Unsupervised strategies for shilling detection and robust collaborative filtering

Article *in* User Modeling and User-Adapted Interaction · February 2009

DOI: 10.1007/s11257-008-9050-4 · Source: DBLP

2 authors:

Bhaskar Mehta
Google Inc.
31 PUBLICATIONS **873** CITATIONS
SEE PROFILE

Wolfgang Nejdl
Forschungszentrum L3S
582 PUBLICATIONS **15,701** CITATIONS
SEE PROFILE

User Model User-Adap Inter (2009) 19:65–97
DOI 10.1007/s11257-008-9050-4

ORIGINAL PAPER

Unsupervised strategies for shilling detection and robust collaborative filtering

Bhaskar Mehta · Wolfgang Nejdl

Received: 19 February 2007 / Accepted in revised form: 21 May 2008 / Published online: 18 July 2008

Abstract Collaborative filtering systems are essentially social systems which base their recommendation on the judgment of a large number of people. However, like other social systems, they are also vulnerable to manipulation by malicious social elements. *Lies and Propaganda* may be spread by a malicious user who may have an interest in promoting an item, or downplaying the popularity of another one. By doing this systematically, with either multiple identities, or by involving more people, malicious user votes and profiles can be injected into a collaborative recommender system. This can significantly affect the robustness of a system or algorithm, as has been studied in previous work. While current detection algorithms are able to use certain characteristics of shilling profiles to detect them, they suffer from low precision, and require a large amount of training data. In this work, we provide an in-depth analysis of shilling profiles and describe new approaches to detect malicious collaborative filtering profiles. In particular, we exploit the similarity structure in shilling user profiles to separate them from normal user profiles using unsupervised dimensionality reduction. We present two detection algorithms; one based on PCA, while the other uses PLSA. Experimental results show a much improved detection precision over existing methods without the usage of additional training time required for supervised approaches. Finally, we present a novel and highly effective robust collaborative filtering algorithm which uses ideas presented in the detection algorithms using principal component analysis.

B. Mehta (✉)
Google Inc., Brandschekenstrasse 110, Zurich 8004, Switzerland
e-mail: bmehta@google.com

W. Nejdl
Forschungszentrum L3S, University of Hannover, Hannover 30167, Germany
e-mail: nejdl@L3S.de

Keywords Shilling · Collaborative filtering · Dimensionality reduction · PCA · PLSA · Robust statistics

1 Introduction

Collaborative filtering technology is being widely used on the web as an approach to information filtering and recommendation by commercial service providers like *Amazon* and *Yahoo!*. For multimedia data like music and video, where pure content based recommendations perform poorly, collaborative filtering is the most viable and effective solution, and is heavily used by providers like *YouTube* and *Yahoo!* Launchcast. For malicious attackers, or a group interested in popularizing their product, there is an incentive in biasing the collaborative filtering technology to their advantage. Such activity is similar in nature to spam observed widely on the web, e.g. link farms for search engine manipulation. Since collaborative filtering is based on social networking, it is also vulnerable to social attacks, i.e. a group of users working together to bias the system. A lot of electronic systems, especially web-enabled ones provide free access to users via a simple registration processes. This can be exploited by attackers to create multiple identities for the *same* system and insert ratings in a manner that manipulates the system. *Profile injection attacks* add a few profiles (say 1–3% of the total profiles) which need to be identified and protected against. Such attacks have been refered to as *shilling* attacks, and attackers as *shillers*. Since user profiles of shillers looks very similar to an authetic user, it is a difficult task to correctly identify shilling attacks. Further, profile injection attacks can be classified in two basic categories: inserting malicious profiles which rate a particular item highly, are called *push* attacks, while inserting malicious profiles aimed at downgrading the popularity of an item are called *nuke* attacks (O'Mahony et al. 2004).

Current techniques in shilling detection are based on characteristic properties of shillers; however, the detection accuracy has been sub-optimal. In particular, by looking at individual users and mostly ignoring the combined effect of such malicious users, current detection algorithms have low accuracy in detecting shilling profiles. In this work, we provide an in-depth analysis of shilling profiles and describe new approaches to detect malicious Collaborative filtering profiles. We provide an unsupervised algorithm which is highly accurate and fast. We also look in depth at properties of shilling profiles, and provide analytical support for optimal shilling strategies which use item means. Finally, we propose a robust collaborative filtering algorithm which is more stable to shilling attacks from the ground up.

2 Shilling in collaborative filtering

In collaborative filtering, users tend to provide public feedback to a certain set of items in the form of numerical votes. The pattern of user votes is aggregated to find out most popular items, or more generally, use a subset of users similar to each other for generating recommendations. In this scenario, it is not possible to send information to end users, and lists of recommended items are automatically generated. Therefore the only form of influencing users in collaborative filtering is manipulation of the underlying algorithm to bias the list of recommended items. This is done by insertion of user profiles which appear normal, but are malicious in intent. We consider this form of manipulation similar to email, web and blog spam. Previous researchers have used the term *shilling* (Lam and Riedl 2004), which has its basis in 20th century vocabulary.[1] Burke et al. have used the phrase *profile injection attacks* to refer to this phenomena (cf. Burke et al. 2006). This clearly emphasizes the fact that user data is added to a recommender system in the form of multiple *user profiles* created by a malicious group of users.

Burke et al. also describe various *models* for generating attack profiles for shilling. We follow the profile models in this work for generating shilling profiles to be added to the collaborative filtering system. Various attack strategies have been discussed in literature (see Burke et al. 2006 for details). These attacks strategies include:

1. *Random attacks*: where a subset of items is rated randomly around the overall mean vote,
2. *Average attacks*: where a subset of items is rated randomly around the mean vote of every item,
3. *Bandwagon attacks*: where a subset of items is rated randomly around the overall mean vote, and some highly popular items are rated with the maximum vote.

Note that Gaussian distributions $\mathcal{N}(\mu, \sigma)$ have been used for generating most of the random votes rather than the uniform random distribution. This implies that attack profiles have votes near, or equal to the mean vote with a very high probability. Also, standard deviation of the complete user dataset (all votes) is used for random and bandwagon attacks, while the standard deviation of the each individual item is used for the average attack.

Formally, a shilling profile can be said to consist of three parts:

1. *Target item*: One item is chosen in the shilling profile and assigned the highest vote for a push attack, or the minimum for a nuke attack. Usually a set of shilling profiles have a common target item for the attack to be effective.
2. *Filler items*: The set of items which are voted for in the spam profile; the above cited attack models are used to generate these votes. Typically these values are generated by a random Gaussian generator $\mathcal{N}(0, 1)$and then scaled to the voting range, with a mean μ and variance σ. Depending on the information available, the mean and variance can be varied for different items for higher impact, or kept the same for a low knowledge attack. The filler size is measured as a percentage of the item set. Typical values for filler size range from 1% to 10%.
3. *Non voted items*: The remaining unrated items form the majority of the profile (Fig. 1).

One more term used in the literature and in this paper refers to the size of the attack; this is known as *attack size* and is measured as a percentage of the total user population. The higher the attack size, the more effective the attack is. However, attack sizes tend to be small, because creating profiles comes with a (human) cost, and therefore only

[1] The Wikipedia entry on Shilling at http://en.wikipedia.org/wiki/Shill provides a brief history of the term.

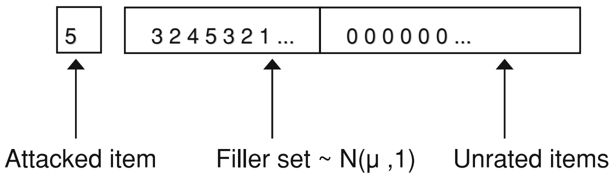

Fig. 1 Typical structure of a shilling profile which participates in a push attack

a few profiles can be inserted by one set of people. Typical values range from 1% to 10%. Smaller attacks are also very effective, and often more difficult to detect.

Obfuscation of shilling profiles: To make spam less detectable, researchers have proposed strategies to make the signatures of spam less prominent. Three strategies have been proposed in (Williams et al. 2006) which we also refer to in our experiments:

1. *Random noise*: Add random noise to each generated vote.
2. *User shift*: Add the same random noise each vote of the same user.
3. *Target shifting*: Instead of using the highest vote for the recommended items, randomly use the next highest vote (e.g. using 4, instead of 5).

3 Related work

Research in the area of shilling attacks has made significant advances in the last couple of years. Early work identified the threat of shilling attacks which were classified into two basic types: *nuke* and *push*. Various attack strategies were then discovered and appropriate metrics were developed to measure the effectiveness of an attack. In the previous section, we mentioned various attack models for generating shilling profiles (Random, Average and Bandwagon attacks).

The most commonly used metric for measuring the effect of shilling attacks is *prediction shift*, which models the difference between average predicted rate of the targeted item, after and before the attack. This metric is defined as follows: suppose an item y has a predicted rating of $v_{u,y}$ for user u. If this predicted rating changes to $\hat{v}_{u,y}$ after a shilling attack, this is the shift in prediction for user u. The total prediction shift is thus:

$$\mathcal{P} = \sum_{u} |\hat{v}_{u,y} - v_{u,y}| \tag{1}$$

Using this metric, it is reported that average attacks are more powerful than random attacks. Further, it has also been shown that k-NN based algorithms for collaborative filtering (e.g. based on Pearson's correlation) were very vulnerable towards such shilling attacks. The second phase of research in this area aimed at finding solutions to the newly discovered threat. It was discovered that item based recommendation algorithms, which measure similarity between items rather than users (Sarwar et al. 2001; Burke et al. 2006) were more robust to such manipulations. However, newer attack models like bandwagon attacks and segment attacks have been shown as successful even against item-based recommendation algorithms.

The earliest shilling detection algorithm based on features of shilling profiles was reported in (Chirita et al. 2005). While this algorithm was successful in detecting shilling attacks with dense attacker profiles, it was unsuccessful against attacks, which are small in size or have high sparsity. Burke et al. (2006) compare their feature-based classification algorithm which performs significantly better than the Chirita et al. algorithm by taking more features into account. The Burke et al. algorithm trains a classifier given enough example attack profiles and authentic profiles and is fairly accurate in detecting shilling attacks of varying sizes and density. Two disadvantages of their approach come to mind: firstly, a supervised approach needs a large number of examples, and can detect only profiles similar to the examples profiles. Secondly, these algorithms perform badly when the shilling profiles are obfuscated. Adding noise, shifting targets, or shifting all user ratings differently makes the attack profiles more difficult to detect for existing feature based detection algorithms. Williams et al. (2006) discusses these obfuscation strategies and their effect on detection precision. Recent algorithms (O'Mahony et al. 2006) have taken up a more principled approach where signal processing theory is used to detect noise which is artificially inserted into the data. We find this direction promising, however the accuracy remains low (15–25%). One principle reason for the low accuracy of this and other approaches that all existing algorithms consider users indivually, instead of looking at the collective effect.

A recent line of work discusses the robustness of several algorithms for collaborative filtering. In 2006, (Mobasher et al. 2006) published their results on stability of k-means and PLSA recommendation algorithms as compared to the k-NN recommendation algorithm. It was shown that model based algorithms for CF are more stable to noisy or malicious neighbors. Recent work (Mehta et al. 2007b; Sandvig et al. 2007) has proposed two more algorithms for robust collaborative filtering: the robust matrix factorization (RMF) algorithm from Mehta et al. is not much more stable than modified forms of SVD, though it is highly resistant to noise. The use of association rules by Sandvig et al. is very effective in stopping the prediction of potentially attacked algorithms. However, the coverage of the prediction algorithm is around 50%, which means that only half the missing ratings can be reliably predicted, thus resulting in low accuracy. We discuss a robust collaborative filtering approach which is more stable than previously reported work.

To summarize, the current state with respect to shilling detection is that feature-based classifiers have been proposed and highly discriminative features have been devised. These algorithms work by correctly identifying the goal of shillers to affect the predicted vote for the attacked item, and identifying profiles which affect the predicted vote significantly. However, shilling profiles can now be constructed such that the obvious signatures which make them stand out are masked, at the cost of lesser impact. Current algorithms fail in detecting obfuscated (Williams et al. 2006) and small attacks and have to evolve in order to win this game. Our detection algorithm provides the next step by correctly detecting even obfuscated attacks with a high accuracy by looking at the combined effect of spam rather than individual users.

4 Characteristics of shilling attack profiles

Existing shilling detection algorithms exploit specific properties of profiles injected in order to identify them.[2] After reverse engineering the profile signatures, appropriate heuristics are used to capture information which characterizes shillers. This is similar to a 2-player game, where a move is made by one player maximizing a certain objective, the counter move by the second player maximizes his/her objective, which are adversarial in nature, and changes the state of the game significantly. In order to understand why shilling detection algorithms work, or don't work, one needs to understand the goals of shillers and methods used to achieve them.

The primary objective of shilling profiles is to maximize (or minimize in the case of nuke attacks) the predicted value for the chosen item for the largest possible number of users. This can be achieved in the following ways: *firstly*, by constructing profiles which are moderately correlated to a large number of users in order to affect them; and *secondly*, by constructing profiles which are highly correlated to a fraction of the users and affect them significantly. In order to achieve these objectives, profiles have to be constructed in a special manner. Most attack strategies involve rating items around the mean vote, which minimizes the deviation from other existing votes, except the item under attack. Various studies in the effect of shilling attacks have been reported in the literature; the results of these studies show that the impact of well constructed profiles can be huge. Even if 1% of all users are shillers, the system can be skewed and an attacked item maybe pushed to top of the recommendation list of many users. Such attacks are especially severe for items without many votes, where shillers can easily become the authoritative users and force higher ratings.

The specific construction of shilling profiles also have interesting properties, some of which are used by detection algorithms. We describe several important properties below, often providing empirical results of specific experiments relevant to a particular property. All data and plots in this section are generated using the MovieLens dataset (Riedl et al. 1998) with 100,000 votes, with 944 users and 1682 movie-items.

1. *Low deviation from mean votes value, but high deviation from the mean for the attacked item*: RDMA (Rating deviation from Mean Agreement) and WDA (Weighted Degree of Agreement) are statistical measures which are based on this idea. These metrics capture am important characteristics of shilling profiles, i.e. they have votes very close to item means, as opposed to normal users who shows a higher deviation.
2. *High similarity with large number of users*: Shillers have a high correlation with a significant number of users due to the mean-like votes for most items. A direct result of being highly correlated with a user is that a shiller becomes an authoritative neighbor and figures prominently in the set of k-nearest neighbors. Figure 2 shows the number of users that each user is k-nearest neighbor for, after injecting 3% shilling profiles. This amounts to 30 attacks profiles inserted into the database of 944 users. Notice that shillers are neighbors for a much larger number of users, in comparison to authentic users. Table 1 also shows the authoritativeness of shillers

[2] Some systems simply look at usage patterns like the rate of votes over a time period.

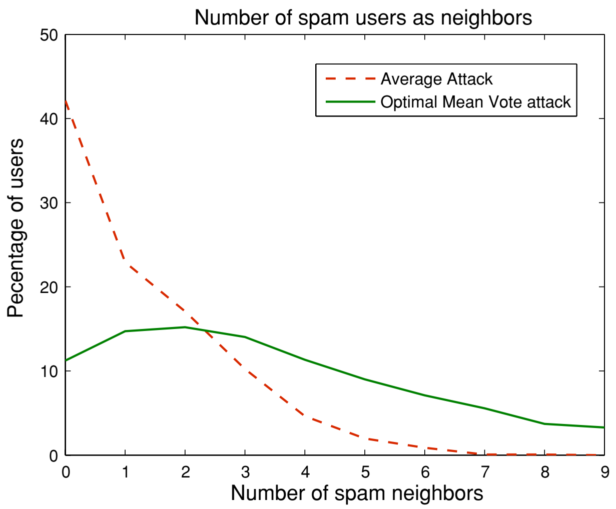

Fig. 2 No. of shillers in top-20 neighbors for every user. Notice that more than 35% users have at least 2 spam neighbors for an average spam attack (7% filler, 5% attack size) vs. 69% for an optimal spam attack using exact mean votes

Table 1 Number of top-20 neighborhoods that each user belongs to

Neighbors	0–20	20–40	40–60	60–80	80–100	100–120
Normal	818	253	62	15	19	9
Spam	0	1	10	13	17	9

as compared to normal users; more than 50% of shillers added by a 5% average attack influence atleast 80 normal users.

3. *Shillers work together*: A large fraction of top-20 neighbors for all users are spam for a well constructed attack. This means that shillers magnify each other's effect and together push the attacked item to a significantly higher rating. While this is an important characteristic, very few algorithms have used this for classification (e.g. *Target Model Focus* in Burke et al. 2006). Figure 2 shows that after a spam attack, the top-20 neighbors of every user are full of shillers. According to our estimates, with a small attack of 3% shilling profiles, approximately 15% of every user's closest neighbors are spam shillers.
4. *Shillers are highly correlated*: Shillers tend to have very high correlation coefficients($>$0.9) due to the same underlying model which is used to generate them. Average and random attacks have been observed to have this characteristic; Chirita et al. have also used this characteristic to construct a metric called *DegSim*, which captures the average similarity for top-*k* neighbors. Recent work has pointed out some obfuscation strategies that can be used to decrease the average similarity and make the shillers less noticeable. Figure 3 demonstrate these properties for a bandwagon attack.

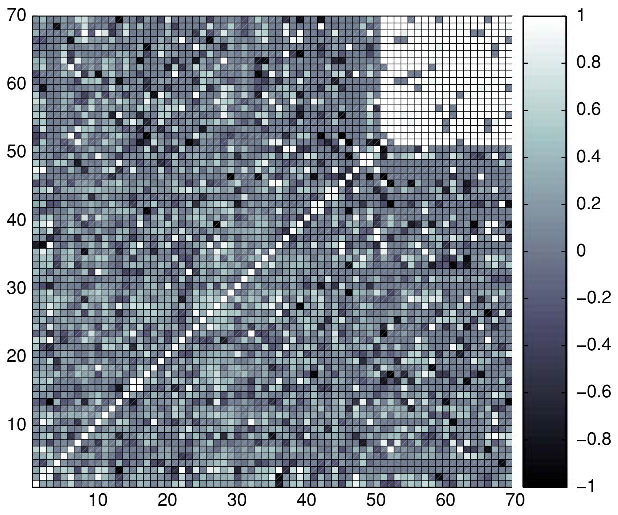

Fig. 3 Shillers are highly correlated: 50 authentic profiles and 20 shilling profiles are used for calculating the Pearson's Correlation coefficient Notice how shillers exhibit a noticeably higher degree of correlation

5 Optimal shilling strategy for a single shiller

In this section, we discuss what the optimal strategy for a shiller should be while constructing a shilling profile. Assume that the end system $\mathcal{S}$ has n users and m items. We use the notation $v_{i,y}$ for the vote give to an item y by a user i, and $\hat{v_i}$ denotes the average vote of a user i. $C_{i,j}$ is the Pearson's correlation coefficient between user i and j.

The Pearson's correlation coefficient is calculated according to the following equation:

$$C_{i,j} = \frac{\sum_y (v_{i,y} - \hat{v_i})(v_{j,y} - \hat{v_j})}{\sqrt{\sum_y (v_{i,y} - \hat{v_i})^2}\sqrt{\sum_y (v_{j,y} - \hat{v_j})^2}} \tag{2}$$

We assume that the system provides recommendation using user-user similarity based collaborative filtering; we further use Pearson's similarity measure. In this scheme, a user's vote on an unknown/unrated item is calculated based on the votes of other users who are similar to the current user. In a general scheme, it is also possible to use all users, and weight their opinion with their similarity to the current user. Formally, the predicted vote for user u_i for an item y can be expressed as

$$v_{i,y} = \hat{v}_i + \frac{\sum_j C_{i,j}(v_{j,y} - \hat{v_j})}{\sum_j |C_{i,j}|} \tag{3}$$

Note that the correlation coefficient is measured only over items that two users have commonly voted on.

Let us add a shiller s to the user set $\mathcal{U}$. This shiller wishes to cause item $\mathcal{I}$ to be recommended more often. The strategy to do this is to change the predicted value of the item $\mathcal{I}$ for as many users as possible. An effective attack would make this value as high as possible. *Prediction shift* is a measure used in literature to measure how effective an attack is. It is defined as the difference in the predicted value of an item before and after a spam attack.

$$\mathcal{P} = \sum_u \grave{v}_{i,y} - v_{i,y} = \sum_u \mathcal{P}_u, \tag{4}$$

where $\grave{v}_{i,y}$ denotes the predicted value of item y for user u_i after a spam attack. $\mathcal{P}_u$ denotes the prediction shift in user u.

Thus the aim of the shiller s is to maximize the prediction shift $\mathcal{P}$. We now make some simplifying assumptions for the analysis: we assume that there is only one shiller present, and that the recommendation is being calculated using all the users, and not just the top-k neighbors. The prediction shift $\mathcal{P}_u$ can be written as:

$$\mathcal{P}_u = \grave{v}_{i,y} - v_{i,y} = \frac{\sum_j C_{i,j}(v_{j,y} - \hat{v_j}) + C_{i,s}(v_{max} - \hat{v}_s)}{\sum_j |C_{i,j}| + |C_{i,s}|} - \frac{\sum_j C_{i,j}(v_{j,y} - \hat{v_j})}{\sum_j |C_{i,j}|}$$

The total prediction shift is thus: $\mathcal{P} = \sum_u \mathcal{P}_u$. Clearly, $\mathcal{P}$ is maximized when each of the respective prediction shifts $\mathcal{P}_u$ are maximized.

$$\begin{aligned}\mathcal{P}_u = \grave{v}_{i,y} - v_{i,y} &= \frac{\sum_j C_{i,j}(v_{j,y} - \hat{v_j}) + C_{i,s}(v_{max} - \hat{v}_s)}{\sum_j |C_{i,j}| + |C_{i,s}|} - const \\ &= \frac{\kappa_1 + \kappa_2 x}{\kappa_3 + |x|} - const\end{aligned}$$

where $\mathcal{P}_u$ has been written as a function of the correlation coefficient $C_{i,s}$ (replacing $C_{i,s}$ by x). Clearly, the attacked item is rated as v_{max} (the maximum allowed rating) by the shiller to have the maximum deviation. Thus κ_2 can be assumed to be a constant. Note that the correlation coefficient lies in $[-1, 1]$. Clearly, for x in $[-1, 0)$, the prediction shift is negative, implying that the correlation coefficient should be positive. Since the derivative $\frac{\partial P_i}{\partial x}$ of P_i is positive everywhere in [0,1], P_i is a strictly increasing function. Thus the maximum value of P_i is reached at $x = 1$.

In general, the overall prediction shift is maximized if the correlation coefficient of the spam profile is maximized with all the user profiles. If the neighborhood of every user is also limited to a fixed size, then clearly, the impact of the spam profile is maximum if the shiller is a part of these neighborhood. Since the neighbors are formed based on the Pearson's correlation, maximizing the correlation with all users is the primary objective.

5.1 Strategy to maximize correlation with maximum users

The above analysis shows clearly that a spam profile must be constructed to maximize correlation with the maximum number of users. Here, we try to motivate the use of mean item votes for maximizing the correlation coefficient.

We use Canonical Correlation Analysis (CCA) to analyze the optimal strategy: canonical correlation analysis seeks vectors a and b such that the random variables $a'X$ and $b'S$ maximize the correlation $\rho = cor(a'X, b'S)$.

Let us construct a random variable $X = (X_1, \ldots, X_n)'$ where X_i represents the user i. Let S represent the shiller's profile. We would like to maximize the correlation between Y and X with the additional constraints that all users are given equal weight. This leads us to use $a = (\frac{1}{n}, \ldots, \frac{1}{n})$. Trivially, $b = 1$. Note that $a'X$ leads to the average of the user profiles (we represent the mean vote of an item y_i by $\hat{y_i}$).

$$a'X = \sum_i \frac{1}{n} X_i = \hat{X}(\hat{y_1}, \ldots, \hat{y_m}) \tag{5}$$

The expression to maximize now is:

$$\rho = \frac{cov(\hat{X}, S)}{\sqrt{var(\hat{X})var(S)}} = \frac{\sum_i (\hat{y_i} - \hat{u})(s_i - \hat{s})}{\sqrt{\sum_i (\hat{y_i} - \hat{u})^2}\sqrt{\sum_i (s_i - \hat{s})^2}}$$

where $\hat{y_i}$ represents the average vote for an item i. and $\hat{u}$ denotes the overall average. It is easy to see that placing $s_i = \hat{y_i}$ maximizes the above expression to make $\rho = 1$; simple vector calculus also produces the same result on replacing the sum with dot-products of appropriately dimensioned vectors. This implies that the optimal strategy for maximizing correlation with all users is to use mean votes for individual items. Attack generation models discussed previously also use this idea for filler votes with the addition of Gaussian noise to make the profiles more varied. Note that attacking an item y_i requires placing the maximum vote for this item; however this does not significantly effect the correlation with other users, since the other votes are still based around the item mean (Fig. 4).

We note here that though the assumptions have made the analysis of optimality simplistic, it is still illustrative of average attacks as begin stronger than other attack types. We also note that there are different *optimal* attacks for different recommendation algorithms, since the function to be maximized will vary. While the above analysis can be justly criticized as being specific to Pearson's based CF, it is still useful for explaining some experimental observations made by previous work, which were not explained in a principled fashion.

6 Shilling detection using soft clustering

Current feature based algorithms tend to pick users with the maximum impact in terms of the measures/features used. However authentic users who are authoritative

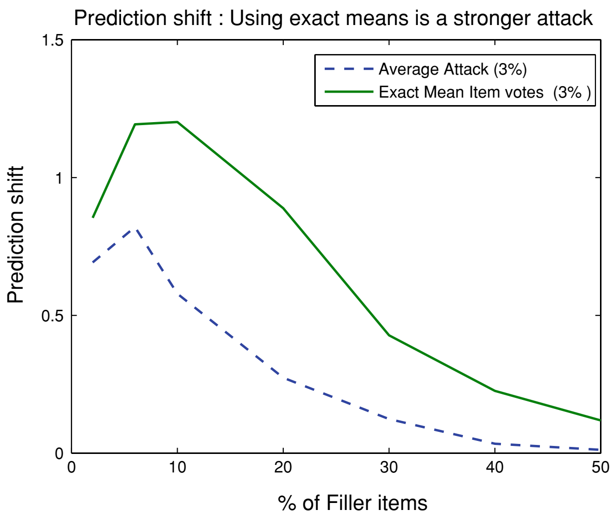

Fig. 4 Prediction shift as a function of filler size measured over the complete user population(excluding shillers). Notice how using exact means has a much stronger prediction shift as compared to an average attack. The algorithm for which prediction shift is being measured is the *k*-nearest neighbors algorithm using Pearson's similarity

and different from many other users can also show significant impact and be falsely classified. Importantly, by working in groups, the effect of shillers is large only in groups, and individual shilling profiles can be undetectable especially when in small numbers. Thus it makes sense to eliminate clusters of shillers, rather than individual shillers. In this and the next section, we outline two algorithms based on this intuition: PLSA is a mixture model which computes a probabilistic distribution over communities (clusters of users) based on latent factors and has been reported to be robust to shilling (Mobasher et al. 2006); PCA is a linear dimensionality reduction model which can be used to select dimensions which are very different, or as in this work, very similar to other dimensions.

6.1 Probabilistic latent semantic analysis

Probabilistic latent semantic analysis (PLSA) is a well known approach for text analysis and indexing used to discover hidden relationships between data (Hofmann 2004); it has also been extended to collaborative filtering. PLSA enables the learning of a compact probabilistic model which captures the hidden dependencies amongst users and items. It is a Bayesian network where latent variables are used to render users and items conditionally independent. The hidden variables can be interpreted as a probability distribution over communities of users or clusters; each user is allowed to be a part of multiple clusters, with a certain probability. The patterns in data along with the model fitting algorithm ensure that the learnt distribution minimizes the log-likelihood of the data.

PLSA is a probabilistic variant of latent semantic analysis(LSA), which is an approach to identify hidden semantic associations from co-occurrence data. The core of PLSA is a latent variable model(also known as the aspect model) for general co-occurrence data which associates a hidden variable $\mathbf{z} \in Z = \{z_1, z_2, \ldots, z_K\}$ with each observation. In the context of collaborative filtering, each observation corresponds to a vote by a user to a item. The space of observations is normally represented as an $M \times N$ co-occurrence matrix (in our case, of M items $\mathcal{Y} = \{y_1, y_2, \ldots, y_M\}$ and N users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$. The aspect model can be described as a generative model:

- Select a data item y from $\mathcal{Y}$ with probability $P(y)$,
- Pick a latent factor $\mathbf{z}$ with probability $P(z|y)$,
- Generate a data item u from $\mathcal{U}$ with probability $P(u|z)$.

As a result we obtain an observed pair (u, y), while the latent factor variable z is discarded. Translating this process into a joint probability model results in the following:

$$P(u, y|z) = \sum_z P(u, y, z) = \sum_z P(y|z)P(z|u)P(u) \tag{6}$$

This model is based on two independence assumptions: first, observation pairs (u, y) are assumed to be generated independently; secondly, conditioned on the latent factor z, data item p_j is assumed to be generated independently of the specified item y. Since in collaborative filtering we are usually interested in predicting the vote for an item for a given user, we are interested in the following conditional model:

$$P(y|u, z) = \sum_z P(y|z)P(z|u) \tag{7}$$

6.2 Soft clustering using PLSA

While accuracy has been a well known advantage of PLSA, recent studies have also concluded that PLSA is a very robust CF algorithm, and is highly stable in the face of shilling attacks. (Mobasher et al. 2006) indicates that the prediction shift for PLSA is much lower than similarity based approaches. However, a clear explanation for this has not been provided so far. We investigated the reasons for PLSA's robustness over many experiments and observed the model to understand the mechanisms. The intuition is that PLSA leads to clusters of users (and items) which are used to compute predictions, rather than directly computing neighbors. However this intuition is challenged by experimental results using a k-means clustering algorithm in the same work. Clearly, shilling profiles deceive clustering algorithms due to their high similarity with normal users.

PLSA is a mixture model where each data vector has its own distribution. Membership to a distribution is however not constrained; a data point can belong (probabilistically) to many distributions, with combination weights chosen so that the observed ratings are explained best. This results in *soft clustering* where a data point can lie in multiple clusters. We posit that this is also the reason why shilling is less effective

Table 2 A run of PLSA based shilling detection for 200 shillers on a dataset with 5000 users

Cluster	Mahalanobis distance	# real	# spam	Cluster	Mahalanobis distance	# real	# spam
1	707.63	222	1	11	426.36	295	0
2	568.06	250	0	12	575.37	237	0
3	372.31	302	0	13	1195.50	169	0
4	391.40	302	0	14	751.19	212	0
5	273.52	352	0	15	576.48	243	0
6	225.16	240	199	16	441.41	275	0
7	894.96	195	0	17	490.81	265	0
8	348.50	309	0	18	619.65	235	0
9	1156.85	173	0	19	535.85	202	0
10	713.30	217	0	20	539.10	253	0

'Real' represents the number of real users in the cluster, and 'spam' represents the number of shillers

against PLSA: shillers are close to many users, but often dominant in one cluster due to their extraordinary similarity. Since user ratings are more noisy than shilling profiles, the likelihood of user ratings being explained by shilling profiles is limited, though not minuscule. This explanation has also been verified experimentally: We learn an model of an EachMovie data-subset with 5000 users to which 200 shilling profiles are added. On learning a PLSA model with *20* communities, we select the dominant community for each user. On analysis, we notice that all the shillers are in either one or two communities. By correctly identifying this community, we can isolate the shillers and remove them. Table 2 shows that this line of reasoning is correct and experimentally verified.

Identifying the community to be removed is vital: noticing how the profiles are close to each other, we have to identify a measure which examines how closely knit a community is: one possibility is to use Mahalanobis distance, which is traditionally used to identify outliers in multivariate data (Mahalanobis 1936). We suggest using the average Mahalanobis distance of a community as follows: for each community $\mathcal{C}$ which is set of users, we find the Mahalanobis distance d_u of each user u as

$$d_u = \sqrt{(u - \overline{u})\mathbf{C}_0^{-1}(u - \overline{u})^T} \ , \tag{8}$$

where the matrix $\mathbf{C}_0$ is the covariance matrix of the community $\mathcal{C}$, and $\overline{u}$ is the mean profile over the same. Notice how $d_u > 0$ since $\mathbf{C}_0$ is positive semi definite. We measure the 'closeness' of the community $\mathcal{C}$ by the average Mahalanobis distance over the user-set of $\mathcal{C}$. The intuition is that the cluster containing shilling profiles will be tighter, leading to lower average distances from the centroid of the cluster.

Initial implementation showed that computing Mahalanobis distances is very time consuming due to the inversion of large covariance matrices. To get around this, we observe that a fixed Mahalanobis distance defines a hyper-ellipsoid which is scaled by the variance in observed data in a direction. If variances are assumed to be one, *Mahalanobis* distance reduces to *Euclidean* distance. Based in this observation, we use z-scores[3] instead of actual discrete votes to find the closeness of a cluster, and thus use the simpler Euclidean distance measure:

$$d_u = \sqrt{(u - \overline{u})(u - \overline{u})^T}, \tag{10}$$

Experimental results (not mentioned in this paper) have showed that these two measures correlate very well if using z-scores.

Algorithm 1 PLSASelectUsers (**D**)

```
1: $\mathbf{D}$ ←z-scores($\mathbf{D}$)
2: Train a PLSA model for $\mathbf{D}$.
3: for all Users $u \in \mathbf{D}$ do
4:     $Comm_u = k$ where $P(z_k|u)$ is maximum
5: end for
6: for all Community $k$ do
7:     $U_k$ ← The set of users $u$ with $Comm_u = k$
8:     $Distance(k) \leftarrow \frac{1}{n}\sum_{u \in U_k}(u - \overline{U_k})^2$
9: end for
10: Return $r$ users with smallest $Distance$ values
```

Output: Return U_k with smallest $Distance$ value

7 PCA for shilling detection

Currently, shilling detection algorithms are feature-based and depend on features with high discrimination between true users and shillers. Such features typically measure the impact of a user; however authentic users who are authoritative and different from many other users can also be falsely classified. Our hypothesis is that the effect of shillers is large only in groups, and individual shilling profiles can be undetectable, especially when in small numbers. Thus it makes sense to eliminate clusters of shillers, rather than individual shillers.

Clustering in such a domain is not without problems. The pattern exhibited by normal users is non-uniform and several clusters could potentially be identified; we can at best hope to discover one cluster of highly correlated shillers, as we saw in the previous method using PLSA. However, an ideal method should be able to rank users in a cumulative similarity with other users; suspicious user profiles which occur at the top of such a list are likely to be shillers with high probability. These patterns should be possible to extract from the correlation matrix of all users.

Similar problems have arisen in the area of *Gene Clustering* where it is important to identity highly correlated genes from experimental data of patients. The Gene Shaving

[3] *z-scores* can be computed for a user u for a item y, where the user has voted $v_{u,y}$, by using the equation

$$z_{u,y} = \frac{v_{u,y} - \hat{v}_u}{\sigma_u}, \tag{9}$$

technique (Hastie et al. 2000) is such a technique in use for clustering highly correlated genes from a set of observed data, which includes missing values. Gene shaving uses a multivariate analysis technique called Principal Component Analysis (PCA) (Jolliffe 2002). PCA is used to find the intrinsic dimensionality of high dimensional data by projecting it to a low dimensional space where the axes selected capture the maximum variation in data. Gene Shaving essentially involves performing PCA and shaving off the top few genes (say 10%) which have the lowest dot product with the principal component. While PCA does not work directly with the *correlation* matrix as desired by us (it works rather on the *covariance* matrix), data transformation like *z-scores* can be used; the covariance of normalized data from a *z*-score transformation is the same as correlation. Thus, we expect PCA to be able to identify which users have the maximum similarity to other users and return a ranked list. Below, we briefly describe PCA and then explain our selection procedure.

7.1 Overview of PCA

Principal component analysis (PCA) is the simplest, and the best (in the mean-square error sense) linear dimension reduction technique. Being based on the covariance matrix of the variables, it is a second-order method. In essence, PCA seeks to reduce the dimensionality of the data by finding a few orthogonal linear combinations (called the *Principal Components*) of the original variables with the largest variance. A principal component is a linear combination of the variables and there are as many PCs as the number of the original variables. The first principal component $s_1 = \mathbf{x}^T w_1$, where the p-dimensional coefficient vector $\mathbf{w}_1 = (w_{1,1}, \ldots, w_{1,p})^T$ solves

$$\mathbf{w}_1 = \underset{|\mathbf{w}|=1}{\text{argmax}} \ \ Var\left(\mathbf{x}^T \mathbf{w}\right), \tag{11}$$

Principally, PCA is equivalent to performing an eigen-decomposition of the *covariance* matrix of the original data. The eigen vectors of the covariance matrix constitute a projection matrix which can be used to transform the basis of data into one conforming to principal components. We now look at PCA more formally: suppose the input data is represented by the matrix $\mathbf{X}^{m \times n}$, where each column corresponds to an observation $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$. For the sake of simplification, we assume that the data is zero-centered.[4] Now the covariance matrix $\mathbf{C}$ is computed. $\mathbf{C}$ is defined as:

$$\mathbf{C} = \frac{1}{n-1} \mathbf{X} \cdot \mathbf{X}^T, \tag{12}$$

Using spectral decomposition theorem (Jolliffe 2002), we decompose the symmetric covariance matrix $\mathbf{C}$ as follows:

$$\mathbf{C} = \mathbf{U} \lambda \mathbf{U}^T, \tag{13}$$

4 If data is not zero centered, a simple linear transform can be used by deducting the mean of every column from each variable.

where λ is a diagonal matrix containing eigenvalues of $\mathbf{C}$. $\mathbf{U}$ contains the corresponding eigenvectors. It can be shown that the principal components (PCs) can be given by the rows of the matrix $\mathbf{S}$ where

$$\mathbf{S} = \mathbf{U}^T\mathbf{X}, \tag{14}$$

By ordering the rows of $\mathbf{U}$ in the order of eigenvalues of $\mathbf{C}$ (which comes from the matrix λ), we get the PCs in ascending order (i.e. the first row represents the first PC, etc.). An important property of this order is that PCs model the overall variance of data $\mathbf{X}$ in proportion of the corresponding eigenvalues.

7.2 Variable-selection using PCA

As stated earlier, we would like to exploit the high inter-correlation (within shillers) present in the correlation matrix and find a group of shillers which vary least with respect to each other. If a dataset has variables which are very similar and highly correlated, then these variables would be uninteresting for a PCA since very low information is added by these highly correlated dimensions. Formally, due to a high value of covariance between two variable, one can be safely removed and the remaining variable used to estimate the removed variable with a high degree of confidence. For multivariate data, dimensions exhibiting a high level of covariance to other dimensions (particularly extraordinary correlation with a few dimensions) will be discarded by a linear second order method like PCA. This means that the discarded dimension will not have an impact of the top principal components as a result of PCA.

We observe that the profiles of shillers are very similar. If we interpret users as variables (i.e. the dimensions of the data are the users, and the observations are the items), we have data where a number of dimensions are very similar. A closer look at our data shows us that the covariance between shillers is much higher than normal users (see Fig. 3). Covariance between normal users is observed to be much lower. This means that PCA of this dataset will compute principal components which are oriented more towards real users who exhibit the maximum variance of the data. We therefore need to select those dimensions, which show the highest covariance with all other dimensions. This amounts to selecting some variables from the original data using PCA, which is known in literature as *Variable-selection using PCA* (Jolliffe 2002).

Note that the typical aim in Variable-selection for multivariate data it to select the most independent subset of variables (Jolliffe 1972). Several approaches have been investigated to find the most representative variable corresponding to the top-n PCs; the most successful ones make this association based on the magnitude to the coefficient in the PC. Our aim is the opposite i.e. to select the least independent variable; hence we chose variables with *smallest* PC coefficients (called at the LC criteria). Note that other methods are also applicable here; we report the most successful ones later in this section.

Algorithm 2 (Mehta et al. 2007a) outlines our proposed approach for variable-selection. Note that we first center the data by substituting votes with *z-scores*. Next we transpose the data to cast users are variables and calculate the covariance matrix. Eigen decomposition of the covariance matrix yields the principal components: for every variable, each PC contains a coefficient. We choose users based on more than one PC; typically the coefficients in the first m PCs. In a practical setup, we recommend $m = 3 - 5$.

Algorithm 2 PCASelectUsers (**D**)

```
1: $\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$)
2: $\mathbf{D} \leftarrow \mathbf{D}^T$
3: $COV \leftarrow \mathbf{D}^T\mathbf{D}$
4: $\mathbf{U}\lambda\mathbf{U}^T$ = Eigen-value-Decomposition($COV$) { Get Eigenvectors of $COV$}
5: $PCA_1 \leftarrow \mathbf{U}(:, 1)$ {First Eigenvector of $COV$}
6: $PCA_2 \leftarrow \mathbf{U}(:, 2)$ {Second Eigenvector of $COV$}
7: $PCA_3 \leftarrow \mathbf{U}(:, 3)$ {Third Eigenvector of $COV$}
8: for all columnid $user$ in $\mathbf{D}$ do
9:     $Distance(user) \leftarrow PCA_1(user)^2 + PCA_2(user)^2 + PCA_3(user)^2$
10: end for
11: Sort $Distance$
```

Output: Return r users with smallest $Distance$ values

7.3 Variable-selection for large datasets

In a practical setup, a recommender system has to deal with millions of users and items. Our previous outlined setup requires a matrix multiplication for calculating the sample covariance matrix (see Eq. 11). Since we additionally invert users and items, the covariance is square in the number of users; moreover, this covariance might be as sparse as the original matrix, though it would be lower in rank.[5] As an example, we consider the NetFlix dataset containing 480,189 users. Performing large scale PCA on such a large matrix ($\sim$ 480,000 $\times$ 480,000) proves to be a daunting task.

On careful observation, we observe that $\mathbf{X}$ is sparse and the zero-centered version can be efficiently stored as a compressed row matrix. We also observe that for $\mathbf{X}^{n\times m}$, the first n eigenvalues are identical for $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^{\mathrm{T}}$.

$$\mathbf{C} = \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}\mathbf{V}\Sigma\mathbf{U}^{\mathrm{T}} = \mathbf{U}\Sigma^2\mathbf{U}^{\mathrm{T}} \tag{15}$$

As previously established, the Principal components of $\mathbf{X}$ are given by $\mathbf{S} = \mathbf{U}^T\mathbf{X}$. Thus, calculating the covariance is unnecessary; we can compute the SVD of $\mathbf{X}$ to get the loading matrix $\mathbf{U}$. Computing the SVD of a large matrix is not without challenges, especially using existing libraries. We resort to and SVD method based on Hebbian learning (Gorrell 2006). Details of the SVD methodology which has been provably successful on the NetFlix dataset are described in Mehta et al. (2007b).

[5] The rank of a covariance matrix is $Min(m, n)$, where the original matrix is $n \times m$.

Table 3 Criteria for variable-selection based on PCA

	Description
LC	Select q variables with the lowest average of absolute values of loading (1–5 dimensions sufficient)
SLC	Same at LC, but using squared loading combination, choosing lowest average of squared loading (1–5 dimensions sufficient)
QLC	Same at LC, but using exponentiated (to power 4) loading combination, choosing lowest average of exponentiated loading (1–5 dimensions sufficient)
B_2	Associate each principal component with a variable based on the maximum absolute loading, and select q variables for the last q dimensions (q dimensions required)

7.4 Finding suspected attack profiles

PCA can find a set of variables which are highly correlated, a fact exploited in the design of VarSelect. VarSelect performs Variable-selection using a selection criteria called LC (see Table 3). There are several other selection procedures discussed in the literature; Al-Kandari and Jolliffe (2005) provides a good overview of these criteria. Table 3 briefly describes various criteria. While several other criteria have been proposed, they require a complete eigen-decomposition, and/or conditional correlation matrices. As an example, consider B_2 (see Table 3): this strategy involves doing a thin SVD, for selecting k variables a k-dimension SVD has to be performed. If we want to eliminate 500 variables, the computation effort involved is very high. In contrast, strategies LC, SLC and QLC require only the first 3–5 eigenvectors. Given that our data is not suitable for complete decomposition, we omit other variable-selection methods relying on that property.

Each of the strategies described in Table 3 gives a ranked list as output; a suitable cutoff point still has to be selected. Table 4 shows the recall of various variable-selection strategies. Indeed the simplest strategy LC performs the best. The numbers reported here have been measured using the first 3 PC dimensions; we find no further improvement using more dimensions. We observe that 50% recall is the lowest, thus we hypothesize that for attacks of up to $x\%$, flagging the top-$2.x\%$ should suffice.

Algorithm 2 is an unsupervised procedure; however it has a parameter r which is the number of users to be removed. In our experiments, we have kept r to be the same as the number of shilling profiles inserted. In the real world however, the number of shilling profiles inserted may be unknown. Thus, approximating r is of significance; we suggest choosing r to be a fixed value of 10% of the total number of users. Choosing too high a value can potentially remove important authoritative users as well. Further, the ranking function can also be varied; we investigate some ranking strategies for shilling profiles below.

While it is impossible to evaluate what a reasonable value for r is, we note that creation of shilling profiles will often involve manual effort. Even if large scale additions of shilling profiles is made possible by exploitation of bugs in service providers or their APIs, large scale infusion of user votes can be better detected using time series methods, i.e. analyzing how quickly normal users vote, with respect to obvious outliers creating hundreds of votes in a day. Given such infusions are detected, we hypothesize that insertion of attack profiles larger than 5% of the user population without detection

Table 4 Detection accuracy for various variable-selection strategies on the large MovieLens Dataset (6040 users) for Random Attacks

Attack size	Filler(%)	Variable-selection strategy		
		LC	SLC	QLC
60	1	60	60	60
	3	59	58	58
	5	57	56	55
	7	54	49	48
	10	48	42	40
	15	40	36	35
190	1	190	187	190
	3	187	184	180
	5	174	168	163
	7	165	155	148
	10	147	138	130
	15	118	42	95
450	1	449	445	443
	3	413	381	363
	5	355	339	330
	7	317	307	294
	10	267	237	226
	15	243	219	208
600	1	563	559	549
	3	480	471	421
	5	466	425	412
	7	398	372	364
	10	343	328	317
	15	294	280	272

We report the number of correctly identified shillers in the top 10% of the ranked list generated by each criteria

is unlikely. It is difficult however, to evaluate if this assumption is representative of real-world systems in the absence of any statistics from service providers. Given this, we explore in the next section how a robust collaborative filtering algorithm performs for different cutoff-values.

8 Robust collaborative filtering using VarSelect

SVD and its derivatives have proven successful in collaborative filtering (Canny 2002; Brand 2003); more evidence of SVD's usefulness for collaborative filtering is in the NetFlix prize and KDD Cup (Bennett et al. 2007) where contestants have used non linear variations of SVD to get improved results cf. (Paterek 2007), as compared to standard CF algorithms. Our recent work (Mehta et al. 2007b) explored a RMF algorithm which uses M-estimators in combination with SVD-like Matrix factorization (MF). Our extension of VarSelect to large datasets provides an insight on how we can use combine PCA VarSelect and SVD based CF to get a robust collaborative filtering algorithm.

The idea can be sketched as follows: given the dataset, we first perform an thin SVD on mean centered data for 3–5 dimensions; this is the same step as discussed in the Sect. 7.3. Following this, we can sort users in an increasing order of their Principal Component coordinates (as discussed in Algorithm 2). As an end point of the SVD procedure on the data matrix, we get a list of suspicious users which are *flagged*. Thereafter, we maintain a blacklist of users who are potentially suspicious, and train an SVD model without contributions from them. Since this model has the same mathematical underpinnings, we do not need to repeat our previous computations. Instead, the SVD procedure is initialized with the already computed SVD, and a few more iterations are performed.

To understand in more detail how this algorithm works, we look deeper in the SVD learning procedure. We use the *Generalized Hebbian Learning* algorithm Gorrell (2006) which learns the rank-1 SVD as follows: assuming that the original data matrix $\mathbf{D}$, we find matrix factors $\mathbf{G}$ and $\mathbf{H}$ such that

$$\mathbf{D} \approx \mathbf{GH} \; s.t. \; rank(\mathbf{G}) = rank(\mathbf{H}) = 1 \tag{16}$$

Matrix factorization can be applied to an $n \times m$ matrix to recover a low rank approximation of under constraints of some cost function. One such cost function is the Frobenius norm:

$$||\mathbf{A} - \mathbf{B}||_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2} \tag{17}$$

Under this cost function, MF reduces to

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} \, ||\mathbf{D} - \mathbf{GH}||_F, \tag{18}$$

which is a formulation that is equivalent to the SVD, if singular values are absorbed appropriately into the left and right singular vectors. We use Hebbian learning (Gorrell 2006) to solve the above optimization.

Mathematically the Hebbian learning rule can be expressed as follows: suppose $\mathbf{u}$ and $\mathbf{v}$ are the first eigenvectors being trained for Matrix $\mathbf{D}$, and $D_{ij} = x$. Further, suppose the eigenvalue σ is absorbed into the singular vectors $\mathbf{u}$ and $\mathbf{v}$ to yield $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. The estimate for x would then be

$$x_{\text{est}} = \hat{u}_i \cdot \hat{v}_j. \tag{19}$$

Since this estimate might have an error, lets suppose further that the residual is represented by $r(x)$.

$$r(x) = x - x_{\text{est}} = x - \hat{u}_i \cdot \hat{v}_j, \tag{20}$$

To get a better estimate of the modified eigenvectors, the Hebbian learning rule updates the value based on the error.

$$\triangle \hat{u}_i = \lambda \cdot \hat{v}_j \cdot r(x), \quad \triangle \hat{v}_j = \lambda \cdot \hat{u}_i \cdot r(x) \tag{21}$$

where λ is the learning rate. It can be shown that with the suitable choice of decaying learning rates, the repeated iteration of the above equations converges to the required eigenvectors if the matrix is complete.[6] After the first pair of singular vectors has been learnt, their projection can be removed ($x \leftarrow x - u_1 \cdot v_1$) and the next pair can be learnt.

Our modification in presence of *flagged* users is to only update the left vectors and not the right vectors (see Eq. 20). In other words, the contributions of suspicious users towards the prediction model is zero, while the model can still predict the votes for flagged users. This elegant solution comes with a very small computational cost of checking if a given user is flagged as suspicious. The complete procedure is explained in Algorithm 3. Note that the entire model works with the z-scores instead of discrete votes, a practice which was found useful in Hofmann (2004).

Algorithm 3 PCARobustCF (**D**, r)

```
1: $\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$)
2: $\mathbf{U}\lambda\mathbf{V}^T$= SVD($\mathbf{D}$) {Get principal components}
3: $PCA_1 \leftarrow \mathbf{U}(:,1), PCA_2 \leftarrow \mathbf{U}(:,2)$ {First and second principal components }
4: for all columnid $user$ in D do
5:     $Distance(user) \leftarrow PCA_1(user)^2 + PCA_2(user)^2$
6: end for
7: Sort $Distance$
8: Flag top $r$ users with smallest $Distance$ values
9: for Factor $f_k$ with $k \leftarrow 1$ to $d$ do
10:    $\mathbf{D} = \mathbf{D} - \mathbf{u}_{k-1} \cdot \mathbf{v}_{k-1}^T$
11:    repeat
12:        $r_{ij} = \mathbf{D}_{ij} - \hat{u}_i \cdot \hat{v}_j$
13:        $\triangle \hat{u}_i = \lambda(\hat{v}_j \cdot r_{ij} - \kappa \cdot \hat{u}_i)$
14:        if $u$ is not flagged then
15:            $\triangle \hat{v}_j = \lambda(\hat{u}_i \cdot r_{ij} - \kappa \cdot \hat{v}_j)$
16:        end if
17:    until Convergence of $\hat{u}_i, \hat{v}_j$ for all $i, j$
18: end for
```

Output: Return Matrix factors **u**, **v**

9 Experimental setup and results

To evaluate the performance of our proposed algorithms algorithm, we use the MovieLens dataset which has been used previously for evaluating shilling detection. To this data, shilling profiles are added which all target the same item which is selected at random. shilling profiles are generated using the well studied models of Average,

[6] For matrices with missing values, the above minimization converges to a local minimum.

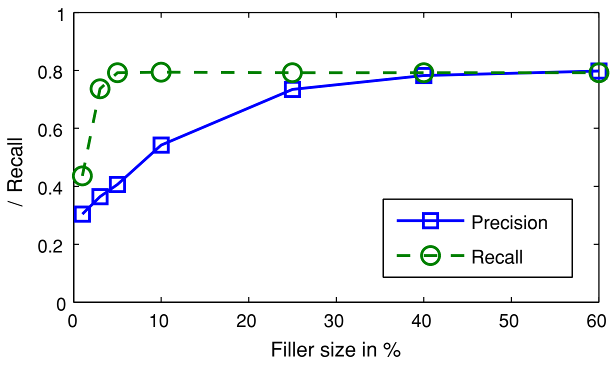

Fig. 5 Detection Recall and Precision for Average Attacks of varying filler sizes using PLSA based detection (*Algorithm 1*). 10% shilling profiles were added to 944 normal users

Random and Bandwagon attacks. We use the generative models explained in Burke et al. (2006) which add Gaussian noise to item or overall averages. It is important to note that we assume the MovieLens dataset has no existing shilling profiles, thus the attack profiles we add are the only shilling profiles present. Experimental results have been found to hold on larger datasets like EachMovie: we present results on the 100k ML dataset to be directly comparable with other reported results. For robust collaborative filtering, we use the larger MovieLens dataset with 6040 users and 3952 items. Again, we assume the larger ML dataset to be shill-free as well (Fig. 5).

9.1 PLSA based shilling detection

Experimental results for PLSA based detection show that shillers are indeed clustered together: in most experiments, all shillers end up in one cluster. Moreover, using the *closeness* measure of a cluster (see Eq. 7) also works well in most cases. For medium and large sized attacks, (see Fig. 6) more than 70% attackers are correctly identified. However the precision is low as many normal users are also misclassified. We find 20 communities to be ideal, which makes each cluster between 2–10%. For small attacks, PLSA based detection is ineffective. For very small filler sizes (% of rated items) and attack sizes (no. of profiles inserted), low recall and precision are observed. Also in 20% of the cases (2 out of 10 trials), the wrong cluster is selected, leading to maximum 80% recall and precision on an average. This experiment also explains the robustness of PLSA against shilling: the effect of spam is large only in the cluster where most shillers are. For all other clusters, the prediction shift is much lesser as the effect is weighted by the cluster weight of shillers, which is usually a small fraction. However, for large attack sizes, we note that large clusters are formed with a majority of the users in the same cluster as shillers, hence explaining the large prediction shift reported in Mobasher et al. (2006).

Drawbacks of PLSA based shilling detection: PLSA based shilling detection works well against strong attacks like the average attack and exploits the extraordinary similarity between shillers; however, for weaker attacks shillers tends to be distributed

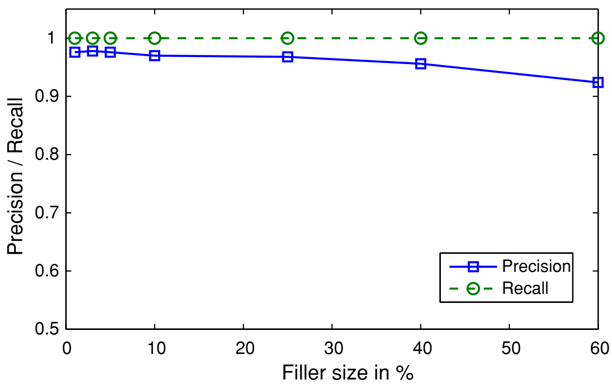

Fig. 6 Detection Recall and Precision for Average Attacks of varying filler sizes using PCA based VarSelect (*Algorithm 2*). 10% shilling profiles were added to 944 normal users

across different communities and hence are impossible to isolate. Similar trends are noted for very small attacks, where 1–3% of the user population is added. Moreover, for weaker attacks, very low precision and recall are observed, meaning that throwing out a cluster can lead to the loss of many real users. The reason for this is that PLSA is a parametized algorithm, where the number of communities (clusters) is taken as input (usually between 20 and 40). The expected size of a cluster is around 50 for our dataset. Clearly, smaller attacks would lead to clustering where shillers are a part of a larger group of users, thus the characteristics of normal users would dominate in that group. Thus, PLSA based detection works well in certain conditions, like large attacks, and fails for small and weak attacks. We still think this is a strong algorithm because using PLSA for recommendation can lead us to inspecting suspicious clusters where extreme similarity is observed, and this is done at low additional cost. In the remainder of this section, we focus on PCA based shilling detection which works much better in a variety of conditions.

9.2 PCA based shilling detection

Variable-selection based on PCA does much better than PLSA based shilling detection (see Fig. 6): more than 90% precision and almost 100% recall is consistantly observed for the same shilling profiles used for PLSA based detection. PCA does very well in identifying profiles which are very similar to most other profiles, since they add very little information. With 10% attack profiles, the top 10% eliminated users are shillers with more than 90% accuracy. Similar numbers are observed for a variety of attacks (*random, average, bandwagon*) in variety of attack sizes, and filler sizes. Further accuracy is gained by considering the first three Principal components, and sorting the variables *together*. To evaluate the performance of our PCA-VarSelect algorithm, we use the MovieLens dataset[7] which consists of 100,034 votes by 944 users over 1682 movies. This dataset has been used for evaluating previous shilling

[7] http://www.grouplens.org/data/ml-data.tar.gz.

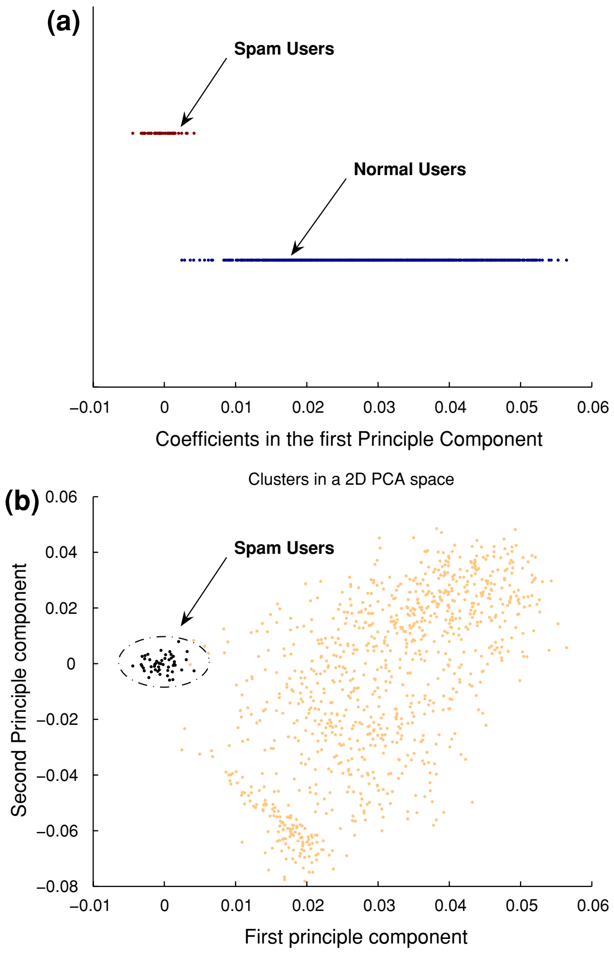

Fig. 7 (**a**) Clusters in the first PC space for normal users and shillers. The coordinates here are the coefficients of each user in the 1st principal component. (**b**) Clusters in 2D space for normal users and shillers. The coordinates here are the coefficients of each user in the 1st and 2nd principal component. Notice that the shillers are centered around the origin, due to their low PC scores

detection algorithms as well. To this data, shillers are added which all target the same item which is selected at random. shillers are generated using the well studied models of Average, Random and Bandwagon attacks.

The results of applying this algorithm are very positive: PCA coefficients show clear clusters of data, which are even more evident if visualized in 2D (1st and 2nd PC), as shown in Fig. 7a, b. While the clusters are very clear for unsophisticated attacks (see Fig. 7), more sophisticated attacks do not significantly alter this property. With 5% attack profiles, the top 5% eliminated users are shillers with more than 90% accuracy. Similar numbers are observed for a variety of attacks (*random, average, bandwagon*) in variety of attack sizes, and filler sizes. Further accuracy is gained by considering the first 3 Principal components, and sorting the variables *together*. To

Table 5 Detection precision for random attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	96.0	96.0	100.0	94.0	96.0	98.0
2%	96.0	98.0	99.0	98.0	97.0	99.0
5%	97.6	97.6	98.0	97.6	98.0	98.4
10%	97.4	98.4	98.6	98.8	98.6	98.6

Table 6 Detection precision for average attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	90.0	92.0	94.0	96.0	90.0	80.0
2%	95.0	96.0	95.0	93.0	89.0	86.0
5%	97.6	98.0	96.8	96.8	94.8	92.8
10%	97.6	97.8	97.6	97.0	96.8	95.6

Table 7 Detection precision for bandwagon attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	78.0	88.0	94.0	94.0	96.0	98.0
2%	82.0	88.0	90.0	97.0	95.0	95.0
5%	88.0	93.6	94.4	96.8	96.0	98.0
10%	87.0	94.2	96.4	97.6	98.4	98.2

Table 8 Detection precision for average + bandwagon attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	74.0	84.0	82.0	80.0	78.0	68.0
2%	76.0	87.0	90.0	89.0	87.0	83.0
5%	87.2	92.0	92.4	93.6	92.8	91.2
10%	85.8	93.2	96.0	95.8	95.0	95.2

do this, assume the three coefficents of each variable represent a point in $\mathbb{R}^3$ and sort the points in order of their distance from origin. Our experiments show an improved performance as compared to using single PCs. Using further PCs does not add to the performance. All numbers we report below use the first three principal components to do variable-selection. Tables 5–8 report detection results on various attack strategies.

Tables 9–12 show the peformance of VarSelect on obfuscated attacks. It is importantly to note that there is no perceivable drop in detection accuracy in the face of obfuscation. It is easy to see why; all obsfication strategies mentioned in literature

Table 9 Detection precision for obfuscated random attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	94.0	100.0	90.0	100.0	94.0	98.0
2%	99.0	97.0	97.0	97.0	96.0	96.0
5%	97.6	98.0	98.0	98.0	97.2	98.8
10%	97.8	98.6	98.6	98.4	98.4	98.2

Three kinds of obfuscation strategies have been used: random noise, user shift and target shifting

Table 10 Detection precision for obfuscated average attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	96.0	94.0	90.0	90.0	84.0	74.0
2%	96.0	95.0	95.0	93.0	89.0	81.0
5%	96.8	95.6	96.4	95.6	92.4	91.2
10%	97.4	96.4	97.4	96.8	96.2	92.6

Three kinds of obfuscation strategies have been used: random noise, user shift and target shifting

Table 11 Detection precision for obfuscated bandwagon attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	78.0	80.0	84.0	96.0	100.0	94.0
2%	79.0	87.0	92.0	94.0	97.0	96.0
5%	85.6	92.8	93.2	95.6	96.8	98.0
10%	85.8	94.0	96.2	97.0	98.0	98.2

Table 12 Detection precision for obfuscated bandwagon + average attacks of varying sizes

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	68.0	72.0	80.0	78.0	72.0	64.0
2%	76.0	81.0	83.0	87.0	83.0	79.0
5%	84.0	85.2	88.4	90.4	87.6	84.4
10%	82.6	89.8	91.8	92.8	92.4	89.2

Three kinds of obfuscation strategies have been used: random noise, user shift and target shifting

are linear transformations, which are not very effective when linear dimensionality reduction is performed. This is an important result: future work should explore which non-linear transformation can be effective in inducing prediction shift, while adding more stealth.

An additional test was performed to evaluate the effectiveness of the PCA-VarSelect algorithm when faced with an uncoordinated attack. In this setting, we introduce unco-

Table 13 Detection precision for a mixture of uncoordinated Attacks of varying sizes and types

Attack size	Filler size					
	1%	3%	5%	10%	25%	40%
1%	74.0	86.0	82.0	86.0	80.0	78.0
2%	78.0	85.0	85.0	88.0	83.0	83.0
5%	82.8	89.2	92.8	92.4	92.8	88.0
10%	85.8	92.0	94.2	94.2	94.8	92.8

ordinated shilling profiles; we add n profiles which attack one of the chosen 10 items randomly, and use either of three attack strategies: random, average, or bandwagon. A sample of 100 such profiles may contain 45 average attack profiles, 30 random attack profiles and 25 bandwagon attack profiles. The attacked item may be different for different attack profiles and each profile may be performing either a push or a nuke attack. In the real world, many different items maybe simultaneously be attacked, by different attacks and a shilling detection algorithm should still perform well against such attacks.

Table 13 shows that PCA-VarSelect can still successfully detect such attacks: we note that VarSelect is fairly accurate on such a mixed attack, though accuracy is 10–15% lower than on coordinated attacks. Since previous work has not reported results on such attacks, it is difficult to compare our approach. Note that for large uncoordinated attacks, detection accuracy is much higher, presumably because the algorithm can find enough regularity between various attacking profiles and items being attacked.

Varselect gives a ranked list of user scores, with the lowest scores corresponding to attack profiles with high probability. In previous work, we have artificially inserted r attack profiles, and have tested for exactly top-r users in the ranked list of PCA scores. In general, r may not be known in advance, and thus our robust CF algorithm needs to detect these parameters, using several heuristics to detect r.

10 Robust Collaborative Filtering

Based on our experimental findings, PCA based user selection performs better than PLSA based detection. A comparison with other reported algorithms shows a clear advantage for PCA-based selection. Burke et al. approach is based on a large set of features which exploit the characteristic properties of shillers. However, the detection procedure results in a large number of false positives. Table 14 compares the reported performance of PCA versus the *Burke et al.* approach. However, drawbacks of both approaches do exist: our PLSA based approach identifies the correct cluster only 4 out of 5 times, and has low recall and precision against smaller attacks. When 50 shilling profiles were injected, the recall and precision were both around 25% lower than the reported numbers for detecting 100 profiles. Adding 1–3% profiles only results in zero recall. Clearly, smaller attacks are harder to detect. PCA based detection is more stable against attack size, but does not perform as well when attack profiles are not

Table 14 Detection precision for Push Attacks of size 1% at different filler sizes compared with other algorithms

Filler	Average attack		Random attack	
	Burke et al.	PCA	Burke et al.	PCA
1%	22	90	26	96
1 Obfuscated	10–22	92		94
5%	23	92	28	100
10%	29	96	29	94
10% Obfuscated	25–30	90		100
20%	32	90	33	96
20% Obfuscated	30–32	84		94
40%	39	80	40	98
40% Obfuscated	29–39	74		98
60%	42	68	47	92

Numbers for the Burke et al. algorithm have been reported from (Burke et al. 2006)

highly correlated. In this case, the attacks also have limited effect since the impact of a shilling profile is high only when it is similar to a number of users. Therefore, low-quality shilling data may not be very well detected by this method.

An interesting issue is the performance of our detection routines on clean datasets, i.e. user data without any shilling profiles. Clearly, the detection algorithms will flag the top-r profiles due to thresholds specified. If the profiles so-identified are removed from the dataset, two drawbacks are possible: *firstly*, the removed users will no longer be able to receive recommendations, resulting in a loss of user satisfaction. *Secondly*, it is possible that authentic and authoritative users are identified wrongly, due to their high level of agreement with other users that are influenced by them. The removal of such users can have a greater impact than the random removal of users, since more users are directly affective, possibly with a large shift in prediction accuracy. With this in mind, we propose a robust collaborative filtering algorithm, which does not remove any user from the database; instead, the influence of suspicious users is limited. We note that this approach is better suited for real-world deployment of recommender systems.

Using large scale VarSelect on NetFlix dataset

We now use the procedure described in Sect. 7.3 to determine if large scale datasets released recently have any evidence of attack profiles present in them. We use the complete NetFlix dataset (with the probe-set included) which has 480, 189 users, 17,770 movies and 100,480,507 votes.[8] We ran our large-scale variant of VarSelect on the NetFlix dataset to examine if it has shilling profiles. Figure 8 plots the coordinates of the first and second PC. We observe that some users have extremely low PCA coefficients. Note how similar this plot looks to the Fig. 7 where artificial profiles form a cluster away from the normal users near the origin. The PCA coordinates for almost

[8] The NetFlix dataset is available at www.netflixprize.com free of cost subject to a team registration.

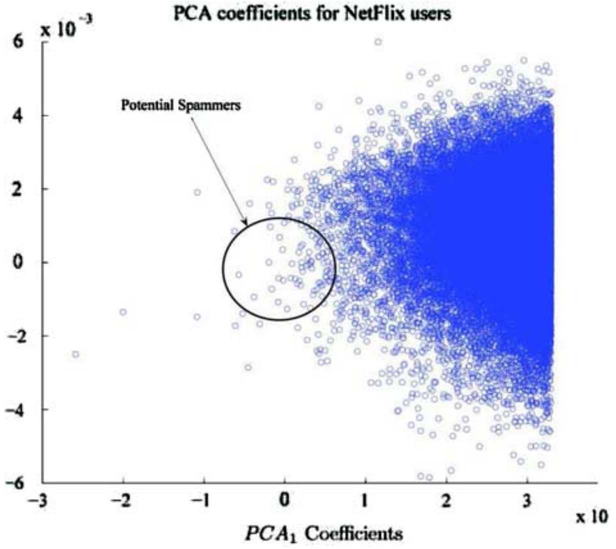

Fig. 8 A plot of the PCA coefficients for the NetFlix dataset. Notice how the outliers look similar to the outliers in Fig. 7 for the MovieLens dataset with artificial shilling profiles inserted

half a million users were obtained by running sparse SVD using Hebbian Learning (also used in Mehta et al. 2007a); the running time was under 20 min. Ofcourse, it is difficult to ascertain if the users *flagged* by VarSelect are indeed shillers or not. Since the user profiles are too large to be humanly examined, and user profiles themselves may not contain sufficient information to be considered as scientific proof, it is difficult to make any conclusions. On a 2-D scale, clustering with our data is also not very successful, as the identified clusters contains several thousand users. This is inline with previous observations that shilling profiles cannot be identified by outlier detection mechanism (*like* clustering). However, we are able to demonstrate that VarSelect can be scaled to the largest CF dataset publically available.

10.1 Robust collaborative filtering using VarSelect

As outlined in Algorithm 3, we have implemented an SVD procedure based on Hebbian Learning. We chose the larger MovieLens dataset with 6040 users and 3942 movies. As in previous sections, we add artificial shilling profiles generated using the Average attack model. We then apply two baseline collaborative filtering algorithms: similarity based k-NN and SVD using Hebbian Learning. 20% of the dataset has been randomly taken out and the used as a test set.

We vary the attack and filler size of the inserted profiles and measure the mean absolute error over the existing votes in the test set. We do not use prediction shift in order to make different algorithms comparable; prediction shifts for more accurate CF algorithms can give artificially good results as compared to less accurate CF

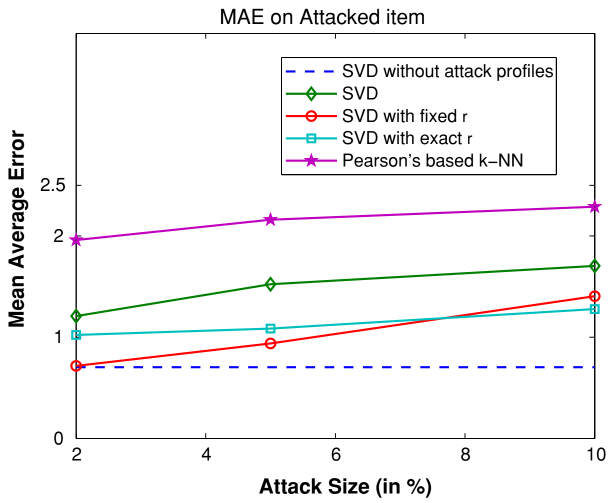

Fig. 9 MAE for Average Attacks of filler size 0.03 using VarSelect based CF (*Algorithm 3*)

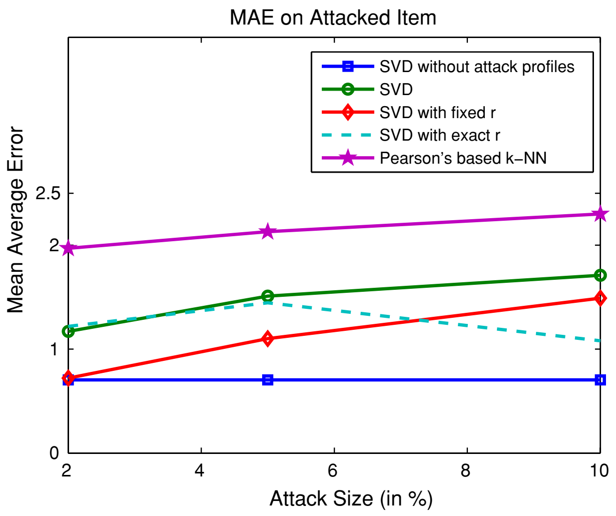

Fig. 10 MAE for Average Attacks of filler size 0.05 using VarSelect based CF (*Algorithm 3*)

approaches. We also compare our results with CF run on the original dataset without any inserted attack profiles.

Figures 9–11 show the desired results; the proposed robust CF algorithm performs significantly better than all approaches in all conditions. In cases of small attack size, our algorithm is near optimal. Note that we chose two variants of our approach: one, where the number of users to be flagged r is kept constant (at 7% of the user population), and another, where r is the same as the number of profiles inserted. Results clearly show a benefit using SVD VarSelect; both high robustness and better accuracy are obtained. The overall performance of the system tested over a larger test set does

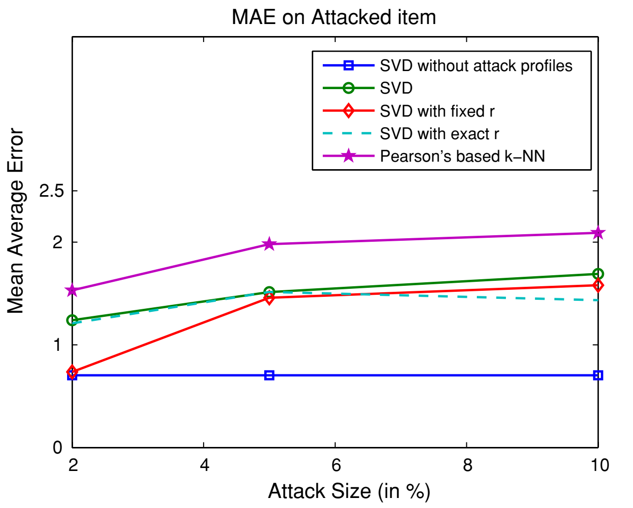

Fig. 11 MAE for Average Attacks of filler size 0.1 using VarSelect based CF (*Algorithm 3*)

Table 15 Overall MAE on a test set for various CF approaches

Attack size	Filler size	SVD without attack	SVD	SVD with VarSelect	Pearson based k-NN
	2	0.6755	0.6760	0.6762	0.8095
3%	5	0.6755	0.6763	0.6768	0.8115
	10	0.6755	0.6776	0.6763	0.8135
	2	0.6755	0.6761	0.6764	0.8077
5%	5	0.6755	0.6767	0.6761	0.8087
	10	0.6755	0.6783	0.6769	0.8093
	2	0.6755	0.6766	0.6768	0.8062
10%	5	0.6755	0.6774	0.6783	0.8062
	10	0.6755	0.6785	0.6778	0.8073

not suffer; only a marginal difference is noted (see Table 15) as compared to CF without any attack data. Other approaches based on pure SVD do not suffer much either; this can be attributed to better accuracy on all non-attacked items due to the addition of many mean-like votes due to profile injection.

11 Conclusion

This paper discusses novel interpretations of known unsupervised algorithms for detecting highly correlated groups of shillers which are hard to detect using other statistical measures. The algorithms presented here are fast and highly accurate and have not been discussed in literature to the best of our knowledge. Due to their unsupervised nature, these algorithms can scale very well, and can be used as a pre-processing step for recommendation algorithms. This paper also provides an explanation for the phenomenal robustness of PLSA under shilling attacks reported recently by Mobasher et al. (2006). In particular, the PCA based variable-selection stands out as an extremely effective detection method. We also present a novel robust collaborative filtering algorithm, which is highly accurate, and stable to shilling. especially, when only a small fraction of profiles are malicious in nature. Clearly, our work is only a starting point for building highly robust recommender systems. One direction would be developing algorithms, which treat attack profiles differently from noisy data, and also use some distinctive features based on voting patterns for further increasing precision of shilling detection. More theoretical work on stability of popular recommender algorithms will add significantly to the state-of-the-art in personalization.

Acknowledgements The authors would like to thank Thomas Hofmann and Peter Fankhauser for various discussions fundamental to this work. We would also like to thank the anonymous and knowledgeable reviewers who made many suggestions to significantly improve this work.

References

Al-Kandari, N.M., Jolliffe, I.T.: Variable selection and interpretation in correlation principal components. Environmetrics **16**(6), 659–672 (2005)

Bennett, J., Elkan, C., Liu, B., Smyth, P., Tikk, D.: KDD Cup and workshop 2007. SIGKDD Explor. **9**(2), 51–52 (2007)

Brand, M.: Fast online SVD revisions for lightweight recommender systems. Proceedings of SIAM International Conference on Data Mining, pp. 37–46 (2003)

Burke, R., Mobasher, B., Williams, C., Bhaumik, R.: Analysis and detection of segment-focused attacks against collaborative recommendation. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'06), pp. 542–547 (2006)

Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 238–245 (2002)

Chirita, P.-A., Nejdl, W., Zamfir, C.: Preventing shilling attacks in online recommender systems. In: WIDM '05: 7th Annual ACM International Workshop, pp. 67–74. ACM Press, New York, USA (2005)

Gorrell, G.: Generalized Hebbian Algorithm for incremental singular value decomposition in natural language processing. In: EACL. The Association for Computer Linguistics, pp. 97–104 (2006)

Hastie, T., Tibshirani, R., Eisen, M., Brown, P., Ross, D., Scherf, U., Weinstein, J., Alizadeh, A., Staudt, L., Botstein, D.: Gene shaving: a new class of clustering methods for ecpression arrays. Genome Biol. **1**, 1–0003 (2000)

Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 89–115 (2004)

Jolliffe, I.: Discarding variables in a principal component analysis. I: artificial data. Appl. Stat. **21**(2), 160–173 (1972)

Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer (2002)

Konstan, J.A., Riedl, J., Smyth, B. (eds.): Proceedings of the 2007. ACM Conference on Recommender Systems, RecSys 2007. ACM, Minneapolis, MN, USA, 19–20 October 2007

Lam, S.K., Riedl, J.: Shilling recommender systems for fun and profit. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 393–402. ACM Press, New York (2004)

Mahalanobis, P.: On the generalized distance in statistics. Proc. Natl. Inst. Sci. India **2**(1), 49–55 (1936)

Mehta, B., Hofmann, T., Fankhauser, P.: Lies and propaganda: detecting spam users in collaborative filtering. In: IUI '07: Proceedings of the 12th International Conference on Intelligent User Interfaces, pp. 14–21. ACM Press, New York (2007a)

Mehta, B., Hofmann, T., Nejdl, W.: Robust collaborative filtering. In: Konstan (2007), pp. 49–56. ACM (2007b)

Mobasher, B., Burke, R.D., Sandvig, J.J.: Model-based collaborative filtering as a defense against profile injection attacks. Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06), pp. 1388–1393. AAAI Press, Boston, MA (2006)

O'Mahony, M., Hurley, N., Kushmerick, N., Silvestre, G.: Collaborative recommendation: a robustness analysis. ACM Trans. Int. Tech. **4**(4), 344–377 (2004)

O'Mahony, M.P., Hurley, N.J., Silvestre: Detecting noise in recommender system databases. In: Proceedings of the International Conference on Intelligent User Interfaces (IUI'06), 29th–1st. Sydney, Australia, pp. 109–115. ACM Press (2006)

Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. Proceedings of KDD Cup and Workshop (2007)

Riedl, J., et al.: MovieLens dataset, available at http://www.cs.umn.edu/Research (1998)

Sandvig, J.J., Mobasher, B., Burke, R.D.: Robustness of collaborative recommendation based on association rule mining. In: Konstan (2007), pp. 105–112. ACM (2007)

Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)

Williams, C., Mobasher, B., Burke, R., Sandvig, J., Bhaumik, R.: Detection of obfuscated attacks in collaborative recommender systems. In: Workshop on Recommender Systems, ECAI (2006)

Authors' vitae

Bhaskar Mehta (born 1979) is a computer scientist and former Postdoctoral Researcher at the L3S Research institute (where this work was done). He formerly worked with Fraunhofer IPSI in Darmstadt. Within L3S, he served as the project manager for the European project PHAROS, which is concerned with audiovisual search (like YouTube). He currently works as a software engineer at Google Inc. at Zurich. He graduated from Indian Institute of Technology, New Delhi, with a B.Tech (2001) and M.Tech degree (2002) in Computer Science and Engineering. He received his PhD in from University of Duisberg-Essen in 2008, graduating *summa cum laude* with a thesis titled *Cross system personalization: Enabling personalization across multiple systems*. His general research interests are Personalization and recommender systems, Web and Graph Search algorithms, data mining and statistical AI.

Wolfgang Nejdl (born 1960) has been full professor of computer science at the University of Hannover since 1995. He received his M.Sc. (1984) and Ph.D. degree (1988) at the Technical University of Vienna, was assistant professor in Vienna from 1988 to 1992, and associate professor at the RWTH Aachen from 1992 to 1995. He worked as visiting researcher / professor at Xerox PARC, Stanford University, University of Illinois at Urbana-Champaign and EPFL Lausanne. Wolfgang Nejdl heads the Information Systems Institute/Knowledge Based Systems[9] as well as the L3S Research Center,[10] and does research in the areas of semantic web technologies, peer-to-peer information systems, adaptive hypermedia systems and artificial intelligence. Wolfgang Nejdl published more than 200 scientific articles and has been programme committee and editorial board member of numerous international conferences and journals.

[9] www.kbs.uni-hannover.de.

[10] www.l3s.de.