data. But it also received the highest average precision of 4% when entire training data was present for the attacker. RMF although did not fluctuate much depending on the amount of training data available, its mean precision for any amount of training data did not exceed 2.2%. Recall and NDCG followed a similar trend.

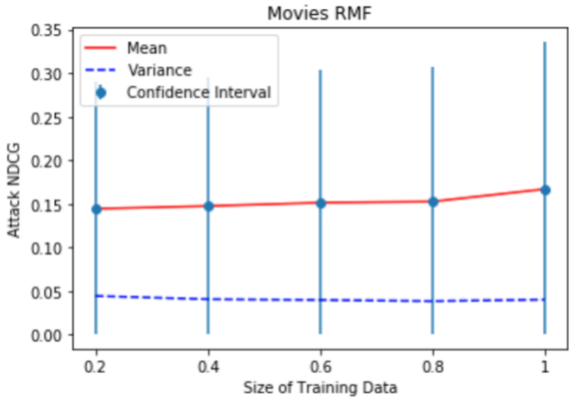
Recall -





NDCG -

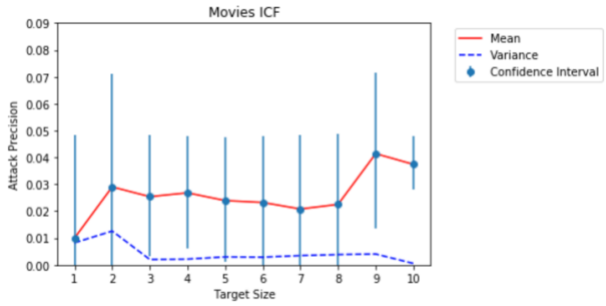


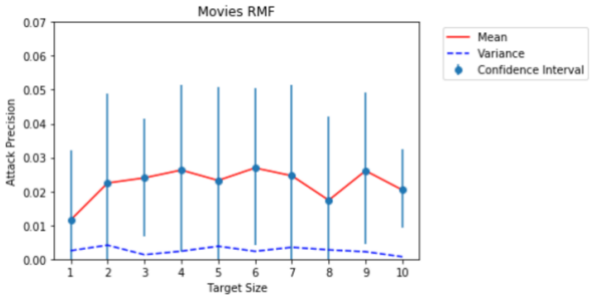


Target Size

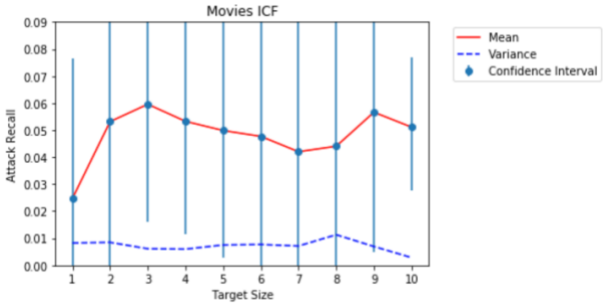
In these graphs, we plot the target sizes 1 to 10 against their mean group metrics.

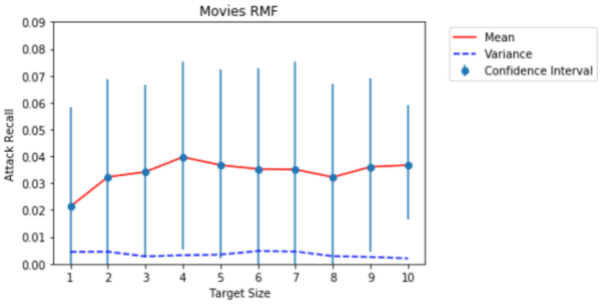
Precision -



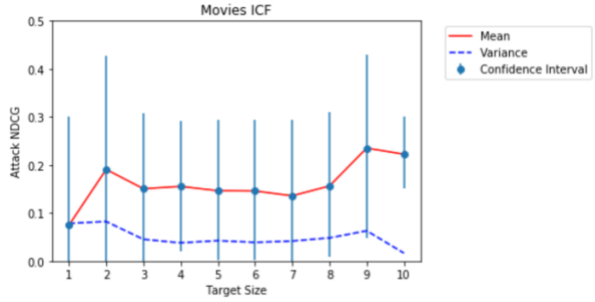


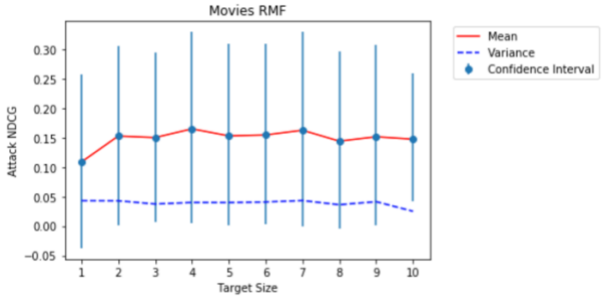
Recall -





NDCG -

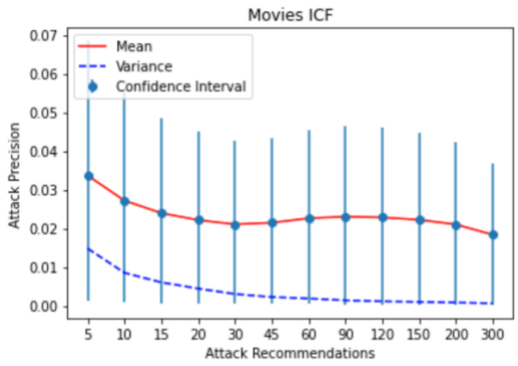


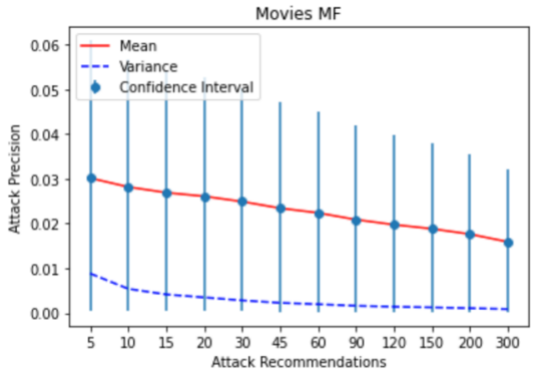


It is unclear to exactly estimate targeting how many users would be the best for an attacker given choices from 1 target to 10 targets, as the confidence intervals for each group spans almost the same interval for all.

Attack Recommendations

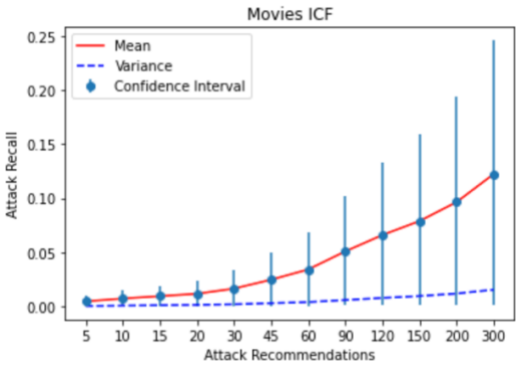
Precision –

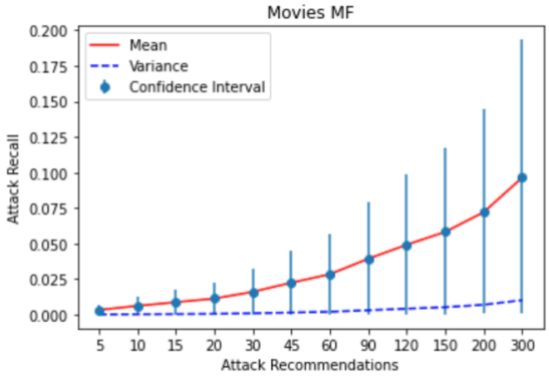




As the number of attack recommendations kept on increasing, both the algorithms saw a decline in precision as most of the attack recommendations that came later on, did not hit any of the target's original items. This penalizes the precision metric.

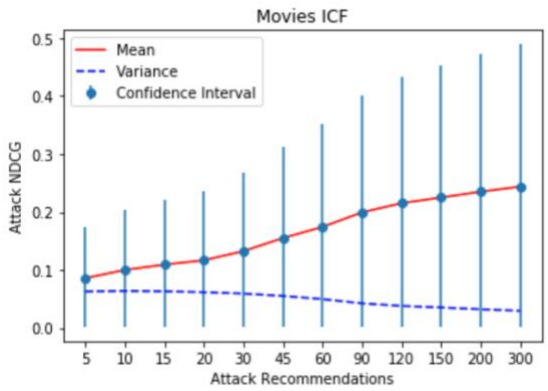
Recall –

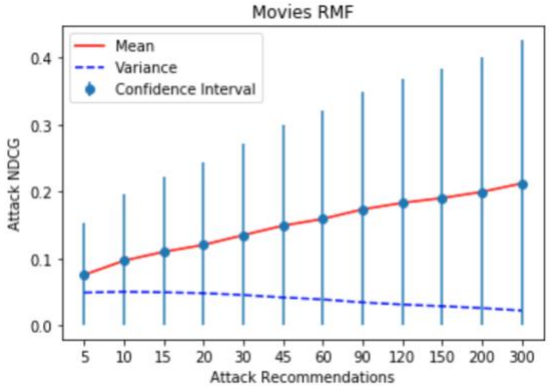




Recall is not penalized when an attack recommendation is not present in target's purchase history. So as the number of chances to guess an item for an attack model increases, the recall also increases proportionally.

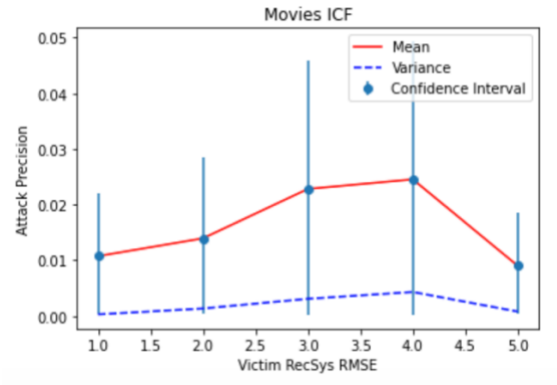
NDCG -

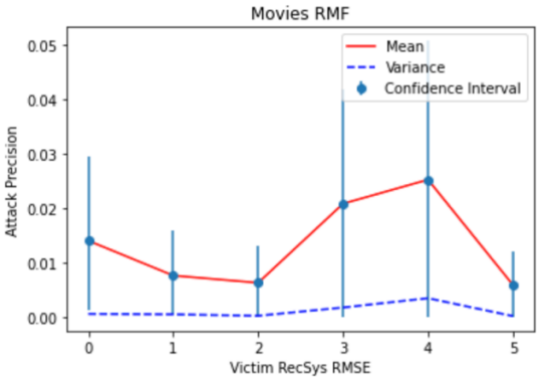




Both the algorithms have almost the same NDCG and it increases as attack recommendations increase and the probability of hitting an item in target's purchase history increases.

(Victim Model RMSE X Precision)





The floor of the Victim Model RMSE of target users is plotted against the precision recorded on them. Targets who had observed a RMSE between 0 and 2 received a very reliable mean precision of 1%. Targets with RMSE between 3 and 4 had a few outliers who influenced the mean precision of their groups in the above graphs.

The metrics obtained on all the experiments performed are as follows -

ICF	Precision	Recall	NDCG
Mean	0.0232	0.043	0.150
Variance	0.0038	0.0066	0.0486
Confidence Interval	(0.0231, 0.0234)	(0.0433, 0.0437)	(0.0149, 0.151)

RMF	Precision	Recall	NDCG
Mean	0.0229	0.034	0.149
Variance	0.0029	0.0037	0.040
Confidence Interval	(0.0228, 0.0230)	(0.0341, 0.034)	(0.149, 0.15)

According to these tables, ICF clearly had a better Recall than RMF but they performed quite similarly on Precision and NDCG with ICF taking a little edge over RMF.

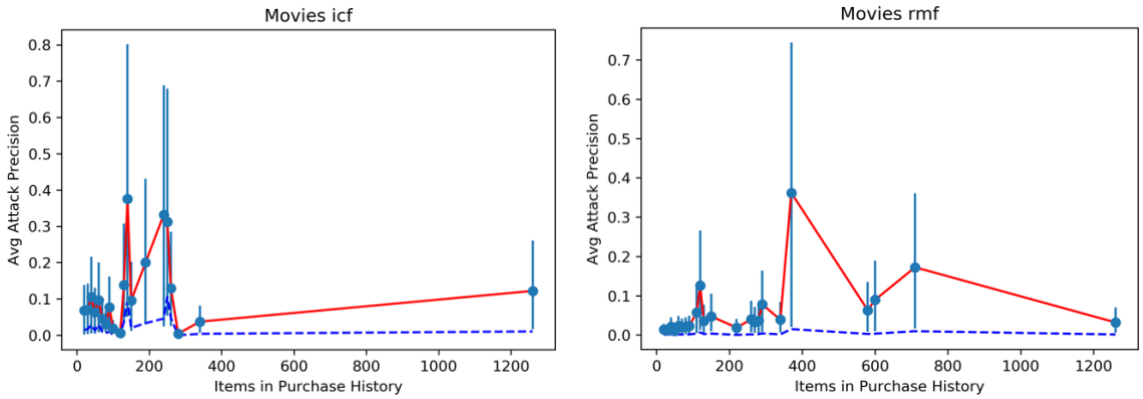
As Model Inversion is a NP-Hard Problem and there is no deterministic algorithm to accurately verify it, we can only run experiments and try to estimate the solution. To check if the solution obtained so far is valid, we again ran a set of experiments to see if the same level of results is achieved. Also, as retrieving items in purchase history is proving to be a herculean task, in these new set of experiments, after every recommendation is observed for a target, we calculate RMSE to see if the attacker can at least infer the target's ratings on their purchase history items. We also stopped fluctuating the amount of training data available to the attacker and estimated the rest of the variables considering the scenario where the attacker has access to 100% training data. Again, different sets of users were targeted by ICF and RMF.

Algorithm2: Retrieve Items + Record RMSE

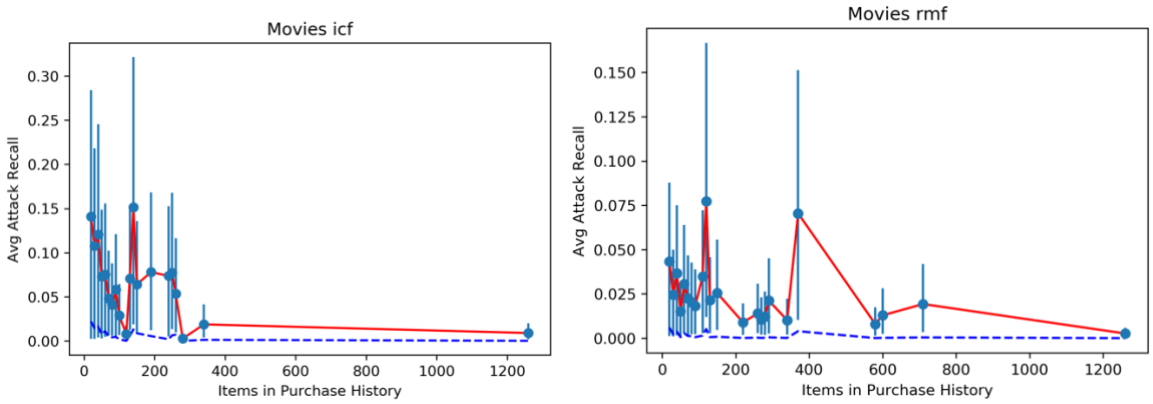
```
for t in targets:
  users_set <- select_10_pairs_of_t_users
  for user in users_set:
    user_items = get_user_items(dataset)
    attacker_data <- train_dataset
    for k_o in observed_recommendations:
      recs <- victim_model.recommend(k_o)
      attacker_data.append(recs)
      for attack_model in models:
        attack_model <- attack_model.create(attacker_data)
        for k_a in attack_predictions:
          pred_items = attack_model.recommend(k_a)
          precision, recall, ndcg, rmse = get_metrics(user_items, pred_items)
          victim_model_rmse <- victim_model.evaluate(user)
          make_log(t, attacker_data, user, len(user_items), k_o, k_a, precision,
                  ndcg, recall, victim_model_rmse, rmse)
```

Purchase History

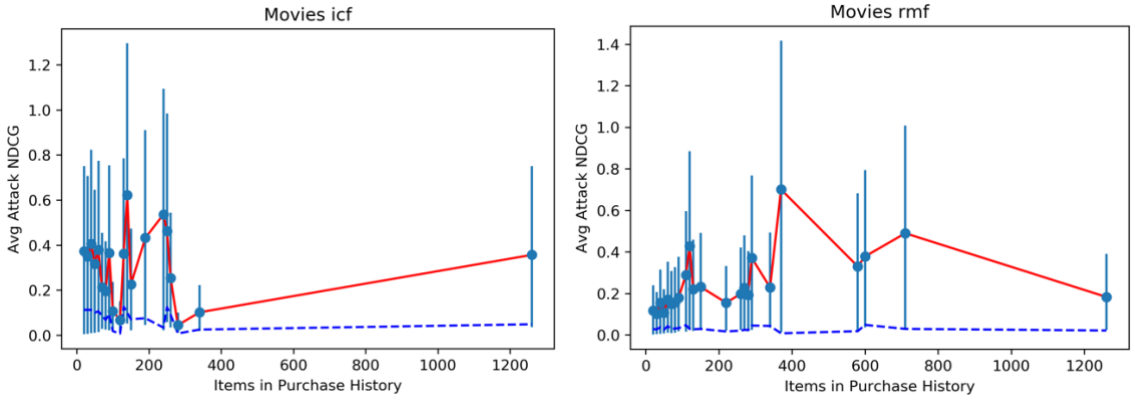
Precision –



Recall -



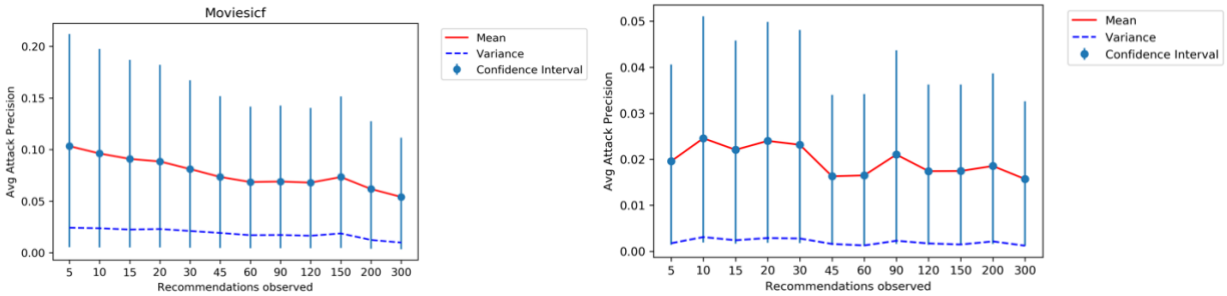
NDCG -



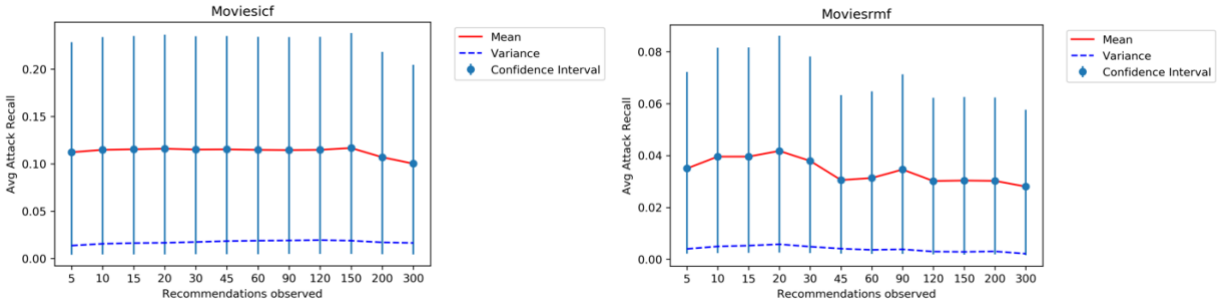
Again, we observe that targets with low items in their purchase history can be easily exploited. It is because the recommendations generated by these fewer purchase history items fall very close to the original items in the Cartesian Space and hence they are very helpful in retrieving back the original items. If the number of items in purchase history is very high, the recommendations generated can be very diverse and this increases the search space for the attacker to navigate and retrieve the original items.

Recommendations Observed

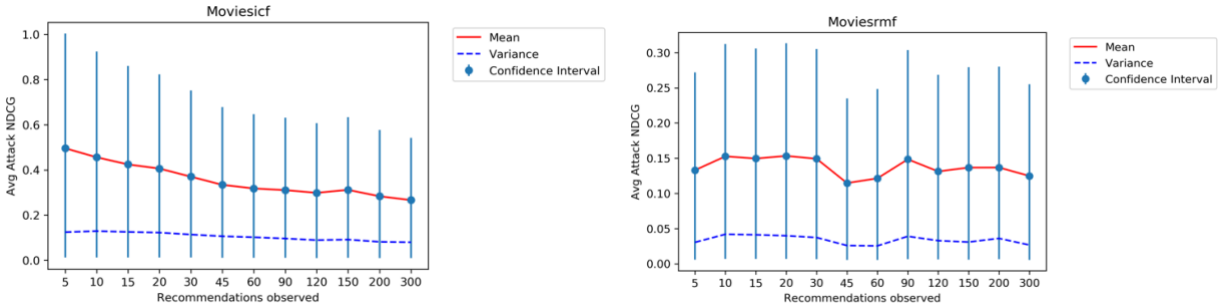
Precision -



Recall -



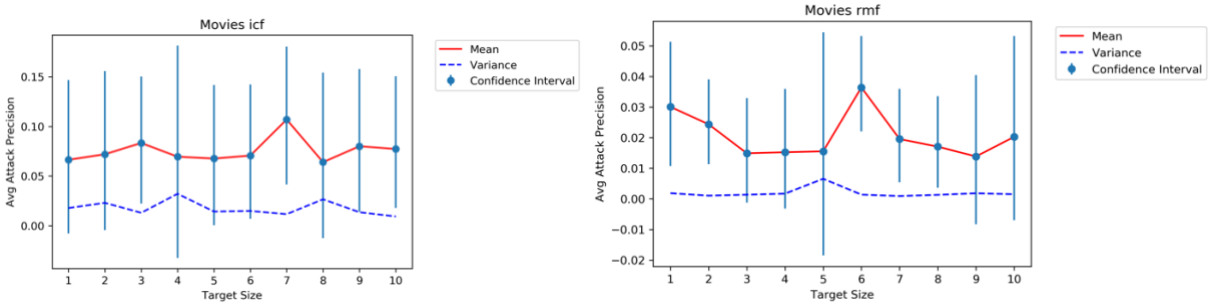
NDCG-



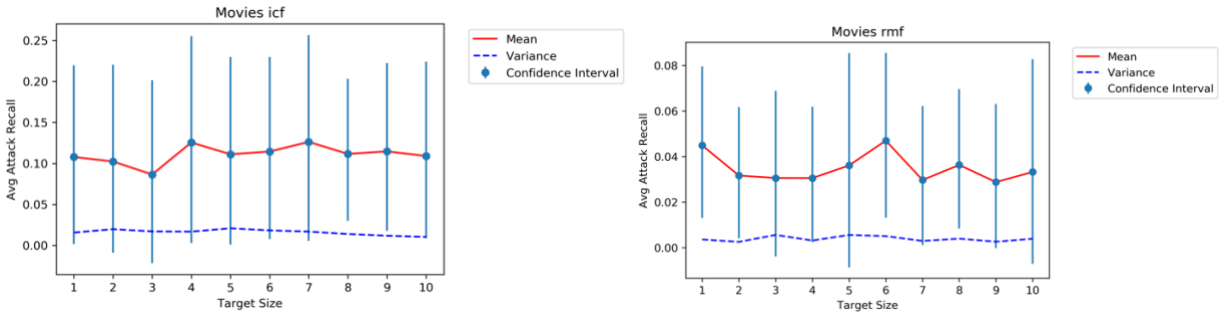
We again observe that both the algorithms display a property of “ideal amount of data to be observed”. Only most relevant recommendations aid the attacker. Hence, we must model an attacker with the capability of ranking all the recommendations they have observed by using each target’s auxiliary information etc.

Target Size

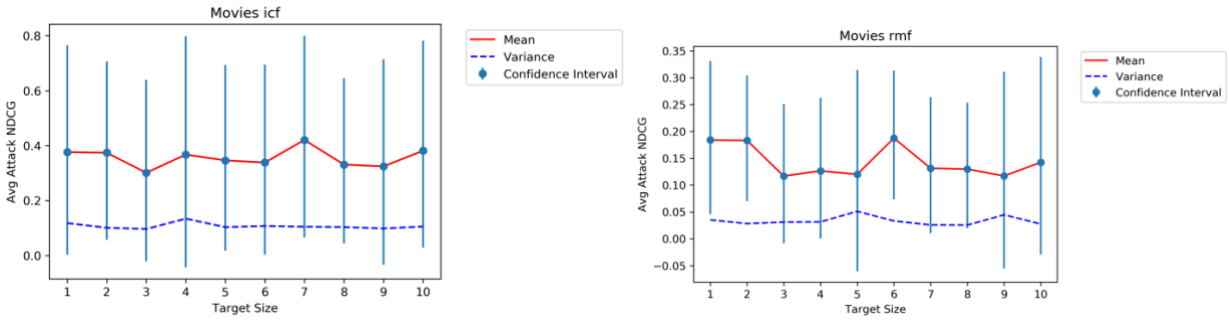
Precision-



Recall-



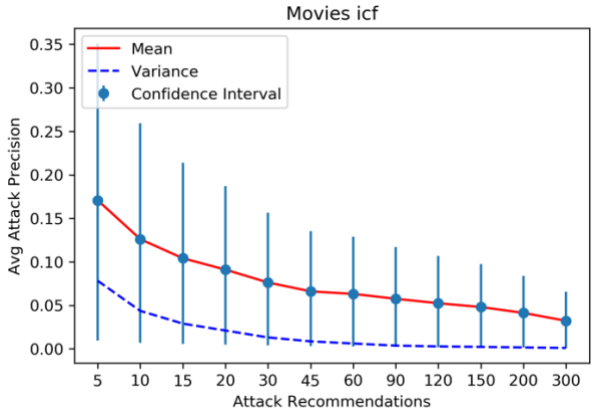
NDCG-

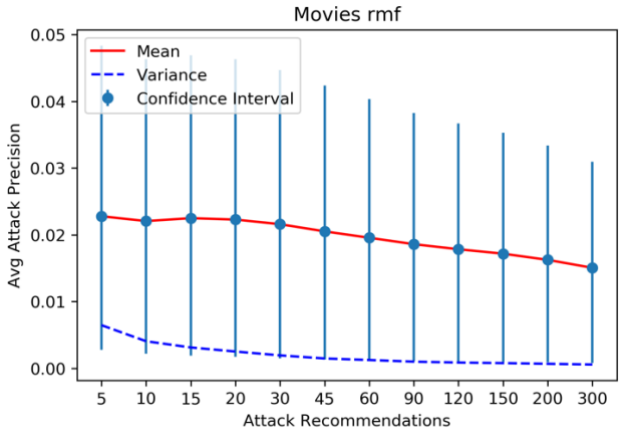


As the number of targets linearly increases, all 3 metrics tend to follow a zigzag-like increase and decrease. In the current data, the 3 metrics seem to peak when 7 or 8 users are selected as targets at once.

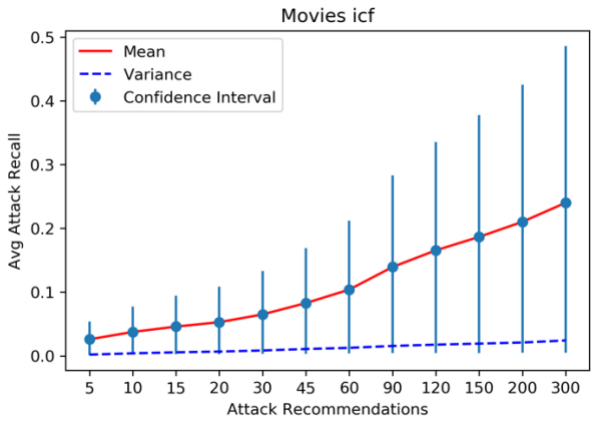
Attack Recommendations

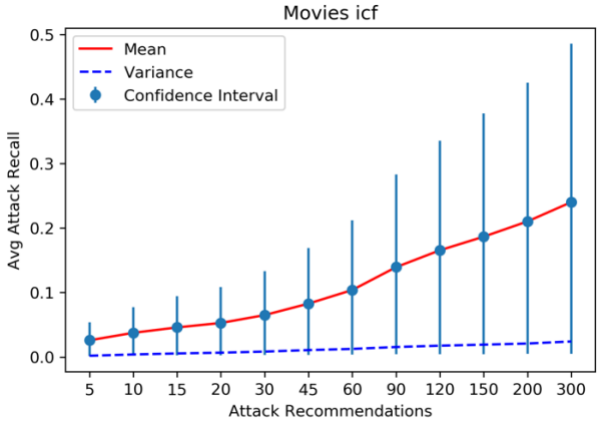
Precision-



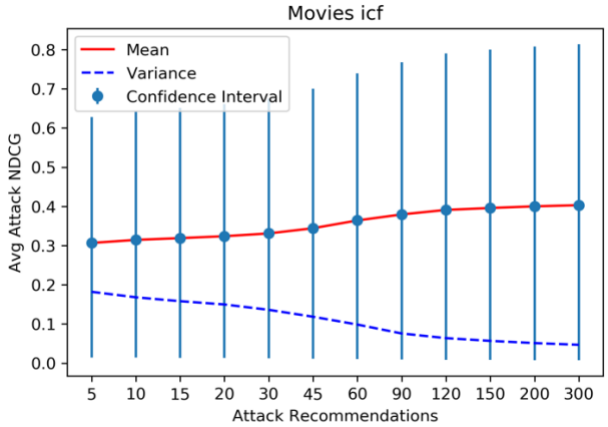


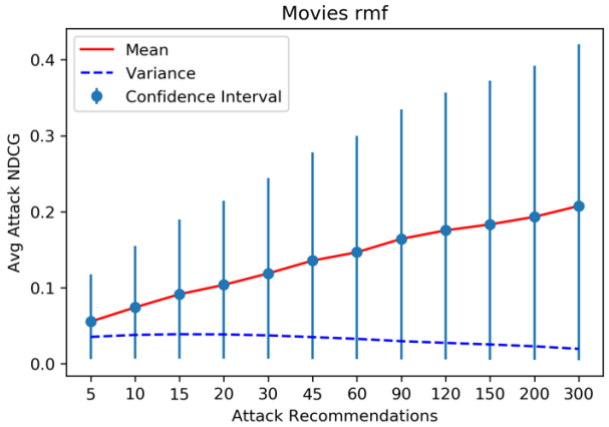
Recall-





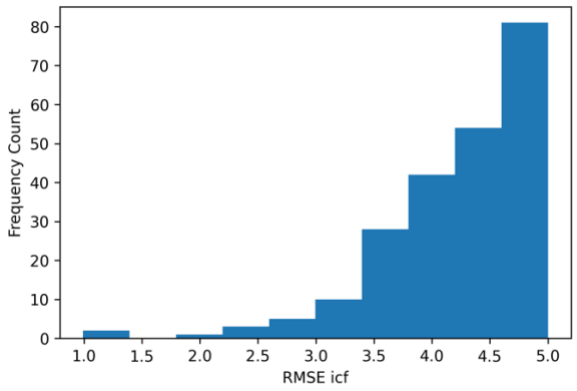
NDCG-





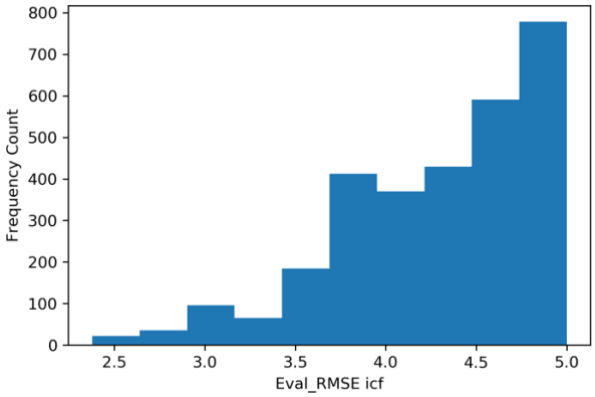
RMSE Analysis

Distribution of RMSE in ICF-Victim RecSys



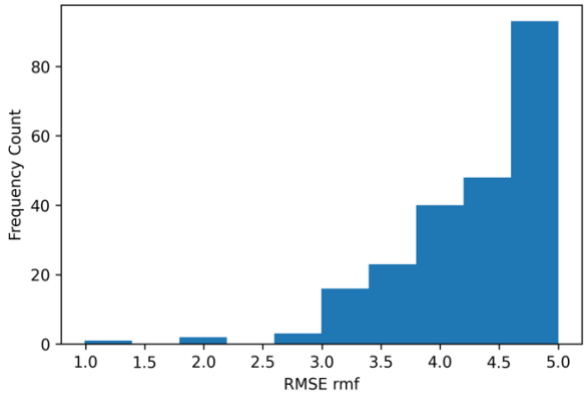
Victim RecSys User RMSE Avg: 4.26

Distribution of ICF Attack Model RMSE



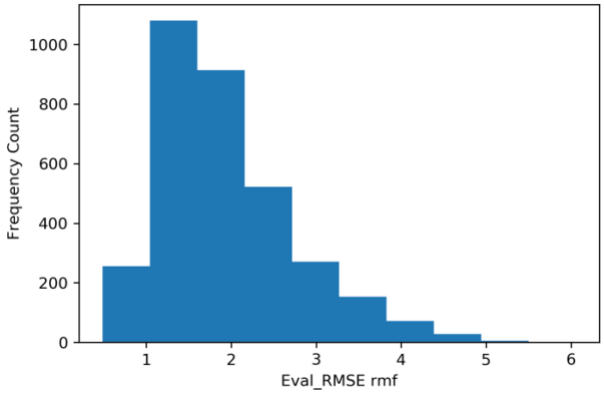
Attack Model User RMSE Avg: 4.29

Distribution of RMSE in RMF-Victim RecSys



Victim RecSys User RMSE Avg: 4.30

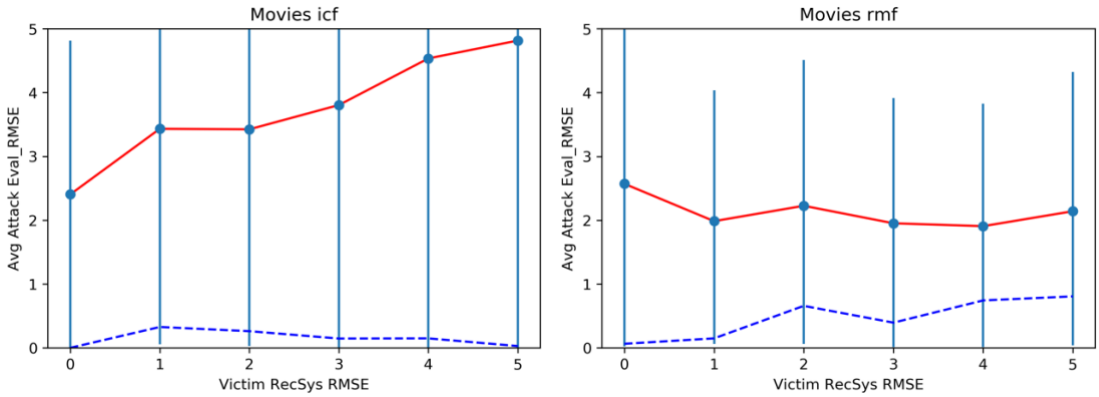
Distribution of RMF Model RMSE



Attack Model User RMSE Avg: 1.94

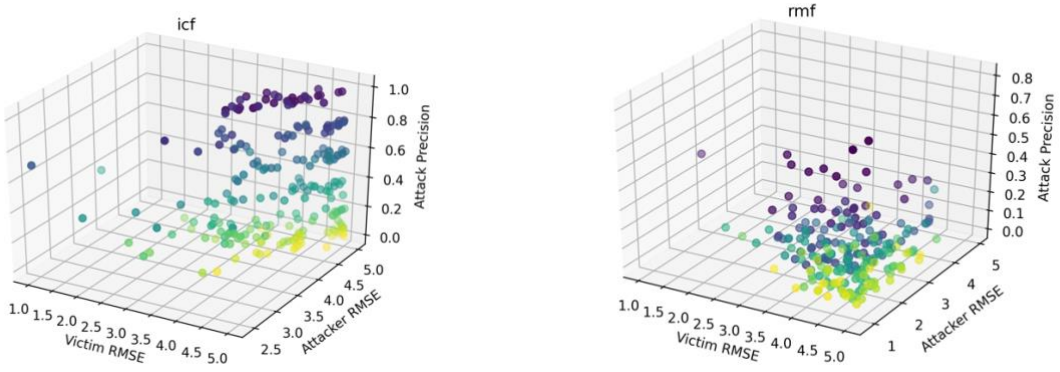
The histograms denote the distribution of Target User RMSEs by the victim model and the attack model. The Victim Recommender System which is a cosine-similarity based item-item collaborative filtering model achieved an Average User RMSE of 4.3 when being targeted by our 2 attack models. The attack model which is also a cosine-based similarity recommender also observed a similar Average User RMSE of 4.3. But the Ranking Matrix Factorization attack model recorded an impressive 1.94 RMSE. This speaks to RMF's superior capability of predicting rating for an unknown (user, item) pair.

(Victim Model RMSE X Attack Model RMSE)



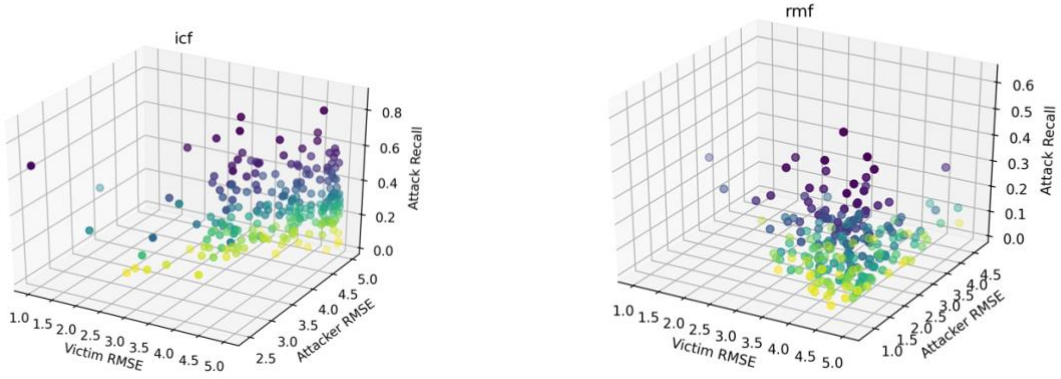
As the ICF Victim Model RMSE increased, the ICF attack model RMSE also increased. But RMF attack model always reported a decent RMSE irrespective of victim model's RMSE.

(Victim Model RMSE X Attacker Model RMSE X Precision) -



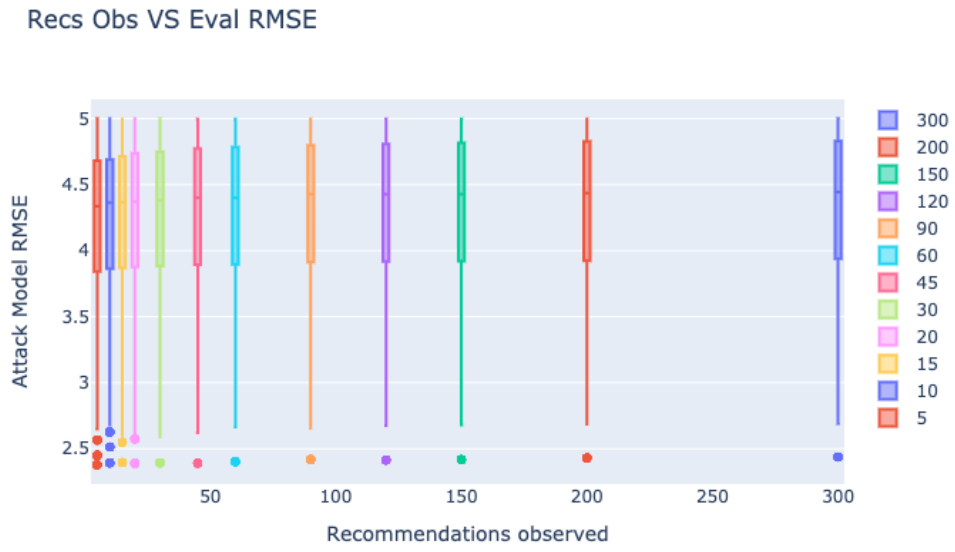
These graphs tell us that the victim model reported an RMSE of 3 or higher for the targets, but the recommendations generated by it could be used by both the attack models to retrieve few of their items. In ICF attack model, even though the model reported a high RMSE for target users than RMF, it performed better in item-retrieval. But RMF reported better RMSE for the users it targeted.

(Victim Model RMSE X Attacker Model RMSE X Recall) -



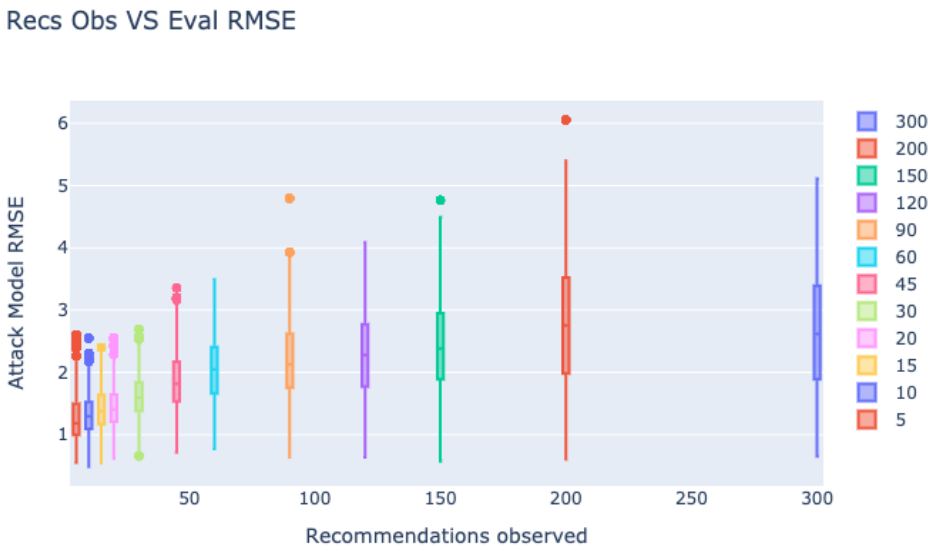
ICF again reported worse RMSE but better Recall than RMF which shined in reporting less RMSE.

ICF Boxplot



The ICF attack model mostly recorded an RMSE in the range of the 3.7 to 4.7 for any number of item recommendations it observed. But there were a few outliers for whom it reported a much better RMSE of 2.5 to 3.0.

RMF Boxplot



The RMF reported very good RMSE for the target users. But as more and more recommendations were being observed, the RMSE range for the targets also increased.

ICF	Precision	Recall	NDCG	RMSE
Mean	0.0773	0.113	0.356	4.29
Variance	0.0189	0.0172	0.109	0.307
Confidence Interval	(0.0760, 0.0787)	(0.111, 0.114)	(0.352, 0.359)	(4.284, 4.295)

RMF	Precision	Recall	NDCG	RMSE
Mean	0.019	0.034	0.137	1.94
Variance	0.002	0.0039	0.034	0.649
Confidence Interval	(0.019, 0.020)	(0.0334, 0.0347)	(0.135, 0.139)	(1.93, 1.95)

ICF outperformed RMF in all 3 item-retrieval metrics in these new experiments but RMF had a better RMSE track record.

Conclusion

Both the algorithms could not succeed to effectively retrieve purchase history when a user is targeted at random. Any success witnessed for a few users can be attributed to it being an outlier. But when the attack models make an attack recommendation and it is not part of the target's purchase history, it is not necessarily an entire failure. The attack models might have correctly predicted the target's future purchases.

The analysis performed also helps us making these key observations –
1) If 100% training data is present, Item-Item collaborative filtering should be employed, otherwise Ranking Matrix Factorization is suitable.

2) There is no optimal number of users to target. Targeting any number of users generated the same mean precision.

3) Targeting users with fewer number of items in their purchase history yields more benefit as the recommendations generated for them would be closely tied to the purchase history items.

4) Utilizing only relevant recommendations for a target is beneficial. Using all the recommendations observed can be counterproductive.

5) To preserve the attack precision, the attack models should stop attacking after 100 attack recommendations.

6) Item-Item collaborative proved to rank its recommendations more efficiently, but Ranking Matrix Factorization was better able to predict item ratings for a user. Hence an optimum attack model would be one which uses ICF to generate recommendations and add the top k recommendations with rating predicted by RMF.

Future Work

- It would be beneficial for the attacker to observe some auxiliary information on a target like name, location etc. as both the algorithms can train on implicit feedback. Other auxiliary information like target's preferences, personality traits and recent activity can also help the attacker in estimating the rating prediction for a (user, item) pair.
- For every item correctly identified on a target, the attack model should retrain itself immediately to improve its performance on the target.
- Having target's partial purchase history can be helpful in filling the other missing data values.

References

[1] Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1. 10.1145/371920.372071.

[2] Koren, Yehuda & Bell, Robert & Volinsky, Chris. (2009). Matrix factorization techniques for recommender systems. Computer. 42. 30-37.

[3] Gandhi, Mona & Jakobsson, Markus & Ratkiewicz, Jacob. (2006). Badvertisements: Stealthy Click-Fraud with Unwitting Accessories. J. Digital Forensic Practice. 1. 131-142. 10.1080/15567280601015598.1

[4] Arora, Nitin & Peri, Sagar. (2017). Browsing Pattern Analysis: What user browsing Patterns Indicate.

[5] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You are what you say: privacy risks of public mentions. In SIGIR, 2006.

[6] Brookman, Justin & Rouge, Phoebe & Alva, Aaron & Yeung, Christina. (2017). Cross-Device Tracking: Measurement and Disclosures. Proceedings on Privacy Enhancing Technologies. 2017. 10.1515/popets-2017-0020.

[7] Calandrino, Joseph & Kilzer, Ann & Narayanan, Arvind & Felten, Edward & Shmatikov, Vitaly. (2011). You Might Also Like: Privacy Risks of Collaborative Filtering. Proceedings - IEEE Symposium on Security and Privacy. 231 - 246. 10.1109/SP.2011.40.

[8] Jonasson, Daniel & Sigholm, Johan. (2020). What is Spyware?.

[9] Hu, Yifan & Koren, Yehuda & Volinsky, Chris. (2008). Collaborative Filtering for Implicit Feedback Datasets. Proceedings - IEEE International Conference on Data Mining, ICDM. 263-272. 10.1109/ICDM.2008.22.

[10] Shokri, Reza, et al. “Membership Inference Attacks Against Machine Learning Models.” 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 3–18. 2017 IEEE Symposium on Security and Privacy (SP), doi:10.1109/SP.2017.41.

[11] Jakobsson, Markus & Stamm, Sid. (2006). Invasive browser sniffing and countermeasures. 523-532. 10.1145/1135777.1135854.