

Securing Collaborative Filtering Against Malicious Attacks Through Anomaly Detection *

Runa Bhaumik, Chad Williams, Bamshad Mobasher, Robin Burke

Center for Web Intelligence, DePaul University
School of Computer Science, Telecommunication, and Information Systems
Chicago, Illinois, USA
{rbhaumik, cwilli43, mobasher, rburke}@cs.depaul.edu

Abstract

Collaborative filtering recommenders are highly vulnerable to malicious attacks designed to affect predicted ratings. Previous work related to detecting such attacks has focused on detecting profiles. Approaches based on profile classification to a large extent depend on profiles conforming to known attack models. In this paper we examine approaches for detecting suspicious rating trends based on statistical anomaly detection. We empirically show these techniques can be highly successful at detecting items under attack and time intervals when an attack occurred. In addition we explore the effects of rating distribution on detection performance and show that this varies based on distribution characteristics when these techniques are used.

Introduction

Recent research has established that traditional collaborative filtering algorithms are vulnerable to various attacks. These attacks can affect the quality of predictions and recommendations for many users, resulting in decreased users' trust of the system. Such attacks have been termed "shilling" attacks in previous work (Burke *et al.* 2005; Burke, Mobasher, & Bhaumik 2005; Lam & Riedl 2004; O'Mahony *et al.* 2004). The more descriptive phrase "profile injection attacks" is also used, and better conveys the tactics of an attacker. The aim of these attacks is to bias the system's output in the attacker's favor. User-adaptive systems, such as collaborative filtering, are vulnerable to such attacks precisely because they rely on user interaction to generate recommendations.

The fact that these profile injection attacks are effective raises the question whether system owners can detect that there is a potential attack against an item. Some researchers have focused on detecting and preventing profile injection attacks. In (Chirita, Nejdl,

& Zamfir 2005) several metrics were proposed for analyzing rating patterns of malicious users and evaluated their potential for detecting such profile injection attacks. A spreading similarity algorithm was developed in order to detect groups of very similar profile injection attackers which was applied to a simplified attack scenario in (Su, Zeng, & Z.Chen 2005). Several techniques were developed to defend against the attacks in (O'Mahony *et al.* 2004; Lam & Riedl 2004), including new strategies for neighborhood selection and similarity weight transformations (O'Mahony, Hurley, & Silvestre 2004). In (Burke *et al.* 2006) a model-based approach to detection attribute generation was introduced and shown to be effective at detecting and reducing the effects of random and average attack models. Their research on preventing profile injection attacks have focused on detecting suspicious user profiles; we examine an alternate approach to this problem by focusing instead on detecting items with suspicious trends using statistical anomaly detection techniques. We envision this type of detection as a component of a comprehensive detection approach.

The main contribution of this paper is an alternate approach to attack detection that does not rely on profile classification. Instead we introduce an item-based approach to detection that identifies what items may be under attack based on rating activity related to the item. In this paper we present and evaluate two **Statistical Process Control** (SPC) techniques. We investigate the X-bar control limit and Confidence Interval control limit techniques for detecting items which are under attack. Our second approach attempts to identify time intervals of rating activity that may suggest an item is under attack. We examine a time series algorithm which successfully identifies suspicious time intervals.

Statistical process control charts have been used in manufacturing to detect whether a process is out-of-control (Shewart 1931). In general, a SPC is composed of two phases. In the first phase the technique estimates the process parameters from historical events and then uses these parameters to detect out-of-control anomalies for recent events. Recommender systems which collect ratings from users can also be thought of as a

*This research was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

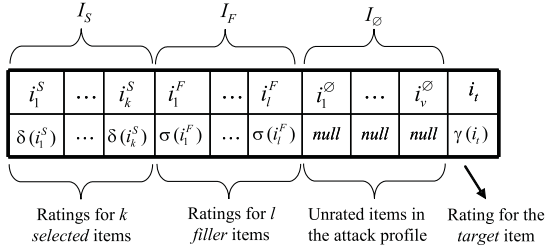


Figure 1: The general form of an attack profile.

similar processes. The rating patterns can be monitored for abnormal rating trends that deviate from the past distribution of items. Our results show that both SPC approaches work well in identifying suspicious activity related to items which are under attack. For time interval detection, our results show this technique performs well at identifying suspicious time intervals, over which an item is under attack.

In addition to the above approaches our research raises the question whether different rating distributions of the target items influence detection effectiveness. Intuitively, items with few ratings or low average rating are easier to manipulate by attackers. On the other hand, attacks against items with many ratings and high average rating will be harder to detect as users have already rated these items highly. Moreover, due to large number of items in our database, it is not possible to monitor each movie separately. Therefore, we have categorized our items based on the rating distribution. The goal of this categorization is to make the distributions within each of the categories more similar. The items that makeup each of these categories are then used to create the process control model for other items within the same category. Our results also confirm that detection performance of suspicious items varies based on rating distribution.

The paper is organized as follows. First, we discuss the attack models previously studied and their impact. We then describe the detailed algorithms and experimental methodology for our detection model. Finally, we present our experimental results and discussion.

Profile Injection Attacks

A profile injection attack against a recommender system consists of a set of *attack profiles* being inserted into the system with the aim of altering the system’s recommendation behavior with respect to a single target item i_t . An attack that aims to promote i_t , resulting in it recommended more often, is called a *push attack*, and one designed to make i_t recommended less often is called a *nuke attack* (O’Mahony *et al.* 2004).

An attack model is an approach to constructing attack profiles, based on knowledge about the recommender system: its rating database, its products, and/or its users. The attack profile consists of an m -dimensional vector of ratings, where m is the total num-

ber of items in the system. The profile is partitioned in four parts as depicted in Figure 1. The null partition, I_\emptyset , are those items with no ratings in the profile. The single target item i_t will be given a rating designed to bias its recommendations, generally this will be either the maximum or minimum possible rating, depending on the attack type. As described below, some attacks require identifying a group of items for special treatment during the attack. This special set I_S usually receives high ratings to make the profiles similar to those of users who prefer these products. Finally, there is a set of filler items I_F whose ratings are added to complete the profile. It is the strategy for selecting items in I_S and I_F and the ratings given to these items that define the characteristics of an attack model. The symbols δ , σ and γ are functions that specify how ratings should be assigned to items in I_S and I_F and for the single target item i_t respectively.

Two basic attack models, originally introduced in (Lam & Riedl 2004) are *random attack*, and *average attack*. In our formalism, for these two basic attacks I_S is empty, and the contents of I_F are selected randomly. For random attack all items in I_F are assigned ratings based on the function σ , which generates random ratings centered around the overall average rating in the database. The average attack is very similar, but the rating for each filler item in I_F is computed based on more specific knowledge of the individual mean for each item. For more complex attacks the I_S set may be used to leverage additional knowledge about a set of items. For example the bandwagon attack selects a number of popular movies for the I_S set which are given high ratings and I_F is populated as described in the random attack (Burke *et al.* 2005). Likewise the segment attack populates I_S with a number of related items which it gives high ratings to, while the I_F partition is given the minimum rating in order to target a segment of users (Burke *et al.* 2005).

Previous research on detecting and preventing the effects of profile injection attacks have focused on specific attack models (Su, Zeng, & Z.Chen 2005; O’Mahony, Hurley, & Silvestre 2004; Burke *et al.* 2006). Our approach, however, is to detect suspicious ratings for an individual item independent of any attack model. Instead we analyze the rating trends of items and identify suspicious changes in these trends which may indicate a particular item is under attack.

Detection Model

In this section we describe two SPC techniques for detecting items under attack, as well as a time-series technique for detecting time intervals an item is under attack.

Statistical Process Control

SPC is often used for long term monitoring of feature values related to the process. Once a feature of interest has been chosen or constructed, the distribution of this

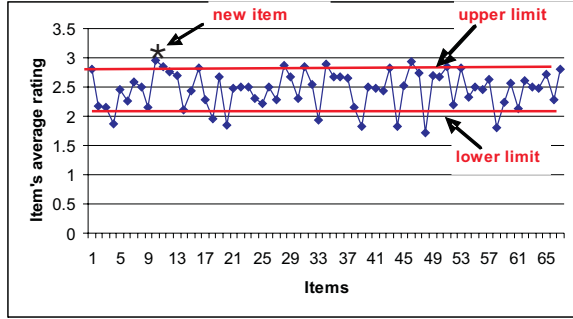


Figure 2: Example of a control chart

feature can be estimated and future observations can be automatically monitored. Control charts are used routinely to monitor quality in SPC. Figure 2 displays an example of a control chart in the context of collaborative filtering where observations are average rating of items, which we assume are not under attack. Two other horizontal lines, called upper limit and lower limit are chosen so that almost all of the data points will fall within these limits as long as there is no threat in the system. This figure depicts that a new item's average rating is outside the upper limit. This indicates an anomaly that possibly be an attack. In the following section we describe two different control limits as our detection scheme.

X-bar control limit One commonly used types of control chart is the X-Bar chart which plots how far away from the average value of a process the current measurement falls. According to (Shewart 1931), values that fall outside of three sigma standard deviation of the average have a valid cause and are labeled as being out of statistical control.

In the context of recommender systems, let us consider the case where we have to identify whether a new item is under attack by analyzing past data. Suppose we have collected k items with similar rating distribution in the same category from our database. Let n_i be the number of users who have rated an item i . According to (Shewart 1931) we can define our upper ($U\bar{x}$) and lower ($L\bar{x}$) control limits as follows:

$$U\bar{x} = \bar{X} + \frac{A*\bar{S}}{c_4(n)\sqrt{n}}$$

$$L\bar{x} = \bar{X} - \frac{A*\bar{S}}{c_4(n)\sqrt{n}}$$

where \bar{X} is the grand mean rating of k items, and \bar{S} is the average standard deviation which can be computed as:

$$\bar{S} = \frac{\sum_{i=1}^k s_i}{k}$$

with s_i being the standard deviation of each item. As n_i is different for each item i , n can be taken as the average of all n_i . The auxiliary function $c_4(n) =$

$\sqrt{\frac{2}{n-1}}\Gamma(\frac{n}{2})\Gamma(\frac{n-1}{2})$, where $\Gamma(t)$ is a complete gamma function which is expressed as $(t-1)!$. When $n \geq 25$, $c_4(n)\sqrt{n}$ can be approximated by $\sqrt{(n-.5)}$ and A is a constant value which determines the upper and lower limit (SPSS 2002). Thus when A is set to 3, we get 3-sigma limit. We set $U\bar{x}$ and $L\bar{x}$ as a signal threshold. A new item is likely under attack, if the average rating is greater than $U\bar{x}$ or less than $L\bar{x}$.

Confidence Interval Control Limit The Central Limit Theorem is one of the most important theorems in statistical theory (Ott 1992). It states that distribution of the sample mean becomes more normalized as the sample size increases. This means that we can use the normal distribution to describe the sample mean from any population, even non-normal ones, if we have a large enough sample. The general rule of thumb is that you need a sample of at least 30 observations for the Central Limit Theorem to apply (i.e., for the distribution of the sample mean to be reasonably approximated with the normal distribution). A confidence interval, or interval estimate, is a range of values that contains the population mean with a level of confidence that the researcher chooses. For example, a 95% confidence interval would be a range of values that has a 95% chance of containing the population mean.

Suppose we have collected a set of k items with similar rating distribution in the same category, and $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_k$ are the mean rating of these k items. The upper ($U\bar{x}$) and lower ($L\bar{x}$) control limits for these sample means can be written as:

$$U\bar{x} = \bar{X} + \frac{C*\sigma}{\sqrt{k}}$$

$$L\bar{x} = \bar{X} - \frac{C*\sigma}{\sqrt{k}}$$

where \bar{X} and σ is the mean and standard deviation of \bar{x}_i 's. The value of C is essentially the z-value for the normal distribution. For example, the value is 1.96 for a 95% confidence coefficient.

In a movie recommender system, the upper and lower boundaries of the confidence interval are considered as the signal threshold for push and nuke attacks respectively. If our confidence coefficient is set to .95, we are 95% sure that all the item averages will fall inside these limits and when an average rating of an item is outside of these limits, we consider the ratings related to this item suspicious.

Time Interval Detection Scheme

The normal behavior of a recommender system can be characterized by a series of observations over time. When an attack occurs, it is essential to detect the occurrences of abnormal activity as quickly as possible, before significant performance degradation. This can be done by continuously monitoring the system for deviations from the past behavior patterns. In a movie recommender system the owner could be warned of a possible attack by identifying the time period during which abnormal rating behavior occurred for an item. Most

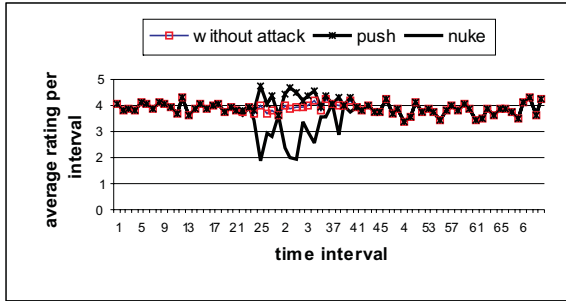


Figure 3: A time series chart

anomaly detection algorithms require a set of training data without bias for training and they implicitly assume that anomalies can be treated as patterns not observed before. Distributions of new data are then compared to the distributions obtained from the training data and differences between the distributions indicate an attack.

In the context of recommender systems, we can monitor an item’s ratings over a period of time. A sudden jump in an item’s mean rating may indicate a suspicious pattern. One can compare the average rating for this item by collecting ratings over a period of time, assuming there are no biased ratings. When new ratings are observed, a system can compare the current average rating to the data collected before. Figure 3 shows an example of a time series pattern before and after an attack in a movie recommender system. The upper and lower curves show the rating pattern of an item after push and nuke attack respectively, whereas the middle curve shows the rating pattern without any attack.

Our time series data can be expressed as a sequence of $\bar{x}_i^t : t = 1, 2, \dots$ where t is a time variable and each \bar{x}_i^t is the average rating of an item i at a particular time t . Suppose μ_i^k and σ_i^k are the mean and standard deviation estimated from the ratings collected from a trusted source for the first k -th interval for an item i . Our algorithm is based on calculating the probability of observing the mean rating for the new time interval. If it is outside a pre-specified threshold, then it deviates significantly from the rating population indicating an attack. Now if \bar{x}_i^t is the average rating for an interval t after the k -th interval, our conditions for detecting an attack interval t is:

$$\bar{x}_i^t > \mu_i^k + Z_{\frac{\alpha}{2}} \frac{\sigma_i^k}{\sqrt{n}}$$

and

$$\bar{x}_i^t < \mu_i^k - Z_{\frac{\alpha}{2}} \frac{\sigma_i^k}{\sqrt{n}}$$

for push and nuke attack respectively. The parameter n is the total number of ratings for the first k -th interval of an item i . The value for $Z_{\frac{\alpha}{2}}$ is obtained from a

normal distribution table for a particular value of α . This algorithm essentially detects the time period over which an item is potentially under attack.

Experimental Methodology

Dataset

We have used the publicly-available Movie-Lens 100k dataset¹ for our experiments. All ratings are integer values between 1 and 5, where 1 is the lowest (most disliked) and 5 is the highest (most liked). Our data includes all the users who have rated at least 20 movies. For all the attacks, we generated a number of attack profiles and inserted them into the system database and then evaluate each algorithm. We measure “size of attack” as a percentage of the pre-attack user count. There are approximately 1000 users in the database, so an attack size of 1% corresponds to 10 attack profiles added to the system.

We propose examining items against the distributions of items with similar characteristics which we term categories. The goal of this categorization is to make the distributions within each of the categories more similar within the underlying populations. The items that makeup each of these categories are then used to create the process control model for other items within the same category. We have categorized items in the following way.

First we defined two characteristics of movies, density (# of ratings) and average rating in the following way.

- low density (LD): # of ratings between 25 and 40
- medium density (MD): # of ratings between 80 and 120
- high density (HD): # of ratings between 200 and 300
- low average rating (LR): average rating less than 3.0
- high average rating (HR): average rating greater than 3.0

Then we partitioned our dataset into five different categories LDLR, LDHR, MDLR, MDHR, and HDHR. For example, category LDLR contains movies which are LD and LR. Table 1 shows the statistics of the different categories computed from the MovieLens dataset. The category HDLR which is high density and low average rating has not been analyzed here due to insufficient examples in the Movie-Lens 100k dataset.

Evaluation Metrics

In order to validate our results we have considered two performance metrics, precision and recall. In addition to investigating the trade offs between these metrics, we seek to investigate how the parameters of the detection algorithm and the size of attacks affect the performance. In our experiments, precision and recall have been measured differently depending on what’s being

¹<http://www.cs.umn.edu/research/GroupLens/data/>

Category	Average # of Ratings	Average Rating
HDHR	245.68	3.82
LDLR	31.26	2.61
LDHR	32.37	3.5
MDHR	97.5	3.54
MDLR	87.45	2.68

Table 1: Rating distribution for all categories of movies

identified. The basic definition of recall and precision can be written as:

$$precision = \frac{\# \text{ true positives}}{(\# \text{ true positives} + \# \text{ false positives})}$$

$$recall = \frac{\# \text{ true positives}}{(\# \text{ true positives} + \# \text{ false negatives})}$$

In statistical control limit algorithms, we are mainly interested in detecting movies which are under attack. So $\# \text{ true positives}$ is the number of movies correctly identified as under attack, $\# \text{ false positives}$ is the number of movies that were misclassified as under attack, and $\# \text{ false negatives}$ is the number of movies which are under attack that are misclassified.

On the other hand, in the case of time interval detection, we are interested in detecting the time interval during which an attack occurred on an item. So $\# \text{ true positives}$ is the number of time intervals correctly identified as being under attack, $\# \text{ false positives}$ is the number of time intervals that were misclassified as attacks, and $\# \text{ false negatives}$ is the number of time intervals that were misclassified as no attack.

Category	Training	Test
HDHR	50	30
LDLR	50	30
LDHR	50	50
MDHR	50	50
MDLR	30	14

Table 2: Total number of movies selected from each category in training and testing phases

Methodology for detection via control limits

In SPC, the process parameters are estimated using historical data. This process is accomplished in two stages, training and testing. In the training stage, we use the historical ratings to estimate the upper and lower control limits. In the testing stage, we compare the new item’s average rating with these limits. If the current average rating is outside of the boundaries we consider that an attack. Table 2 shows the number of movies selected during training and testing phases. In the training phase, we used the ratings for all movies in the training set to compute the control limits.

Our evaluation has been done in two phases. In the first phase, we calculated the average rating for each movie in the test set and checked whether it lies within the control limits. We assumed that the MovieLens dataset had no biased ratings for these movies and considered the base rating activity to have *no attack*. We then calculated the false positives, which are the number of *no attack* movies that were misclassified as under attack by our detection algorithm. In the second phase we generated attacks for all the movies in the test set. Two types of attacks were considered: push and nuke. For push attacks we gave a maximum possible rating 5 and for nuke attacks we gave a minimum possible rating 1. The average rating of each movie was then computed and checked whether it fell within the control limits. We then computed true positives, the number of movies correctly identified as under attack and false negatives, the number of movies which are under attack that were misclassified as *no attack* movies. Precision and recall have then been computed using the formula in the evaluation section.

Methodology for time interval detection

For the time interval detection algorithm, we relied upon the time-stamped ratings which were collected in the MovieLens dataset over a seven month period. Our main objective here is to detect the time interval over which an attack is made. The original dataset was sorted by the time-stamps given in the MovieLens dataset, and broken into 72 intervals, where each interval consists of 3 days.

For each category, we selected movies from the test set shown in Table 2. For each test movie, first we obtained ratings from sorted time-stamp data and computed mean and standard deviation prior to the t -th interval, which we assume contains no biased ratings. We set t to 20, which is equivalent to two months. Our assumption here is that the system owner has collected data from a trusted source prior to the t -th interval, which is considered as historical data without any biased ratings.

An attack (push or nuke) was then generated and inserted between the t -th and $(t + 20)$ -th interval chosen at random times. We choose a long period of attack (20 intervals) so that an attacker can easily disguise himself as a genuine user and will not be easily detectable. During this time, we identified the time intervals as *attack* or *no attack* depending on whether our system generates an attack at that time interval or not. For each subsequent interval starting at t -th interval, we computed the average rating of the movie. If the average rating deviate significantly from the distribution of historical data, we considered this interval as a suspicious one. At this stage, we calculated the precision and recall for detecting attack intervals for each movie and averaged over all test movies.

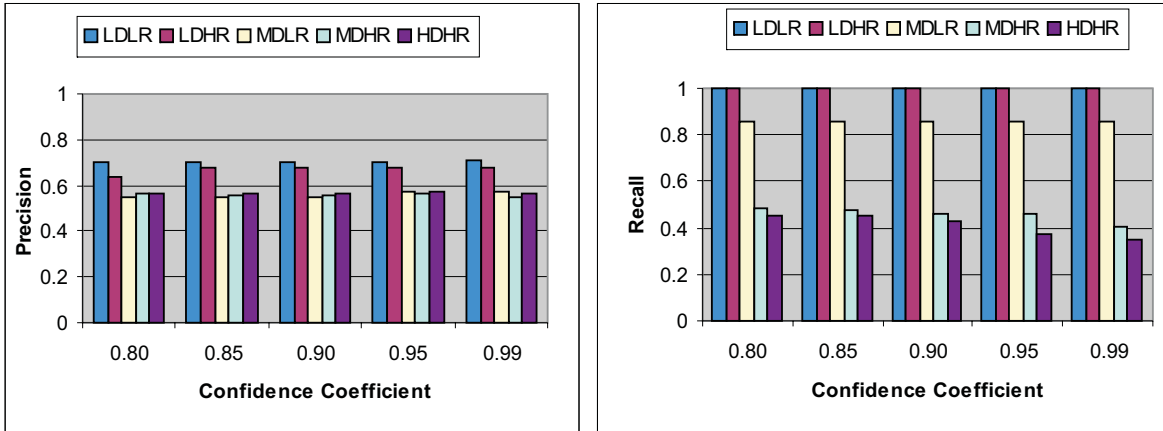


Figure 4: The Precision and Recall graphs for all categories varying confidence coefficient at 1% push attack size using Confidence Interval control limits

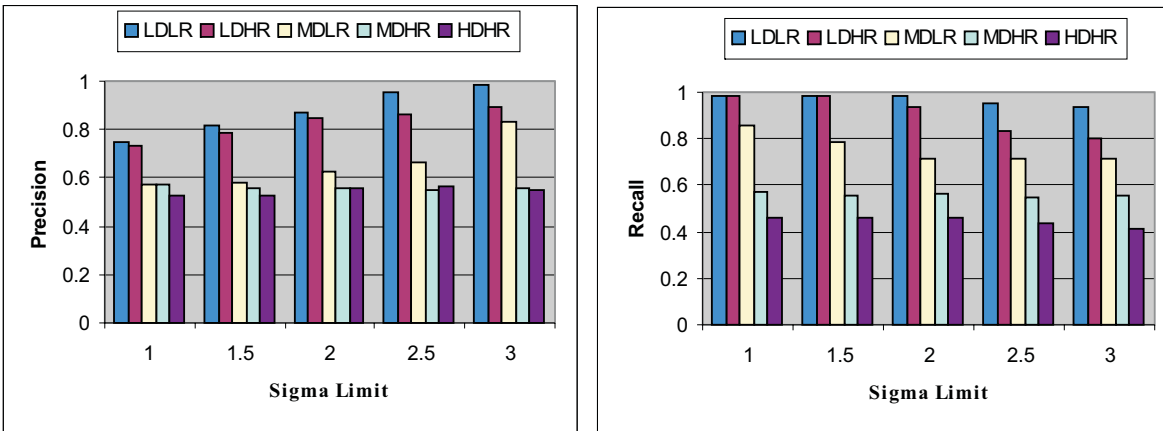


Figure 5: The Precision and Recall graphs for all categories varying sigma limit at 1% push attack size using X-bar control limits

Experimental Results

In our first set of experiments we built a predictive model for different categories of movies using SPC algorithms. Test items were then classified as either an item under attack or not under attack. We varied the parameters of the detection algorithms throughout our experiments to examine their affect on detection performance.

Figure 4 shows the effect of the confidence coefficient for all categories in a push attack using the Confidence Interval algorithm. In this case, precision and recall did not change significantly while confidence coefficients change from .80 to .99, which indicates the number of false positives and true positives remain the same within this range for different categories.

On the other hand, we can observe in Figure 5, that as sigma increases, precision increases and recall decreases for all categories except HDHR and MDHR in a

push attack using X-bar algorithm. As sigma increases the upper limit increases and the average rating of an item which is not under attack from these two categories still falls between the limits and is classified correctly. Recall results reflect the intuition that the average rating of a target item with a high average rating does not increase enough to fall outside the upper control limit; thus the item is misclassified as an item not under attack. The results shown here confirm that items with few ratings and low average are easy to detect using this technique.

Figure 6 shows the results for all categories of movies at different attack sizes in a push attack, using X-Bar algorithm where sigma limit is set to 3. The recall chart shows that at lower attack sizes precision and recall are low for both the HDHR and MDHR categories. This observation is consistent with our conjecture that if an item is already highly rated then it is hard to distinguish

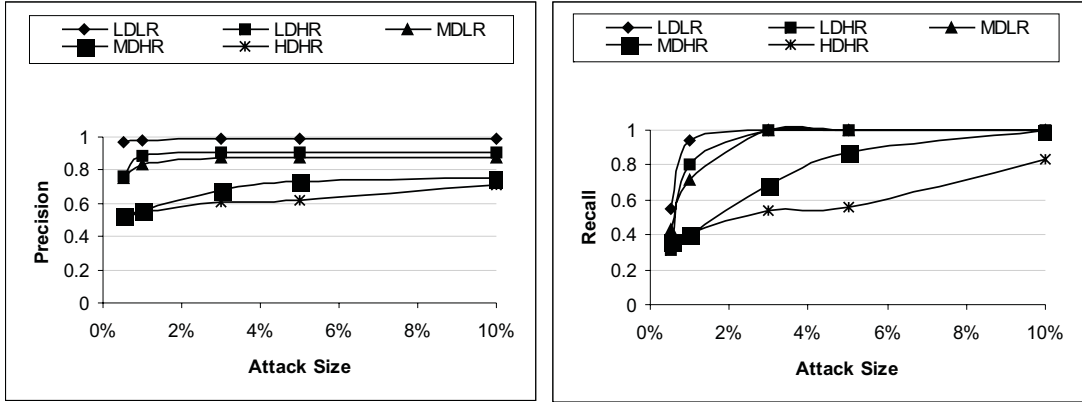


Figure 6: The Precision and Recall graphs for all categories, varying push attack sizes using X-Bar control limits (sigma limit set to 3)

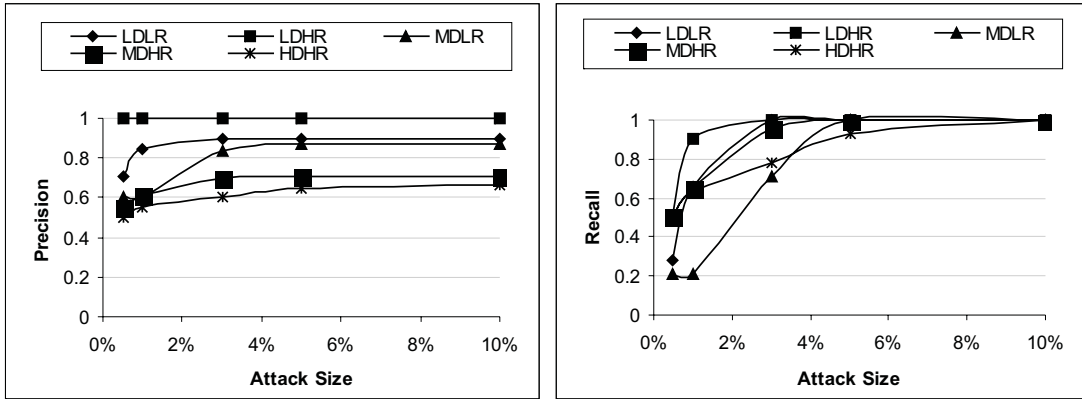


Figure 7: The Precision and Recall graphs for all categories, varying nuke attack sizes using X-Bar control limits (sigma limit set to 3)

from other items in this category after an attack. On the other hand the recall measures for LDLR, LDHR, and MDLR are 100% at 3% attack size. The lower the densities or average rating, the higher the recall values at different attack sizes. Similar results were obtained (not shown here) for the confidence interval control limit algorithm. Precision values in X-bar algorithm are much higher than Confidence Interval algorithm indicating X-bar algorithm works well in terms of classifying correctly no attack items, although both algorithms work for identifying suspicious items. As the figures depict, the performances also vary with different categories of items. This is consistent with our conjecture as the number of ratings (density) decrease, the algorithms detect the suspicious items more easily.

The next aspect we examined was the effectiveness of detection in the face of nuke attacks against all categories of movies. Figure 7 shows the results for all categories of movies varying attack sizes in a nuke attack using X-Bar algorithm where sigma limit is set to

3. The precision increases as the attack size increases, indicating this algorithm produces fewer false positives at higher attack sizes. The recall chart shows that the detection rate is very high even at 3% attack size for all categories. At lower attack sizes low density movies are more detectable than higher densities against nuke attack. It is reasonable to assume that the higher the densities, the lower the chance of decreasing average rating below the lower limit at lower attack sizes. The results depicted here confirm that this algorithm is also effective at detecting nuke attacks and the performance varies with different categories. The same trend has been obtained for the Confidence Interval algorithm not shown here.

The main objective of the time series algorithm is to detect a possible attack by identifying time intervals during which an item's ratings are significantly different from what is expected. In this experiment, first we obtained ratings from sorted time-stamp data for each item in the test dataset and computed mean and

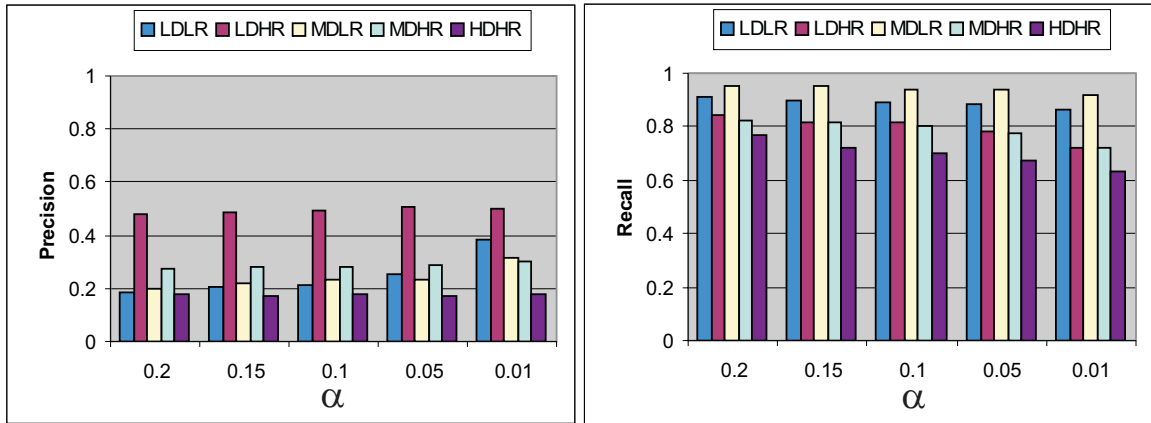


Figure 8: The Precision and Recall graphs for all categories varying sigma at 1% push attack size using time series algorithm

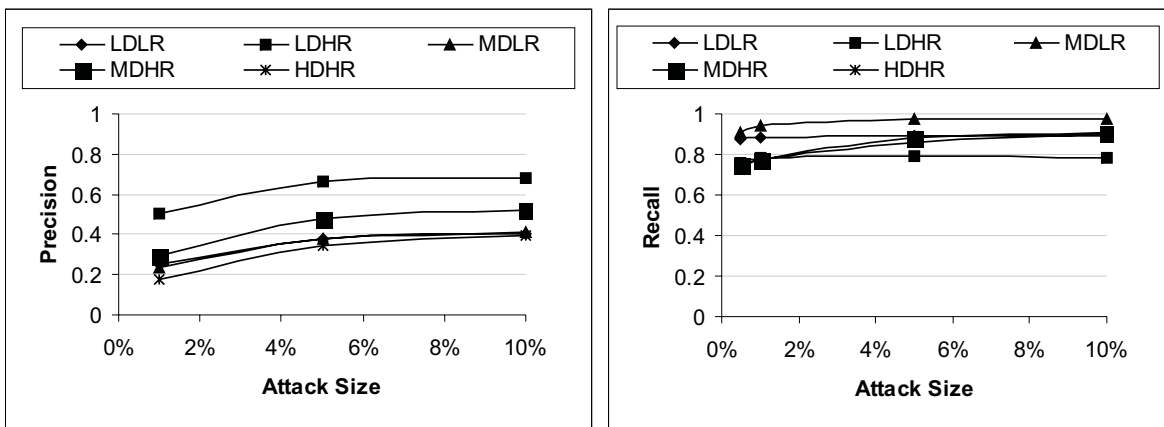


Figure 9: The Precision and Recall graphs for all categories, varying push attack sizes using time series algorithm (α set to .05)

standard deviation prior to the t -th interval, which we assume contains no biased ratings and consider this as our historical data. An attack (push or nuke) was then generated and inserted between the t -th and $(t+20)$ -th interval chosen at random time. Now for each subsequent interval starting at t -th interval, we compute the average rating of the movie. If the average rating deviates significantly from the distribution of historical data, we flag this interval as a suspicious one.

Figure 8 shows the effect of detection threshold for all categories in a 1% push attack using the time series algorithm. The result shows clearly that recall decreases as the probability of detecting an attack interval decreases for all categories except LDLR and MDLR. These two categories which are already low rated, a small attack could not increase average rating significantly in a time interval. On the other hand, precision is inversely affected by α values with low rated items.

In this case, the highly rated item's average appeared to be as an attack in a time interval. Moreover, precision are very small at 1% attack size for all categories.

The overall effect of this algorithm against all categories of movies are shown in Figure 9 against a push attack. The time interval detection rate for highly rated items is low at small attack sizes which indicates that it is very hard to detect the attack interval against a push attack. On the other hand, the results are opposite in nature against a nuke attack which is depicted in Figure 10. As expected, the highly rated items are easily detectable against a nuke attack.

The time interval results show that the period in which attacks occur can also be identified effectively. This approach offers some particularly valuable benefits by not only identifying the target of the attack and the type of attack (push/nuke), but also identifies the time interval over which the bias was injected. This

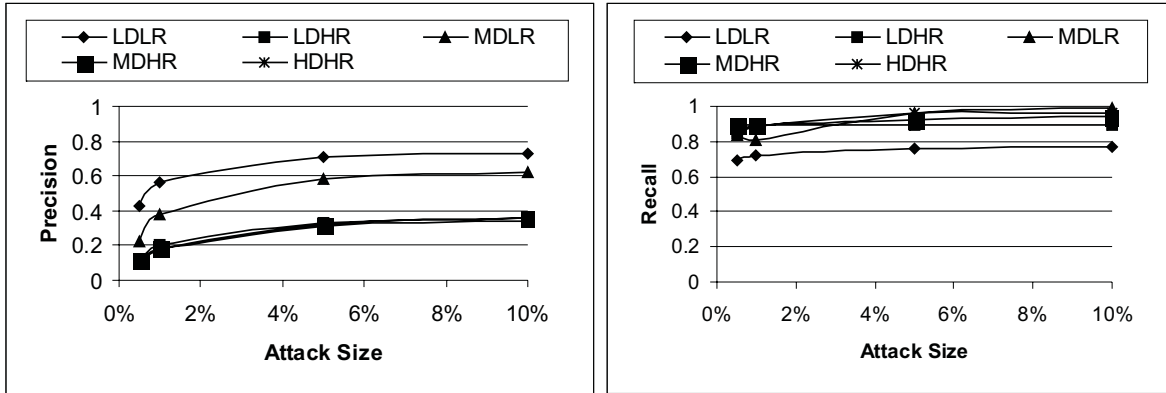


Figure 10: The Precision and Recall graphs for all categories, varying nuke attack sizes using time series algorithm (α set to .05)

combination of data would greatly improve a system’s ability to triangulate on the most suspicious profiles. This technique could be combined with profile classification to further weight the suspicion of profiles that contributed to an item during a suspected attack interval. For very large datasets with far more users than items, profile analysis is likely to be resource intensive; thus it is easy to see the benefit of being able to narrow the focus of such tasks. One of the side effects of time based detection is forcing an attacker to spread out their attack over a longer period of time in order to avoid detection.

In the experiments above, we have shown that there are some significant differences in the detection performance over different groups of items based on their rating density and their average ratings. In particular, with the techniques described above, the items that seem most likely to be the target of a push attack (LDLR, LDHR, MDLR) are effectively detected at even low attack sizes. For nuke attacks the detection of likely targets (LDHR, MDHR) is also fairly robust. Across the SPC detection schemes, detection was weakest for the HDHR group. However, as these items are the most densely rated, they are inherently the most robust to injected bias.

The performance evaluation indicated that the interval-based approach generally fairs well in comparison to SPC, even at lower attack sizes for detecting attacks. But precision was much lower in interval-based approach than in SPC. However, these two types of algorithms may be useful in different contexts. The interval-based method focuses on the trends rather than the absolute rating values during specific snapshots. Thus, it is useful for monitoring newly added items for which we expect high variance in the future rating distributions. On the other hand the SPC method works well for items that have well established distribution for which significant changes in a short time interval may be better indicators of a possible attack. As noted ear-

lier, however, we do not foresee these algorithms to be used alone for attack detection. Rather, they should be used together with other approaches to detection, such as profile classification, in the context of comprehensive detection framework.

Conclusions

Previous research on detecting profile injection attacks has focused primarily on the identification and classification of malicious profiles. In this paper we have focused on an alternate approach to this problem by detecting items with suspicious trends. We investigated the effectiveness of three statistical anomaly detection techniques in identifying rating patterns that typically result from profile injection attacks in collaborative filtering recommender systems. We show that anomaly detection techniques can be effective at detecting both items under attack and the time intervals associated with the attack for both push and nuke attacks with high recall values. Our results empirically show that detection performance varies for different categories using these methods and the most vulnerable items are also the most robust with these schemes. In our experiments we have focused on detecting anomalies in shifts in average rating, in the future we plan to explore the effectiveness of other indicators of suspicious trends such as changes in variance and rating frequency.

In addition to the direct benefits described above, all three of these techniques offer a crucial difference to profile classification alone; they are profile independent. Profile classification is fundamentally based on detecting profile traits researchers consider suspicious. While profiles that exhibit these traits might be the most damaging, there are likely ways to deviate from these patterns and still inject bias. Unlike traditional classification problems where patterns are observed and learned, in this context there is a competitive aspect since attackers are likely to actively look for ways to beat the classifier. Given this dynamic, detection schemes that

combine multiple detection techniques that examine different aspects of the collaborative data are likely to offer significant advantages in robustness over schemes that rely on a single aspect. We envision combining the techniques outlined in this paper with other detection techniques to create a comprehensive detection framework.

References

- Burke, R.; Mobasher, B.; Zabicki, R.; and Bhaumik, R. 2005. Identifying attack models for secure recommendation. In *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*.
- Burke, R.; Mobasher, B.; Williams, C.; and Bhaumik, R. 2006. Detecting profile injection attacks in collaborative recommender systems. In *Proceedings of the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2006)*.
- Burke, R.; Mobasher, B.; and Bhaumik, R. 2005. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*.
- Chirita, P.-A.; Nejdl, W.; and Zamfir, C. 2005. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, 67–74. New York, NY, USA: ACM Press.
- Lam, S., and Riedl, J. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International WWW Conference*.
- O'Mahony, M.; Hurley, N.; Kushmerick, N.; and Silvestre, G. 2004. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology* 4(4):344–377.
- O'Mahony, M.; Hurley, N.; and Silvestre, G. 2004. Utility-based neighbourhood formation for efficient and robust collaborative filtering. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC 04)*, 260–261.
- Ott, R. L. 1992. *An Introduction to Statistical Methods and Data Analysis*. Duxbury.
- Shewart, W. A. 1931. *Economic Control of Quality of manufactured Product*. Van Nostrand.
- Su, X.; Zeng, H.; and Z.Chen. 2005. Finding group shilling in recommendation system. In *WWW 05 Proceedings of the 14th international conference on World Wide Web*.