Attack Resistant Collaborative Filtering

Bhaskar Mehta
Google Inc.
Brandschenkestr 110, 8004 Zurich, Switzerland
mehta@l3s.de

Wolfgang Nejdl
L3S / Universität Hannover
Appelstrasse 4, 30167 Hannover, Germany
nejdl@l3s.de

ABSTRACT

The widespread deployment of recommender systems has lead to user feedback of varying quality. While some users faithfully express their true opinion, many provide noisy ratings which can be detrimental to the quality of the generated recommendations. The presence of noise can violate modeling assumptions and may thus lead to instabilities in estimation and prediction. Even worse, malicious users can deliberately insert attack profiles in an attempt to bias the recommender system to their benefit.

While previous research has attempted to study the robustness of various existing Collaborative Filtering (CF) approaches, this remains an unsolved problem. Approaches such as Neighbor Selection algorithms, Association Rules and Robust Matrix Factorization have produced unsatisfactory results. This work describes a new collaborative algorithm based on SVD which is accurate as well as highly stable to shilling. This algorithm exploits previously established SVD based shilling detection algorithms, and combines it with SVD based-CF. Experimental results show a much diminished effect of all kinds of shilling attacks. This work also offers significant improvement over previous Robust Collaborative Filtering frameworks.

Categories and Subject Descriptors

H.3 [**Information Storage And Retrieval**]: Information Search and Retrieval; K.4.4 [**Computers and Society**]: Electronic Commerce – *Security*

Keywords

Collaborative Filtering, Shilling, Robust SVD

1. INTRODUCTION

Personalization for large scale systems is a well researched topic in Computer Science with several successful algorithms and improvements over past years. While early algorithms exploited similarity in small groups amongst a large population of users, later algorithms made use of advanced statistical models. Improvement in accuracy was the main objective of previous research, though research has been conducted to incorporate aspects like Trust and Privacy. One important issue with Collaborative Filtering (CF) to emerge recently is the vulnerability towards well designed attacks. Such attacks require a group of users to collude and insert malicious ratings on chosen items; this is aimed at manipulating a recommender system to recommend a chosen item more/less frequently than normally. Such attacks have been called *shilling attacks* [9] or *profile injections attacks* [2]. While recent algorithms [2, 10, 11, 21] are successful in identifying shilling attacks in collaborative filtering, it is desirable to develop algorithms which are robust to shillers from the ground up. Previous work [21] has used a probabilistic model based on Singular value Decomposition (SVD) to detect shillers, however the detection rate is low, and the algorithm is ineffective against average attacks.

Recent work [11] introduced a detection algorithm based on Principal Component Analysis (and hence SVD) which was very accurate against a broader range of attacks. Our aim in this paper is to built detection into the CF in a computationally effective manner. A robust collaborative filtering algorithm would provide protection from insertion of random noise as well as attack profiles injected into the system without any explicit input or tuning. Recent work concluded that noise resistant statistical methods using robust regression are not completely effective for robustifying collaborative filtering [12].

The contribution of this paper is a robust CF algorithm which is stable against moderate shilling attacks on large datasets. Our proposed algorithm leverages the accuracy of PCA-based attack detection [11] while preserving the predictive accuracy of SVD, which has also been successfully exploited previously [19, 21].

2. BACKGROUND & RELATED WORK

Collaborative Filtering (CF) [8] is one of the most popular and successful filtering techniques that has been used to date. It is used in a setting where users have a choice of a number of items (say, a book store) and can rate items that they know about. Collaborative Filtering helps users to make choices based on the opinions of other similar users in a system and find relevant items that they may not have explored so far. The basic idea employed is that users who agree with each other on some items based on their ratings are likely to agree or disagree on future items.

Collaborative filtering algorithms have been classified into two general categories, commonly referred to as *memory-based* and *model-based* algorithms. *Memory-based algorithms* are the more prevalent of the two categories and use all available data in order to make a prediction for the selected user. Memory based CF algorithms retain all relevant data in memory and compute the required prediction on demand in real. *Model-based algorithms* operate differently by abstracting from the observed data and creating a statistical model of observed data. This model is learnt based on known ratings and is subsequently used in the recommendation process. The CF algorithm presented in this paper is a model-based algorithm as well.

2.1 Shilling and Collaborative Filtering

Collaborative Filtering systems are essentially social systems which base their recommendation on the judgment of a large number of people. Like other social systems, they are also vulnerable to manipulation by malicious social elements. In a well knonw incident, a loosely organized group managed to trick the Amazon recommender into recommend to some readers of the book *Six Steps to a Spiritual Life* (written by the evangelist Pat Robertson), a book for gay men[1].

A lot of web-enabled systems provide free access to users via a simple registration process. This can be exploited by attackers to create multiple identities for the *same* system and insert ratings in a manner that affect the robustness of a system or algorithm, as has been studied in recent work [9, 13]. *Shilling attacks* add a few user profiles which need to be identified and protected against. Shilling attacks can be classified into 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 [13]. Various attack strategies were then invented; these include [2]:

1. *Random attacks*, where a subset of items is rated randomly[2] 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, and some popular items are rated with the maximum vote.

Random and Bandwagon attacks are low-knowledge attacks requiring information only about some popular items and overall vote statistics. Average attacks require more information , and have been shown to be near optimal [10] in impact. They have also been observedly difficult to detect [21].

The strength of shilling attacks is specified using two metrics: *filler size* and *attack size*. Filler size is the set of items which are voted for in the attacker profile, usually measured in %. Attack size refers to the number of shilling profiles inserted into user data. The impact of the attacks is measured by the increase in the number of users to whom an attacked item is recommended. Generally, average attacks are strogner than random or bandwagon attacks.

2.2 Detection of Shilling attacks

Recent research in this area aimed at detecting algorithms for profile injection attacks. The earliest shiling detection algorithm was invented by Chirita et al. [3] and exploited features of spam profiles. 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. Mobasher et al. [2] compare their feature-based classification algorithm which performs significantly better than the Chirita algorithm by taking more features into account. The Mobasher et al. [2] algorithm trains a classifier given enough example spam and authentic profiles and is fairly accurate in detecting spam attacks of varying sizes and density. However, as a supervised approach, it needs a large number of examples, and can detect only profiles similar to the examples profiles. Secondly, both algorithms perform badly when the spam 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. [20] discusses these obfuscation strategies and their effect on detection precision. O'Mahony et al. [14] have taken up a more principled approach using signal processing theory to detect natural and malicious noise; however, the accuracy remains low (15–25%).

Recent work [10, 11] provide highly accurate algorithm called *VarSelect* for detecting shilling attacks. The algorithm exploites the *group effect*: a property of shillers being effective when working in groups. Since shillers want to maximize their effect, they need to work together. A result of this is that attack profiles are similar to one another; this property can be exploited by methods based on dimensionality reduction to detect spam with high accuracy. Clearly, detection procedures can be applied only sparingly due to their highly computational and batch nature. Our approach in this paper uses the insight gained in the design of detection procedures to create a robust Collaborative Filtering Algorithm; VarSelect is an important step in our proposed algorithm.

2.3 Robustness in Collaborative Filtering

The earliest work on Robust CF was a modified k–NN algorithm with heuristics for neighbor selection [15] where the concept of profile utility was introduced. Association rules have also been used to add robustness to CF at the cost of decreased accuracy and coverage [16], though at the cost of decreasd accuracy, and much lower coverage. Recent work [12] has investigated the effectiveness of robust statistics in protecting against shilling attacks with an algorithm called Robust Matrix Factorization (RMF). This approach similar in spirit to SVD but is more stable to noisy data. The uniqueness in this algorithm is the usage of M-estimators, which bounds the effect of outliers and noisy data. Experimental results have shown that application of Robust statistics adds significant stability (10-30% lower prediction shift); however the algorithm is not completely immune in the face of attacks.

Still, RMF adds significant stability as compared to other CF methods like PLSA and k-NN. The major positive outcome of this work is that RMF outperforms all other algorithms based on latent semantics (PLSA, SVD) in prediction accuracy.

Clearly, RMF is only a partial success; however it allows us to understand that attack profiles are quite homogeneous. Detecting such profiles is clearly possible (see [11]), since they exhibit characteristic properties; notably, the similarities between attack profiles are much higher than with normal users, due to the way these profiles are constructed. Ideally, the detection phase should be built into the recommendation algorithm; detected attack profiles can then be removed and the recommendation phase can begin. However, there are two major problems: firstly, the detection phase is expensive and may need to be repeated periodi-

[1]Story at `http://news.com.com/2100-1023-976435.html`.

[2]Note that Gaussian distributions $\mathcal{N}_{\mu,\sigma}$ have been used for generating the random votes rather than the uniform random distribution

cally, resulting in high computational costs. Secondly, the accuracy of detection methods is not 100%, thus removing false positives will adversly affect the user experiance.

In the next section we explore SVD in detail since the algorithm describe in this work depends heavily on SVD based methods.

3. ATTACK RESISTANT CF USING SVD

Our strategy for robust collaboratve filtering is to exploit previously developed detection strategies in a computationaly SVD stands for Singular Value Decomposition; it is a method of factorizing a matrix into two orthonormal matrices and a diagonal matrix. SVD has become an important linear algebra procedure over the last 2 decades due to its extensive application in Information Retrieval and Data mining. It has been used for Latent Semantic Analysis [4] and Collaborative Filtering [18] with much success. Since SVD is fundamental to the algorithm in this paper, we explore SVD in detail. Notably, we explain a recent iterative algorithm for SVD using Generalized Hebbian Learning [6]. Further, we briefly explain the Robust Matrix Factorization algorithm described in [12] which is also based on SVD and is robust variant of SVD. Finally, we explain our proposed VarSelect SVD variant asa robust CF solution.

3.1 Singular Value Decomposition (SVD)

SVD is a more general form of Eigen value decomposition (EVD) [3] since it is applicable to rectangular matrices as well. SVD factorizes a rectangular $n \times m$ matrix $\mathbf{D}$ as follows

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} , \qquad (1)$$

where $\mathbf{U}, \mathbf{V}$ are unitary normal matrices and $\mathbf{\Sigma}$ is a diagonal matrix of size $\text{rank}(\mathbf{D}) \leq \min(m, n)$, where $\text{rank}(\mathbf{D})$ is the rank of the matrix $\mathbf{D}$. Moreover, the entries on the diagonal of $\mathbf{\Sigma}$ are in non-increasing order such that $\sigma_i \geq \sigma_j$ for all $i < j$. Note that we may chose to set all singular values $\sigma_i = 0, i > k$ for some $k \leq \text{rank}(D)$ (say $k = 10$), leading to an low rank approximation $\mathbf{D}_k$ of the matrix $\mathbf{D}$.

$$\mathbf{D}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^{\mathrm{T}} , \qquad (2)$$

where $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$ are now $n \times k$, $k \times k$ and $m \times k$ dimensional matrices, respectively. It can be shown that $\mathbf{D}_k$ is the minimizer of $\|\mathbf{D} - \hat{\mathbf{D}}\|_2$ for all matrices $\hat{\mathbf{D}}$ of rank less or equal to k.

SVD for Collaborative Filtering: Applications of SVD to Collaborative Filtering assume the representation of user-item ratings by such a $n \times m$ matrix $\mathbf{D}$. Here each of the n users corresponds a row in the matrix, whereas the m items are represented as columns, with D_{ij} representing the vote of user i on item j. The application of SVD to $\mathbf{D}$ leads to a low rank estimate $\hat{\mathbf{D}}$, which generalizes the observed data, since it may result in non-zero values $\hat{D}_{il}$, even for user-item pairs (i, l) that are unrated (often set to zero in $\mathbf{D}$, i.e. $D_{il} = 0$).

Typically, user–item matrices are very sparse ($\leq 5\%$ non-zero entries).Initial applications of SVD to CF (c.f. [18]) compensated for sparsity by replacing the missing values by overall mean. This approach, though more successful than previous CF approaches, is highly biased towards the used means. In addition, lack pf sparsity meant a larger computational problem to solve. In the last decade, there has been significant research on SVD for large and sparse matrices e.g. *PROPACK*[4] and *SVDPACK*[5]. However, these approaches do not treat missing values in a principled fashion, either treating them as zeros, or doign mean imputation. [21] discusses the use of the Expectation Maximization [5] procedure to approximate SVD optimally in the log-likelihood sense. However, their approach requires a SVD to be performed at each EM iteration, which is computationally very expensive and not practical for large matrices with millions of rows and columns.

A recent algorithm by Gorrell [6] proposed a new approach to computing SVD for virtually unbounded matrices. This method is based on the Generalized Hebbian Algorithm [17] and calculates SVD by iterating through only observed values. The method has been found to be highly accurate for CF and scales easily to the NetFlix dataset with 100 million votes. Below we describe this approach in detail.

3.2 SVD using Hebbian learning

Gorrell [6] extends an existing method for eigen decomposition to non-symmetric matrices of arbitrary sizes. In her approach (referred to now onwards as SVD-GHA), multiple eigen-values/vectors can be computed with this simple observation: the second eigen-value/vector of a matrix can be calculated by removing the projection of the previous eigenpair. This means that if $\mathbf{u}_1$ and $\mathbf{v}_1$ are the first singular vectors corresponding to the largest eigenvalue σ_1, then a matrix $\mathbf{D}_{rem}$ can be defined as follows

$$\mathbf{D}_{\text{rem}} = \mathbf{D} - \mathbf{u}_1\sigma_1\mathbf{v}_1^{\mathrm{T}} , \qquad (3)$$

The first eigen-value of $\mathbf{D}_{\text{rem}}$ is exactly the *second* eigenvalue of $\mathbf{D}$. This observation can be generalized to compute the first k eigenvectors/eigenvalues of a large sparse matrix. This method had been referred to as Hotelling's Deflation Method [7].

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

$$x_{\text{est}} = \hat{u}_i \cdot \hat{v}_j \text{ . with error } r(x) = x - x_{\text{est}} \qquad (4)$$

The total error is a sum of residuals $r(x)$

$$E = \sum_{x=\mathbf{D}_{ij}} r(x)^2 \qquad (5)$$

The above error can be minimized by Gradient Descent, following the derivative of the estimate at every observed matrix entry:

$$\triangle \hat{u}_i = \lambda \cdot \hat{v}_j \cdot r(x) \ , \ \triangle \hat{v}_j = \lambda \cdot \hat{u}_i \cdot r(x) \, , \qquad (6)$$

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. Webb [19] modified this basic algorithm by introducing weight decay regularization and range clipping. Several contestants of the *NetFlix Prize* use modified versions of the above algorithm.

[3]EVD decomposes a square matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is an Unitary normal Matrix and $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues of $\mathbf{A}$

[4]http://soi.stanford.edu/ rmunk/PROPACK/

[5]http://www.netlib.org/svdpack/

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

3.3 Robust Matrix Factorization

Robust regression problems have been studied in a linear setting where observables Y and inputs X are known and Y is assumed to be noisy. Robust Matrix Factorization (RMF) is algorithm which performs a robust SVD for CF using an Alternating fitting scheme [12]. The core idea is the use of bounded cost functions, which limit the effect of outliers. There is an entire body of work on the kind of bounded functions which are effective against noise; these functions are called Maximum Likelihood estimators or *M-estimators*. Mehta et al. chose the Huber M-estimator which is defined as follows:

$$w(r) = \begin{cases} r \leq \gamma & 1\,, \\ r > \gamma & \frac{\gamma}{|r|} \end{cases} \tag{7}$$

Armed with a robust estimator, we would like the perform the following Matrix factorization: assume we want to find the rank–1 factors $\mathbf{G}, \mathbf{H}$ as for data $\mathbf{D}$. such that

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} \sum_{D_{ij} \neq 0} \rho(D_{ij} - G_i \cdot H_j)\,,\ \text{s.t. } \rho(r) = \begin{cases} |r| \leq \gamma & \frac{1}{2\gamma} r^2\,, \\ |r| > \gamma & |r| - \frac{\gamma}{2} \end{cases}$$

We solve the above optimization using *Iteratively Reweighted Least Squares* (see [12] for details). First $\mathbf{H}$ is minimized with a fixed $\mathbf{G}$; then $\mathbf{H}$ is fixed and $\mathbf{G}$ is minimized. This repeated till both factors converge. Given the rank–1 estimates $\mathbf{G}, \mathbf{H}$, higher rank estimates can be easily computed in a similar manner to SVD-GHA.

Experiments show that Robust Matrix factorization algorithm also performs well in the face of moderate attacks. Clearly, the effect of shilling is low at small attack sizes, as the majority opinion is given more importance. However, once the number of votes by shillers are more than actual users, RMF starts treating the shillers' view as the majority opinion. Mehta et al. also show that RMF is more tolerant to shilling and model deviations than SVD and pLSA: the prediction accuracy of RMF is higher than any other method ; this trend continues even in the face of attacks.However for larger attacks, RMF is clearly inadequate at a robust CF algorithm. In the next section, we show how the RMF and SVD frameworks can be further robustified to yield our desired robust CF algorithm.

3.4 VarSelect SVD for Collaborative Filtering

VarSelect [11] is a variable selection algorithm based on PCA for detecting attack profiles. Shilling profiles tend to be highly correlated, which is a result of the colluded nature of shilling attacks. It is known that for multivariate data, highly correlated variables add very little information, and thus are eliminated by dimensionality reduction methods. VarSelect uses Principal Component Analysis to find which users add least information, and produces a ranking of users in order of utility. Experiments have shown that shillers are found with high precision at the top of these rankings.

VarSelect SVD

We first describe the broad framework for our proposed algorithm. SVD and PCA are closely related since PCA can be achieved via SVD. 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 $s_1 = \mathbf{w}_1^T \mathbf{x}$, where the p-dimensional coefficient vector $\mathbf{w}_1 = (w_1, ..., w_p)$ solves

$$\mathbf{w}_1 = \underset{|\mathbf{w}|=1}{\operatorname{argmax}}\ Var\left(\mathbf{w}^T \mathbf{x}\right)\,, \tag{8}$$

Principally, PCA is equivalent to performing an eigen decomposition of the *covariance* matrix of the original data. Since we want to combine VarSelect with Collaborative Filtering, SVD provides the required framework. The framework we devise would support two phases: detection (followed by removal of profiles/votes), and recommendation model building. For efficiency, it is required that these two phases can share computational steps. Since the detection may not be perfect, no user profiles should be completely deleted and even suspected attackers should be able to receive recommendations. Further, the entire procedure should be unsupervised, i.e. no further input should be required after the detection phase has been performed (e.g. thresholding how many shillers are there in the system).

An important observation we make here is that Principal components can be computed directly from the data 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 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$.[7]

$$\mathbf{C} = \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}} = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^{\mathrm{T}} \tag{9}$$

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 $\mathbf{U}$. This saves a significant computation effort as Eigen-decomposition of large covariance matrices is very expensive. Note that PCA requires $\mathbf{X}$ to be zero-mean.

Notice also, that the VarSelect procedure does not require all eigenvector-eigenvalue pairs to be computed. Our experiments have also shown that the first 3–5 Principal components suffice to detect attack profiles reliably. Thus a complete SVD is not required: instead, partial eigendecomposition can be performed. Such routines are available as *svds* and *eigs* in MATLAB and Octave, using the Arnoldi method[8]. Shillers can be easily detected (using the VarSelect procedure) by applying *svds* on z-scores of the user data matrix.

Dealing with Suspect Attack profiles

As noted earlier, Varselect gives a ranked list of user scores, with the lowest scores corresponding to attack profiles with high probability. In previous work, the authors have artificially inserted r attack profiles, and have tested for exactly top-r users in the ranked list of PCA scores. In general, r maynot be known in advance, and thus our Robust CF algorithm needs to detect such parameters. We propose below several heuristics to detect r.

Finding suspected Attack profiles: PCA can find a set of variables which are highly correlated, a fact exploited in the design of Varselect. Varselect essentially performs Variable Selection using a selection criteria called NLC. There are several other selection procedures discussed in Literature ([1] provides a good overview of these criteria). Table 1 briefly describe various criteria. While several other criteria have been proposed as well, they require a complete eigendecomposition, and/or conditional correlation matrices. As

[7]Though we don't use this property here, we note that this can be used to chose which matrix to use for computing eigenvalues, especially is one is much smaller.

[8]The FORTRAN package ARPACK is available as free software at `www.caam.rice.edu/software/ARPACK/`

Algorithm 1 VarSelectSVD ($\mathbf{D}$)

```
1: $\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$) {$\mathbf{D}$ has $N$ users and $M$ items}
2: $\mathbf{U}\lambda\mathbf{V}^{\mathrm{T}}$= SVD($\mathbf{D}$,3) {Get 3 principal components $\mathbf{U}^T$}
3: $PCA_1 \leftarrow \mathbf{U}(:,1)$,$PCA_2 \leftarrow \mathbf{U}(:,2)$,$PCA_3 \leftarrow \mathbf{U}(:,3)$ {First 3 PC loadings }
4: for all columnid $user$ in $\mathbf{D}$ do
5:    $Score(user) \leftarrow (|PCA_1(user)| + |PCA_2(user)| + |PCA_3(user)|)/3$ {using LC ranking scheme}
6: end for
7: Normalize and Sort $Score$ {$Score$ now sum to 1.}
8: $r_1 \leftarrow$ number of users with $Score$ below $\frac{1}{N}$
9: $r_2 \leftarrow N/5$ {Cutoff set to 20%.}
10: $r \leftarrow min(r_1, r_2)$
11: Flag top $r$ users with smallest $Score$ values
12: for Factor $f_k$ with $k \leftarrow 1$ to $d$ do
13:    $\mathbf{D} = \mathbf{D} - \mathbf{G}_{k-1} \cdot \mathbf{H}_{k-1}^{\mathrm{T}}$
14:    repeat
15:       $res_{ij} = \mathbf{D}_{ij} - \hat{G}_i \cdot \hat{H}_j$ {set $\kappa = 0.01$}
16:       $\triangle\hat{G}_i = \lambda(\hat{H}_j \cdot res_{ij} - \kappa \cdot \hat{G}_i)$
17:       if $u$ is not flagged $or$ $1 < D_{ij} < 5$ then
18:          $\triangle\hat{H}_j = \lambda(\hat{G}_i \cdot res_{ij} - \kappa \cdot \hat{H}_j)$
19:       end if
20:    until Convergence of $\hat{G}_i, \hat{H}_j$ for all $i, j$
21: end for
Output: Return Matrix factors $\mathbf{G}, \mathbf{H}$
```

an example, consider B_2 (see Table 1): this strategy involves doing a thin SVD, however 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.

Each of the strategies described in Tab. 1 gives a ranked list as output; however a suitable cutoff point still has to be selected. Table **??** shows the recall of various variable selection strategies. we note that the simplest strategy LC performs the best. The numbers reported use 3 dimensions; we find no further improvement by using more dimensions. We choose the following heuristic: normalize the scores so that the sum to 1, and then choose all user with scores below $\frac{1}{n}$ for n users. We observe also that 50% recall is the lowest; thus we hypothesize that for attacks of upto 10%, flagging top-20% should suffice. We limit selected users to be less than 20%. These selected users are known as *flagged* users.

Table 1: Criteria for Variable Selection based on PCA

	Description
LC	Select q variable with the lowest average of absolute values of Loading. (1-5 dimensions sufficient)
SLC	Squared loading combination, choosing lowest average of squared loading (1-5 dimensions sufficient)
QLC	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)

Table 2: Detection Accuracy for various Variable Selection strategies on the large MovieLens Dataset (6040 users) for random attacks. We report the number of correctly identified shillers in the top 10% of the ranked list generated by each criteria.

Attack size	Filler(%)	Variable selection Strategy		
		NLC	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

Computing Recommendations

The recommendation model is finally based on SVD as well. In essense, we perform SVD on the data matrix treating flagged users in a special manner. To simplify the prediction model, we absorb the eigenvalues into the left and right factors, as in the GHA based SVD method. As in Sec. 3.2, the data matrix is factorized into a left matrix $\mathbf{G}$ and a right matrix $\mathbf{H}$, such that the Frobenius norm of the error is minimized. Under this error function, the factorization reduces to:

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} ||\mathbf{D} - \mathbf{GH}||_F, \tag{10}$$

In the context of Collaborative Filtering, note that the left matrix $\mathbf{G}$ is user specific, i.e. each user has a corresponding row encoding their hidden preferences. Similarly, the right matrix $\mathbf{H}$ contains a column for each item. The solution to the above optimization requires iterating through all user votes and performing Hebbian updates via Eq. (6). Every vote influences both $\mathbf{G}$ and $\mathbf{H}$.

Our modification in presence of *flagged* users is to only update the left vectors and not the right vectors. 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. For normal users, we update both left and right vectors as with SVD-GHA. This elegant solution comes with a very small computational cost of checking if a given user is flagged as suspicious. Note also that the model can be initialized with values learnt from the partial SVD performed for PCA. We note that this results in faster convergence for the already computed dimensions. Additionally, we use a regularization parameter κ (set to 0.01); this step has been found to provide better model fitting and faster convergence. Algorithm 1 describes all steps of our algorithm.

One issue with the above algorithm is that coverage is low for high number of suspected users r. It is possible that some items are voted on mostly by flagged users, hence enough information may not be known even interested users may not be recommended that To improve coverage, we ignore only the extreme votes of the flagged users (i.e. maximum vote 5/5 and minimum 1/5); middle votes can be still used to train the right vectors. This removal significantly weakens potential bandwagon attacks as well as average attacks.

4. EXPERIMENTAL RESULTS

As outlined in Algorithm 1, 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 SVD using Hebbian Learning as the baseline Collaborative Filtering algorithm. 20% of the dataset has been randomly taken out and the used as a test set.

Our experimentation strategy involves adding attack profiles generated by well established and standard models, and then applying various CF algorithms on this data. Three metrics are crucial to testing robustness of a CF algorithm: these are *prediction shift*, *hit ratio*, and *Mean Average error*.

Metrics

Prediction Shift measures the change in prediction of the attacked item (before and after attack) of a CF algorithm. This metric is also sensitive to the strength of an attack, with stronger attacks causing larger prediction shift.

$$\mathcal{P} = \frac{1}{N}\sum_u |\grave{v}_{u_i,y} - v_{u_i,y}| = \frac{1}{N}\sum_u \mathcal{P}_u \, , N = \# \text{ users} \quad (11)$$

Hit Ratio measures the effect of attack profiles on top-k recommendations. Since the end effect of a recommender system is a list of items recommended to a particular user, this metric captures the fraction of users affected by shilling attacks. Let $H_{u,i} = 1$ if an item i is a top-k recommendation to user u, and $H_{u,i} = 0$ otherwise. Hit ratio is a fraction between 0–100% and is defined as follows:

$$\mathcal{H} = \frac{100}{N} \times \sum_u \Delta H_{u,i} \, , N = \# \text{ users} \quad (12)$$

Finally, *Mean Average Error* is the overall prediction error on missing values. We measure MAE over the test set, which contains 20% of all votes in the dataset. MAE is commonly used to compare the predictive accuracy of CF algorithms; here, we are interested in finding out the effect of flagging users. Clearly if too many users are flagged, the prediction model is trained with lesser data. A comparison of prediction accuracy of our proposed algorithm with the CF on original data (without attack profiles) provides us a measure of accuracy sacrificed for gain in robustness.

Experimental Strategy

We vary the attack and filler size of the inserted profiles and measure the Mean average 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 effective algorithms can give artificially good results as compared to less accurate CF approaches. We also compare our results with CF run on the original dataset without any inserted spam.[9]

[9]An underlying assumption is that the MovieLens dataset itself has no shilling profiles. This assumption is unverified.

Table 3: Effect of random Shilling attacks on various CF approaches

Attack size	Filler size	SVD-GHA		RMF		PCARobustCF	
		Hit Ratio	Pred Shift	Hit Ratio	Pred Shift	Hit Ratio	Pred Shift
1%	1%	4.90	1.23	2.15	0.96	0.00	0.26
	3%	5.32	1.22	2.21	0.97	0.00	0.25
	5%	5.90	1.23	2.59	0.96	0.00	0.25
	7%	6.57	1.23	2.30	0.95	0.00	0.25
	10%	6.49	1.19	2.08	0.92	0.01	0.33
3%	1%	9.42	1.63	14.34	1.71	0.00	0.25
	3%	12.63	1.69	12.18	1.69	0.00	0.26
	5%	12.71	1.65	11.32	1.61	0.00	0.29
	7%	13.30	1.62	10.91	1.56	0.01	0.35
	10%	12.18	1.55	10.10	1.51	0.24	0.40
7%	1%	22.74	2.07	25.43	2.20	0.00	0.24
	3%	24.07	2.04	23.34	2.10	0.00	0.28
	5%	21.77	1.95	21.17	1.98	0.05	0.31
	7%	19.86	1.85	21.06	1.92	2.46	0.60
	10%	17.28	1.73	20.04	1.53	4.69	0.78
10%	1%	30.54	2.21	30.53	2.29	0.00	0.24
	3%	30.92	2.16	27.72	2.20	0.10	0.35
	5%	26.22	2.04	25.93	2.10	2.74	0.66
	7%	22.70	1.93	25.28	2.03	5.09	0.84
	10%	19.50	1.78	21.60	1.89	7.86	0.99

Discussion

The impact of shilling attacks: Tables 3 and 4 shows the desired results; the proposed Robust CF algorithm performs significantly better than all approaches in all conditions. Our experiments show that VarSelect SVD adds significant robustness over SVD and RMF. For smaller random attacks, VarSelect SVD is virtually unaffected by the presence of attack profiles. Note that the hit ratio is close to perfect (0%) for such attacks, and the prediction shift is less than one-fourth that of SVD and a third that of RMF. Till addition of 450 attackers, VarSelect SVD is almost perfect, suffering only when higher filler rates are used. For huge attack volumes numbering 10% of the entire user population, there is noticible impact. However, this impact is still far less of SVD or RMF. Note that our baseline methods are already significantly better than k-NN and PLSA; thus the improvement due to VarSelect is very significant.

The picture is a little worse for average attacks. Average attacks are stronger than other forms of attack and are also dangerous in small amounts, We note that for large attacks, the impact is higher than for random attacks; prediction shifts increase as the strength of the attack increases. The reason for this is that VarSelect is less accurate for detecting Average attacks; thus shillers who are not flagged continue to have an impact. This is akin to a setup where is no detection and a small number of shillers are present.

We also notice that hit ratios are lower for all attack sizes than SVD & RMF; occasionally however, we notice a negative hit ratio. This indicates some users (say set $mathcalU_s$) who might have been normally recommended the attacked item, may not be recommended that item any more. On detailed examination we observe that this is a result of users in $\mathcal{U}_s$ being *flagged.* Thus recommendations for some users may change with VarSelect SVD due to flagging. We finally note that bandwagon attakcs have simialr results as random attacks, with slightly higher prediction shift; we omit the results due to lack of space.

Table 4: Effect of Average Shilling attacks on various CF approaches

Attack size	Filler size	SVD-GHA		RMF		PCARobustCF	
		Hit Ratio	Pred Shift	Hit Ratio	Pred Shift	Hit Ratio	Pred Shift
1%	1%	8.14	1.25	10.78	1.24	-0.02	0.38
	3%	10.11	1.28	9.09	1.25	-0.14	0.40
	5%	9.93	1.24	7.30	1.22	-0.11	0.40
	7%	10.67	1.24	7.79	1.22	0.14	0.42
	10%	11.10	1.24	6.29	1.19	0.82	0.74
3%	1%	17.16	1.59	18.92	1.61	-0.14	0.41
	3%	19.51	1.61	15.20	1.56	-0.13	0.41
	5%	19.93	1.58	14.87	1.54	0.11	0.43
	7%	19.12	1.55	13.04	1.27	1.74	0.57
	10%	19.50	1.52	13.98	1.49	13.44	1.01
7%	1%	31.24	1.86	24.16	1.85	-0.09	0.40
	3%	32.86	1.85	21.31	1.79	-0.02	0.44
	5%	31.66	1.80	21.60	1.73	6.43	0.75
	7%	30.64	1.78	20.78	1.71	16.25	1.12
	10%	31.76	1.76	21.76	1.68	24.32	1.38
10%	1%	39.57	1.96	29.44	1.96	0.08	0.39
	3%	39.98	1.94	23.71	1.89	0.82	0.37
	5%	38.27	1.90	25.49	1.84	14.11	1.08
	7%	38.20	1.87	24.88	1.79	23.53	1.35
	10%	36.86	1.77	24.39	1.70	31.44	1.48

Predictive performance: We also investigated the predictive performance of various CF algorithms on a held out test set, when shilling attacks are added. Table 6 shows the effect on VarSelect SVD in comparison to other algorithms. We note that all SVD based algorithms in the test perform well and significantly better than k-NN. Also, there is very small departure from the baseline (SVD without attackers); RMF infact outperforms SVD, a result which has also been noted in [12]. VarSelect performs slightly better than SVD, but is less accurate in prediction than RMF. Note however, that all results are in a band of $\pm 1.5\%$ which is not very significant statistically. The important conlcusion is that VarSelect SVD provides additional robust at no additional cost of predictive accuracy.

The impact of parameter r: We are also interested in finding how stable the VarSelect algorithm is to the selection of the number of flagged users r. We fix r to values between 5–90% and run the VarSelect algorithm over a dataset where 350 attackers have been added following an Average Attack model. One caveat though: to improve coverage of prediction, we ignore only extreme vote for the flagged users, thus losing only limited data and not complete data; this is also necessary for numerical stability. We know from previous experiments that a 5% attack is a rather large attack, however we expect the model to be more stable at higher fraction of untrusted users. Fig. 1 & 2 provide experimental proof of this hypothesis. As the nubmer of flagged users is increased, we observe a decreasing hit ratio, with zero additional users being recommended the attacked item. Similarly, the prediction shift also decreases quickly at first, and then stabilizes to a minimum. At higher values of r, we observe a slight increase in prediction shift, this is possibly due to a converge of predicted values towards the user mean, which is what the SVD model predicts in the absense of much data about an item.

Dependence on Dataset: One additional characteristic of the algorithm we explore is data dependence. It is well known that different datasets show different characteristics. Specifically, the small Movielens and large Movielens datasets have different number of users (944 vs 6040), sparsity (6% vs 4%) and average votes per user. Since Mehta et al. report much higher detection rates for average attacks with the smaller ML dataset, we verify if there is indeed such a large difference. Our results (Tab. 5) show that indeed VarSelect works much better on the smaller dataset, and has worse results with more data. This shows that the success of VarSelect is dependant on the characteristics of the dataset; the ranking mechanisms used for Variable Selection may have to be modified based on data. This is a direction for future work.

Table 5: Effect of random Shilling attacks on the smaller MovieLens dataset.

Attack size	Filler size	SVD-GHA		RMF		PCARobustCF	
		Hit Ratio	Pred Shift	Hit Ratio	Pred Shift	Hit Ratio	Pred Shift
5%	7%	18.01	1.46	19.07	1.62	0.00	0.35
	10%	20.41	1.50	17.94	1.61	0.00	0.37
	15%	23.66	1.55	19.17	1.61	0.00	0.38
10%	7%	29.52	1.69	27.47	1.78	0.00	0.37
	10%	30.72	1.73	25.78	1.75	0.00	0.34
	15%	34.11	1.77	24.54	1.77	0.00	0.36
15%	7%	35.45	1.83	30.01	1.86	0.00	0.30
	10%	39.83	1.88	30.19	1.86	0.00	0.33
	15%	41.84	1.88	29.55	1.85	0.04	0.32

The impact of pure noise: Often, a lot of spam is purely junk, with no specific pattern, but random insertion of data. This phenomenon has been observed both with email spam and web spam. We investigated the effect of random noise on the predictive accuracy of SVD, RMF and Varselect. Such attack profiles have a subset of filler items filled with a uniform random generator between 1 and 5. We observe that the predictive accuracy of both RMF and VarSelect SVD is higher than SVD-GHA; SVD-GHA suffers from a 2% increase in MAE on a hidden test set as compared to SVD without any noise. The impact on RMF and Varselect is less than 0.5%, or too little for statistical significance. Notably, attack based on uniform random generators have very low prediction shift and hit ratio.

5. CONCLUSIONS

In this paper, we describe a robust and accurate algorithm for Collaborative filtering which is very stable in the face of shilling attacks. This algorithm combines the detective ac-

Table 6: Overall MAE on a test set for various CF approaches after attack profiles (Average Model) have been inserted.

Attack Size	Filler Size	SVD without attack	SVD	RMF	SVD with VarSelect	Pearson based k-NN
3%	2%	0.6755	0.6760	0.6691	0.6762	0.8095
	5%	0.6755	0.6763	0.6694	0.6768	0.8115
	10%	0.6755	0.6776	0.6699	0.6763	0.8135
5%	2%	0.6755	0.6761	0.6689	0.6764	0.8077
	5%	0.6755	0.6767	0.6699	0.6761	0.8087
	10%	0.6755	0.6783	0.6703	0.6769	0.8093
10%	2%	0.6755	0.6766	0.6719	0.6768	0.8062
	5%	0.6755	0.6774	0.6723	0.6783	0.8062
	10%	0.6755	0.6785	0.6769	0.6778	0.8073

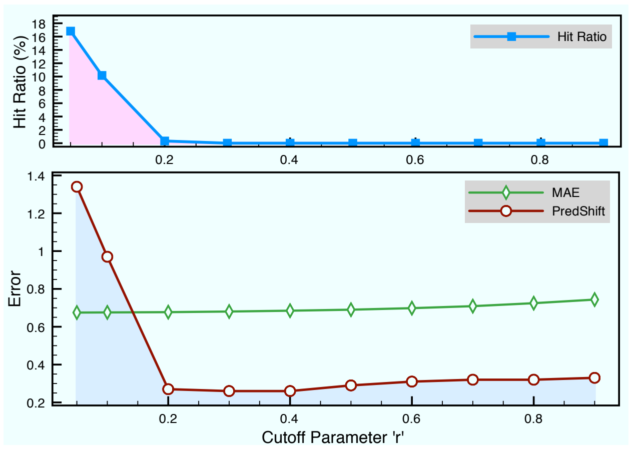

Figure 1: MAE, Pred Shift and Hit ratio for 5% Average Attacks (7% filler) with various values of r.

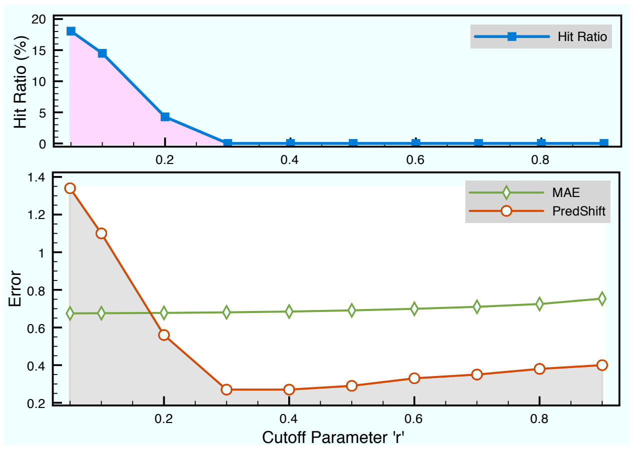

Figure 2: MAE, Pred Shift and Hit ratio for 5% Average Attacks (10% filler) with various values of r.

curacy of previously established detection models based on SVD, and is also extremely accurate. A variety of experiments show that attacks of different strength are rendered much weaker by VarSelect SVD. In addition, the algorithm is highly scalable due to its computational efficiency and use of sparsity.

Directions of future work include improved ranking mechanisms for eliminating users. We also believe that the dimension of time brings important information and has not been exploited in this algorithm; shillings attacks are concentrated in a short period of time as opposed to real users. We also need to devise more accurate ways of detecting the cutoff parameter to save flagged users from any potential impact. Finally, we would like to explore more about the properties of this algorithm e.g. prediction shifts on items related to the attacked item.

6. REFERENCES

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

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. pages 542–547. ACM Press New York, NY, USA, 2006.

[3] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *WIDM '05: 7th annual ACM international workshop*, New York, USA, 2005. ACM Press.

[4] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[6] G. Gorrell. Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In *EACL*, 2006.

[7] H. Hotelling. Analysis of a Complex of Statistical Variables Into Principal Components. 1933.

[8] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[9] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 393–402, New York, NY, USA, 2004. ACM Press.

[10] B. Mehta. Unsupervised shilling detection for collaborative filtering. In *AAAI*, pages 1402–1407, 2007.

[11] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 14–21, New York, NY, USA, 2007. ACM Press.

[12] B. Mehta, T. Hofmann, and W. Nejdl. Robust Collaborative Filtering. In *In Proceedings of the 1st ACM Conference on Recommender Systems*. ACM Press, October 2007.

[13] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Inter. Tech.*, 4(4):344–377, 2004.

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

[15] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Efficient and secure collaborative filtering through intelligent neighbour selection. In *Proceedings of the 16th European Conference on Artificial Intelligence, 22nd–27th*, pages 383–387, Valencia, Spain, Aug 2004. IOS Press.

[16] J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. *Proceedings of the 2007 ACM conference on Recommender systems*, pages 105–112, 2007.

[17] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study, 2000.

[19] B. Webb. Netflix update: Try this at home. `http://sifter.org/~simon/journal/20061211.html`, 2006.

[20] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik. Detection of Obfuscated Attacks in Collaborative Recommender Systems. In *Workshop on Recommender Systems, ECAI*, 2006.

[21] S. Zhang, Y. Ouyang, J. Ford, and F. Makedon. Analysis of a low-dimensional linear model under recommendation attacks. In *SIGIR*, pages 517–524, 2006.