Chiron: A Robust Recommendation System

Saber Shokat Fadaee  
College of Computer and Information Science  
Northeastern University  
saber@ccs.neu.edu

Hossein Azari Soufiani  
Google Research  
azari@google.com

Ravi Sundaram  
College of Computer and Information Science  
Northeastern University  
koods@ccs.neu.edu

ABSTRACT

Recommendation systems have been widely used by commercial service providers for giving suggestions to users based on their previous behaviors. While a large portion of users faithfully express their opinions, some malicious users add noisy ratings in order to change the overall ratings of a specific group of items. The presence of noise can add bias to recommendations, leading to instabilities in estimation and prediction. Although the robustness of different recommendation systems has been extensively studied, designing a robust recommendation system remains a significant challenge as detecting malicious users is computationally expensive. In this work, we propose Chiron, a fast and robust hybrid recommendation system that is not only faster than most state-of-the-art methods, but is also resistant to manipulation by malicious users.

1. INTRODUCTION

Users of commercial service providers such as Netflix, Spotify, and Amazon are provided with a large selection of recommended choices while using these online services. Recommendation systems aid users in the challenging task of finding the best video, music, book, or product out of all the possible options that they can have while using these systems. In this regard, collaborative filtering-based recommendation systems play an increasing role in helping people locate their favorite items in an immense dataset. In addition to providing helpful recommendations to users, these systems are also beneficial for the companies in raising their sales. However, since a good recommendation usually results in increased sales, some might find it profitable to shill recommendation systems by providing false information.

“Collaborative filtering (CF)” algorithms predict how much a user prefers a set of items, and produce a ranked list of items that would benefit or match her interests the most. In recommendation systems based on collaborative filtering, users rate specific items and receive recommendations for unrated ones. In “content-based” recommendation systems, users receive recommendations based on the domain knowledge of items or users. There are hybrid recommendation systems that combine both methods. All of these different systems are vulnerable to malicious attackers intending to manipulate the recommendations to suit their needs. Such attackers are known as “shills” and those attacks have been referred to as “shilling” or “Sybil” attacks.

Collaborative filtering methods do not use any information about users or items except for a partially observed rating matrix. The latter contains information provided by different users regarding different items, and the entries of this matrix are usually either binary or ordinal. In some recent works, this partially observed matrix was assumed low-ranked or locally low-ranked [19] and used matrix factorization methods for predicting missing values in the rating matrix [8]. The well-known matrix factorization methods are: singular value decomposition (SVD) [22], non-negative matrix factorization (NMF) [12], [11], probabilistic matrix factorization (PMF) [25], and non-linear probabilistic matrix factorization (NLPFM) [10].

An inquiry made by a recommendation system can be viewed as a rank aggregation problem and properties of models can be combined using statistical approaches [1, 27, 28]. We follow the same general direction, however, our model is concerned with rating data rather than ranking data.

In this work, we introduce Chiron, a fast and robust hybrid recommendation system. We conduct experimental studies to compare the robustness of our algorithm with the current state-of-the-art methods. We provide theoretical results to demonstrate its robustness to attacks by malicious users, and present experimental results to indicate why Chiron is the most robust recommendation system, and why the presence of an attack does not affect its performance.

1.1 Related Work

Many psychological studies have shown that people tend to agree with opinions of others regardless of their factual correctness. Cosley et al. [6] showed that prediction manipulation in a recommendation system can affect people in that system and, in some cases, mislead people into accepting a negative or unfitting recommendation. Therefore, people’s perceived value of items are influenced by the ratings of a recommendation system. Chirita et al. [6] demonstrated the presence of only three attackers in the neighborhood of one user is enough to create a significant change in prediction and move an unpopular item to the top five recommended items.

Lam et al. [8] and O’Mahony et al. [21] showed that many of the well-known recommendation systems are vul-
nerable to attacks and proposed different methods to distinguish honest raters from attackers. In another work, Resnick et al. [24] proposed a new method to limit the damage an attacker can do with a fixed number of Sybils and make effective use of information from honest raters. Another approach that has been studied is to motivate honest users to rate the items in order to counteract the attacks [3].

However, using detection algorithms as a preprocessing step can be computationally expensive. Therefore, others have proposed building robust recommendation systems [18, 19]. While recommendation systems have been widely investigated, less attention has been devoted to studying their vulnerability to manipulations. Mehta et al. [17] discussed a robust recommendation system’s characteristics.

O’Mahony et al. [20] performed empirical studies on the robustness of user-user kNN algorithms. They showed that attackers can successfully manipulate recommendation systems both by pushing and mucking attacks. Push attacks happen when vicious users try to increase ratings of specific items, and muke attacks happen when they do the opposite. They also presented a theoretical analysis of the effect of noise on the performance of CF algorithms and carried out several studies with real world datasets.

Seminario et al. [26] examined the trade-offs between accuracy and robustness of user-based and item-based CF recommendation systems and showed that the former achieves relatively positive marks on both properties. However, in an item-based CF recommendation system there exists a trade-off between its accuracy and robustness.

In this work, we propose a new generative model for recommendation systems that not only considers the users’ evaluations of items, but also takes into account the items’ evaluations of users. In some cases, items could be informative, and have their own evaluations of users. This extra information could be used to add priors to the system in order to promote users who give honest ratings.

2. CONTRIBUTION

We propose a new model which considers the quality of a rater (e.g. customers providing feedback), as well as the quality of an item (e.g. a restaurant or a product). We demonstrate how this approach is robust to attacks by malicious users. Additionally, we examine our model’s performance on three real-world datasets, and compare the success rate of attacks on our system as well as the state-of-the-art collaborative filtering recommendation systems. We also compare the running time of different methods on a different dataset and conclude that Chiron is not only robust to attacks, but is also the second fastest algorithm among the varied collection of current recommendation systems.

3. MODEL

Suppose we have $m$ raters who rate $n$ items with a score in the range of 1 to $K$. Now we define $P_i = (p_{i1}, ..., p_{iK})$ as the vector of probabilities of different ratings for item $i$, and $Q_j = (q_{j1}, ..., q_{jK})$ as the vector of probabilities of different ratings by rater $j$. Now in Chiron, the probability that an item $i$ receives rating $k$ by user $j$ is calculated as follows:

$$\Pr(r_{ij} = k) = \lambda_1 \theta_{ijk} + \lambda_2 \eta_{ijk} + \lambda_3 \tau_{ijk}$$  \hspace{1cm} (1)

In the formula $\eta_{ijk}$ loosely represents how likely are the neighbors of the user $j$ to give ranking $k$ to the item $i$ (a user-user kNN algorithm), and $\tau_{ijk}$ loosely represents how likely is the user $j$ to give the ranking $k$ to the items that are similar to the item $i$ (an item-item kNN algorithm). With $\lambda_1$, $\lambda_2$ and $\lambda_3$ we control the influence of each part of the model. Note that $\sum_{k=1}^{K} \lambda_k = 1$. Each user has a vector of latent preferences, and the influence of that vector is controlled by $\lambda_1$. We use $\lambda_2$ and $\lambda_3$ to control the influence of the neighbors of item $i$ and user $j$ on the final decision. The intuition behind this model is simple. Whether someone likes an item or not depends on three signals: the affinity between her latent preferences and the item’s latent attributes, and the influence of her friends on her decision and her decision regarding the similar items in the past.

Given a matrix of observed ratings for users and items, we first compute the Pearson’s correlation coefficient between ratings of a user $j$ and ratings of the rest of the users ($-j$), and the Pearson’s correlation coefficient of ratings of item $i$ and ratings of the rest of the items ($-i$). We assume user $j_1$ is friends with user $j_2$ if $j_2$ is among the top $\delta$ users who have preferences that have positive Pearson’s correlation with preferences of user $j_1$. In our model a real friend is someone who shares the same interests with us. We can define neighbors of specific item by using the same approach. The size of $\delta$ is usually set to either 50 or 100. For the sake of consistency, we use 50 for all the neighborhood methods.

After measuring the influences of a user’s friends on her decision, it is time to compute her latent preferences $\theta$. $\theta$ is responsible for keeping the model robust, and it is defined as follows:

$$\theta_{ijk} = \frac{p_{ik} q_{jk}}{\sum_{t=1}^{K} p_{it} q_{jt}} \hspace{1cm} (2)$$

Different combinations of $P_i$, $Q_j$, and corresponding $\Pr(r_{ij} = k)$ for three cases are illustrated in Figure 1. In the first case, we have a user $j$ who gives a high rating to most items (e.g. rates items only when she loves them). For an item $i$ which receives a poor rating by most users and other ratings with the same probability, Chiron predicts that user $j$ gives either the lowest or the highest rating to that item with a high probability.

In the second case, user $j$ rates items as 1 to 5 with an ascending probability, and item $i$ is rated with a descending probability. In this case, Chiron predicts that this user
would rate this item as 1.2.4, or 5 with the same probability and rates it as 3 with a lower probabiliy.

In the last case, user $j$ tends to rate items as either very good or very bad. On the other hand, we have an item $i$ which is mostly rated as average (3). Chiron predicts that user $j$ gives any rating to item $j$ with the same probability.

As a side note, we should also mention that $\theta$ somewhat reminiscent of Polytomous Rasch model[23].

With $\theta$, we try to evaluate ratings by considering users’ rating habits. Suppose for convenience $z_{ijk} = 1$ if $r_{ij} = k$, and $z_{ijk} = 0$ otherwise. Then the log likelihood function of (2) is:

$$
\mathcal{L}(P, Q; Z) = \sum_{ijk} z_{ijk} \log \frac{p_{ik}q_{jk}}{\sum_l p_{il}q_{jl}}
$$

(3)

$$
= \sum_{ijk} z_{ijk} \log p_{ik}q_{jk} - \sum_{ij} \log \sum_l p_{il}q_{jl}
$$

(4)

$$
= \sum_{ik} SI_{ik} \log p_{ik} + \sum_{jk} SJ_{jk} \log q_{jk}
$$

$$
- \sum_{ij} \log \sum_l p_{il}q_{jl}
$$

(5)

where $SI_{ik}$ denotes the number of times an item $i$ is rated $k$ by different users, and $SJ_{jk}$ denotes the number of times that a rater $j$ has rated different items with $k$.

4. ESTIMATION

In order to compute the maximum likelihood of $\theta$, we take the partial derivative of equation (3) with respect to $P$ and $Q$ and set them to zero:

$$
\frac{\partial \mathcal{L}(P, Q; Z)}{\partial p_{ik}^{*k*}} = \sum_j z_{ij}^{*k*} \frac{q_{jk}^{*k*}}{p_{ik}^{*k*}} - \sum_j \frac{q_{jk}^{*k*}}{\sum_l p_{il}q_{jl}}
$$

(6)

$$
\frac{\partial \mathcal{L}(P, Q; Z)}{\partial q_{jk}^{*k*}} = \sum_i z_{ij}^{*k*} \frac{p_{ik}^{*k*}}{q_{jk}^{*k*}} - \sum_j \frac{p_{ik}^{*k*}}{\sum_l p_{il}q_{jl}}
$$

(7)

Which leads to the following equations:

$$
q_{jk}^{*k*} = \frac{\sum_i z_{ij}^{*k*}}{\sum_i \sum_l p_{il}q_{jl} z_{ij}^{*k*}}
$$

(8)

$$
p_{ik}^{*k*} = \frac{\sum_j z_{ij}^{*k*}}{\sum_j \sum_l p_{il}q_{jl} z_{ij}^{*k*}}
$$

(9)

Algorithm 1 Chiron

```plaintext
procedure Initialize
for $l \leftarrow 1$ to $K$ do
    $q_{ul} \leftarrow p_{ul} \leftarrow 1.0$
end for

procedure Update P and Q
while converged == False do
    for $u \leftarrow 1$ to $m$ do
        for $j \leftarrow 1$ to $K$ do
            Update $q_{uj}$ using (8)
        end for
    end for
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow 1$ to $K$ do
            Update $p_{ij}$ using (9)
        end for
    end for
    if (10) and (11) then
        converged \leftarrow True
    end if
end while

procedure Prediction
for $u \leftarrow 1$ to $m$ do
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow 1$ to $K$ do
            Calculate $Pr_{uij}$ using (2)
            Calculate $\eta_{uij}$ using user-user kNN
            Calculate $\tau_{uij}$ using item-item kNN
        end for
        $r_{ui} \leftarrow 0.0$
        for $j \leftarrow \text{MinRating}$ to $\text{MaxRating}$ do
            $r_{ui} \leftarrow r_{ui} + \lambda_1(j \ast \text{Pr}_{uij}) + \lambda_2\eta_{uij} + \lambda_3\tau_{uij}$
        end for
    end for
end for
```

Since we would like to optimize both $p_{ik}$ and $q_{jk}$, our model is not convex. One possible solution is to fix one of the unknown parameters, and solve the optimization problem for the other. We use the average alternating projections method to provide a set of estimators. For this purpose we first fix the $p_{ik}$ and solve the optimization problem for the $q_{jk}$, and then fix the $q_{jk}$ and solve the problem for the $p_{ik}$ and continue until convergence. We assume the model converges when both of the following hold:

$$
\frac{1}{m} \sum_{1 \leq j \leq m} \sum_{k} (q_{jk} - q_{j+k})^2 < \epsilon
$$

(10)

and

$$
\frac{1}{n} \sum_{1 \leq i \leq n} \sum_{k} (p_{ik} - p_{i+k})^2 < \epsilon
$$

(11)

In equations (10) and (11), $\epsilon$ is a very small number, and min and max represent the minimum and maximum rating in a data set. In the three real-world data sets that we examined, Chiron converges in at most six steps.

Now in the following theorem, we prove that the presence of one or more users who rate(s) items with only one rating, does not affect the performance of Chiron.

**Theorem 1.** //The estimators above are robust to the existence of one or multiple zero entropy raters.

**Proof.** Assume without loss of generality that a user $u$ gives rating $r$ to a set of items, and does not rate the rest of items. Then in the equation (3), $q_{ukr} = 1$ for $k = r$, and $q_{uk} = 0$ for $k \neq r$. In this case, $\sum_{iuk} z_{iuk} \log \sum_l p_{il}q_{ul} = 0$. Therefore, the result of $\mathcal{L}(P, Q; Z)$ is independent of $Q_u$. \(\square\)

Our proposed algorithm is presented in algorithm 1. The updating phase runs in $O(tmnK + mn^2 \log n + nm^2 \log m)$. 


|
|---|---|---|---|---|---|
|Data set | m | n | # ratings | Average | Density |
|Netflix 3m1k | 4,427 | 1,000 | 56,136 | 3.275 | 1.27% |
|MoveLens 100K | 943 | 1,682 | 100,000 | 4.091 | 6.30% |
|MovieLens 1M | 6,039 | 3,883 | 1,000,181 | 3.233 | 4.27% |

Table 1: Data sets used in the experiments, and their characteristics including the number of users, items, total number of ratings available, overall average ratings for all items, and rating density defined as \(100 \times \frac{\text{number of ratings}}{m \times n}\).

In which \(t\) represents the number of updates before the algorithm converges, \(m\) is the number of users, \(n\) is the number of items, and \(K\) is the number of different rankings in the system (5 or 10). \(O(mn^2 \log n)\) is for calculating all the \(\tau_{ijk}\), and \(O(nm^3 \log m)\) is for calculating all the \(\eta_{jk}\). Since \(K\) is constant, and \(t\) is so small compared to \(n\) and \(m\), the running time complexity of Chiron is \(O(mn^2 \log n + nm^3 \log m)\).

5. EXPERIMENTAL SETTING

We examined the robustness, accuracy, and running time of Chiron in rating prediction and compared it with state-of-the-art recommendation systems\(^1\)\(^2\). There’s a trade off between robustness and accuracy, and we can set it using the \(\lambda_1, \lambda_2\) and \(\lambda_3\). If we want a more robust model we should give more weight to \(\lambda_1\), and if we want to make it more accurate we should rely more on the neighborhood methods. In our experiments we set \((\lambda_1 = 0.5)\), and give less weight to the influence of the other users \((\lambda_2 = 0.1)\), and more weight to the user’s ratings of similar items \((\lambda_3 = 0.4)\).

In subsection 5.1 we introduce the datasets we are using in our experiments. Then in 5.2 we introduce various attack strategies, and evaluation methods for comparing shilling attacks. After that we explain why we chose a specific kind of attack for experiments. Finally in 5.3 we introduce the different recommendation systems, and the toolkit we are using to compare them.

5.1 Datasets

We used the MoveLens100K, MovieLens1M, and Netflix3m1k databases for our experiments. The first two data sets are gathered by GroupLens Research Project\(^3\) at the University of Minnesota. The last one is provided by Netflix in the Netflix prize \(^2\). Prea software \(^15\) gathered all these data sets in its toolkit. In Table 1 we summarize information regarding each dataset.

In each dataset each user rated at least 20 movies from 1 (defined as did not like) to 5 (liked very much). We performed a cross-validation by splitting each dataset into a training set (80%) and a test set (20%) and did not use the information of the test set during the training of different algorithms, and compared the predicted ratings with actual ratings of the test set. We repeated our experiments 10 times and used the averaged results.

5.2 Attack Design

In this paper, we are only concerned with shilling attacks in which attackers try to manipulate a recommendation system by introducing fake users, and subsequently fake ratings. We only focus on push attacks since they are more successful than nuke attacks \(^9\). The effect of an attack is measured by the deviation in predicted ratings before and after adding the attack profiles. The most common metric for evaluating recommendation systems is Mean Absolute Error which is used to measure accuracy in predicting ratings:

\[
MAE = \frac{\sum_{i=1}^{n} |ActualRating_i - PredictedRating_i|}{n}
\]

In equation (12), \(n\) represents the total number of ratings predicted in the test run.

Two important metrics that are used for evaluation of different shilling attacks are the attack size and the filler size \(^17\). The attack size is the ratio of added shilling profiles to the original data set. For example, a 10% attack size indicates that the number of shilling profiles added to the system is equal to 10% of the users in the original data set. Another metric that is used for evaluation of different shilling attacks is the filler size. The filler size is the set of items which are voted for in the attacker profile.

We target a set of 20 items for the push attack. We repeat each experiment 10 times, and consider the mean value across these 10 times for each item in order to make sure our results are statistically significant.

The most effective attack models are derived by reverse engineering the recommendation algorithms to maximize their impact. As Burke et al. \(^4\) mentioned, the most common recommendation systems attack methods are random, average, and bandwagon. In a random attack, the assigned ratings made by attackers are around the overall mean rating with standard deviation 1.1. In average attacks, the assigned ratings made by attackers are around the mean rating of every item and standard deviation 1.1. Bandwagon attack is similar to the random attack, and some popular items are rated with the maximum rate.

Lam et al. \(^9\) observed that the user-user algorithms respond very well to all attacks. On the other hand, the item-item algorithms respond far less strongly. Moreover, push attacks on the user-user algorithm are more successful than nuke attacks. In that setting, no average or random attack increases the mean absolute error by more than 0.023. They also realized that nuke random attacks actually increase the prediction accuracy for item-based recommendation systems by a slight margin, but they couldn’t explain why. They suspected that if an item is pushed for a specific group of users, it causes the effect where the item is nuked for other users in the system.

Random and Bandwagon attacks do not require much knowledge about the set of items they are attacking. They only need information about some popular items and their overall means. Creating random ratings within a certain average interval will allow the attacker to have a high impact in making decisions for other users. On the other hand, average attacks require more information and are shown to be near optimal in impact \(^10\). They are also very challenging to detect \(^29\), and are stronger than random or bandwagon attacks \(^17\). Therefore, in this work we are only concerned with the average attacks. We are aiming to observe how Chiron performs in presence of the average attacks compared to other state-of-the-art methods.

5.3 Experiments

We use Prea \(^14\) to compare our proposed model with
different recommendation systems. The different algorithms we select to compare with Chiron fall into these two categories: memory-based neighborhood methods, and matrix factorization methods.

Memory-based neighborhood methods use the knowledge about similar users or items to give predictions about the unrated items. Memory based methods that we use in this experiment for comparison are: User-based Collaborative Filtering, User-based Collaborative Filtering (Default Voting), Item-based Collaborative Filtering, Item-based Collaborative Filtering (Inverse User Frequency), and Slope One.

On the other hand, matrix factorization methods build low-rank user or item profiles by factorizing training datasets with linear algebraic methods. The matrix factorization methods that we use in this experiment are: Regularized SVD, Non-negative Matrix Factorization (NMF), Probabilistic Matrix Factorization (PMF), Bayesian Probabilistic Matrix Factorization (BPMF), and Non-linear Probabilistic Matrix Factorization (NLPMF).

6. RESULTS AND DISCUSSION

The results of running different recommendation systems on the Netflix3M1K, MovieLens100K, and MovieLens1M datasets are shown in Table 2. We compared the prediction accuracy growth among different collaborative system methods after an attack size 100% on the first two data sets. A 100% attack is a shill, in which the number of fake users is equal to the number of genuine users. The mission of attackers is to promote a certain list of items (20 items in our experiment) and give an average rating to another set of unrated items. Memory based methods that we use in this experiment are: Regularized SVD, Non-negative Matrix Factorization (NMF), Probabilistic Matrix Factorization (PMF), Bayesian Probabilistic Matrix Factorization (BPMF), and Non-linear Probabilistic Matrix Factorization (NLPMF).

In another similar experiment, we attacked both the Netflix3M1K and MovieLens100K data sets with a 10% attack and gradually increased the attack size until it reached 100%. The changes in prediction accuracy are illustrated in Figure 2 and 3. In both of them the performance of Chiron almost remains unchanged with increases in the attack size while the prediction accuracy of other methods drops.

When some attackers are added to the data set, unlike some state-of-the-art methods Chiron does not require to process the entire data set again. Adding new users in the data set, only changes the related $P_i$s and $Q_j$s. Moreover, the results of our experiments suggest that Chiron is highly stable to low, moderate, and high number of attack profiles inserted on the attack items. Chiron is also able to handle multiple simultaneous attacks on different items.

7. CONCLUSION

Various methods exist for protecting recommendation systems against attacks by malicious users. More research has been done in the detection of attackers rather than proposing a robust recommendation system. Besides, the level of spam in real world data is often high, and simple spam detection methods are often reverse engineered. In this paper, we proposed a new and fast model which is empirically robust to attacks, explored its characteristics, and provided compelling evidence of its robustness. Chiron can not easily be manipulated since since it relies on modeling and other users ratings. Furthermore, we have illustrated its improved robustness and performance in comparison with other state-of-the-art methods and concluded that Chiron does not lose accuracy in the presence of shill attacks.

In future works, we plan to discuss more theoretical aspects of Chiron, and analyze its performance under different circumstances such as different entropy raters. We are also planning to apply this approach to different applications, e.g. aggregating the ratings in a P2P network. But most importantly, we are willing to user some Bayesian non parametric methods to compute the hyper parameters $\lambda_1$, $\lambda_2$ and $\lambda_3$.

8. ACKNOWLEDGMENTS

We thank Ardavan Saeedi for his insightful comments and discussion.

9. REFERENCES

[1] Hossein Azari Soufiani, David C Parkes, and Lirong Xia. A statistical decision-theoretic framework for social choice. In Advances in Neural Information Processing Systems 28, NIPS ’14, pages 3185–3193. 2014.
| Method                  | Data set           | Before | After | Growth | Before | After | Growth | Running time seconds |
|------------------------|--------------------|--------|-------|--------|--------|-------|--------|----------------------|
|                        | Netflix3M1K        | 0.772  | 0.985 | 27%    | 0.724  | 0.924 | 25%    | 1.4122               |
|                        | MovieLens100k      | 0.970  | 0.979 | 101%   | 0.964  | 0.965 | 0.5%   | 2.1570               |
|                        | MovieLens1M        | 0.769  | 0.976 | 28%    | 0.718  | 0.923 | 25%    | 28.0322              |
| user-based CF          |                    | 0.772  | 0.985 | 27%    | 0.724  | 0.924 | 25%    | 1.4122               |
| user-based DF          |                    | 0.769  | 0.976 | 28%    | 0.718  | 0.923 | 25%    | 28.0322              |
| item-based CF          |                    | 0.756  | 0.970 | 28%    | 0.722  | 0.923 | 27%    | 10.5486              |
| item-based DF          |                    | 0.769  | 0.976 | 28%    | 0.718  | 0.923 | 25%    | 28.0322              |
| Slope One              |                    | 0.775  | 1.045 | 34%    | 0.744  | 0.985 | 32%    | 22.48                |
| Regular SVD            |                    | 0.819  | 1.528 | 86%    | 0.729  | 0.982 | 34%    | 363.97               |
| Non negative MF        |                    | 0.868  | 1.745 | 101%   | 0.780  | 1.043 | 33%    | 1.045                |
| Probabilistic MF       |                    | 0.786  | 1.280 | 62%    | 0.775  | 0.984 | 26%    | 793.22               |
| Bayesian PMF           |                    | 0.793  | 1.319 | 66%    | 0.745  | 0.977 | 31%    | 14.20                |
| Non-linear PMD         |                    | 0.847  | 1.797 | 112%   | 0.841  | 0.919 | 9%     | 2.551.39             |
| Chiron                 |                    | 0.781  | 0.951 | 21%    | 0.744  | 0.910 | 22%    | 37.86                |

Table 2: Changes in prediction accuracy after an attack size of 100% in the Netflix3M1K, and MovieLens100k data sets, plus running time of different algorithms in the MovieLens1M data set. The statistically significant result is shown in bold in each column.

[2] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In In KDD Cup and Workshop in conjunction with KDD, 2007.

[3] Rajat Bhattacharjee and Ashish Goel. Algorithms and incentives for robust ranking. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’07, pages 425–433, Philadelphia, PA, USA, 2007.

[4] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. Classification features for attack detection in collaborative recommender systems. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’06, pages 542–547, New York, NY, USA, 2006. ACM.

[5] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In Proceedings of the 9th ACM Conference on Recommender Systems, pages 43–50. ACM, 2015.

[6] Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir. Preventing shilling attacks in online recommender systems. In Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management, WIDM ’05, pages 67–74, New York, NY, USA, 2005. ACM.

[7] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users’ opinions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’03, pages 585–592, New York, NY, USA, 2003. ACM.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, August 2009.

[9] Shyong K. Lam and John Riedl. Shilling recommender systems for fun and profit. In Proceedings of the 13th International Conference on World Wide Web, WWW ’04, pages 393–402, New York, NY, USA, 2004. ACM.

[10] Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML ’09, pages 601–608, New York, NY, USA, 2009. ACM.

[11] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, October 1999.

[12] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing Systems 13, NIPS ’01, pages 556–562, MIT Press, 2001.

[13] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In ICML (2), volume 28 of JMLR Proceedings, pages 82–90. JMLR.org, 2013.

[14] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. CoRR, abs/1205.3193, 2012.

[15] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. Prea: Personalized recommendation algorithms toolkit. J. Mach. Learn. Res., 13(1):2699–2703, September 2012.

[16] Bhaskar Mehta. Unsupervised shilling detection for collaborative filtering. In Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2, AAAI’07, pages 1402–1407. AAAI Press, 2007.

[17] Bhaskar Mehta and Thomas Hofmann. A survey of attack-resistant collaborative filtering algorithms. IEEE Data Eng. Bull., 31(2):14–22, 2008.

[18] Bhaskar Mehta and Wolfgang Nejdl. Attack resistant collaborative filtering. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’08, pages 75–82, New York, NY, USA, 2008. ACM.

[19] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Trans. Internet Technol., 7(4), October 2007.

[20] Michael P. O’Mahony, Neil J. Hurley, and Guénaël C. M. Silvestre. Detecting noise in recommender system databases. In Proceedings of the 11th International Conference on Intelligent User Interfaces, IUI ’06, pages 109–115, New York, NY,
[21] Michael P. O’Mahony, Neil J. Hurley, and Guenole C. M. Silvestre. Promoting recommendations: An attack on collaborative filtering. In *Database and Expert Systems Applications*, volume 2453 of *Lecture Notes in Computer Science*, pages 494–503. Springer Berlin Heidelberg, 2002.

[22] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup Workshop at SIGKDD’07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, 2007.

[23] G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. MESA Press, 1960.

[24] Paul Resnick and Rahul Sami. Manipulation-resistant recommender systems through influence limits. *SIGecom Exchanges*, 7(3), 2008.

[25] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML ’08, pages 880–887, New York, NY, USA, 2008. ACM.

[26] Carlos E. Seminario and David C. Wilson. Robustness and accuracy tradeoffs for recommender systems under attack. In *FLAIRS Conference*. AAAI Press, 2012.

[27] Hossein Azari Soufiani, Hansheng Diao, Zhenyu Lai, and David C. Parkes. Generalized random utility models with multiple types. In *Advances in Neural Information Processing Systems 27*, NIPS ’13, pages 73–81, 2013.

[28] Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. Random utility theory for social choice. In *Advances in Neural Information Processing Systems 26*, NIPS ’12, pages 126–134, 2012.

[29] Sheng Zhang, Yi Ouyang, James Ford, and Fillia Makedon. Analysis of a low-dimensional linear model under recommendation attacks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’06, pages 517–524, New York, NY, USA, 2006. ACM.