Robust Collaborative Filtering

Bhaskar Mehta
L3S / Universitat Hannover
Appelstrasse 4, 30539
Hannover, Germany
mehta@l3s.de

Thomas Hofmann
Google Inc.
Freigutstrasse 12, 8002
Zurich, Switzerland
thofmann@google.com

Wolfgang Nejdl
L3S / Universitat Hannover
Appelstrasse 4, 30539
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.

Robust statistics is an area within statistics where estimation methods have been developed that deteriorate more gracefully in the presence of unmodeled noise and slight departures from modeling assumptions. In this work, we study how such robust statistical methods, in particular M-estimators, can be used to generate stable recommendation even in the presence of noise and spam. To that extent, we present a Robust Matrix Factorization algorithm and study its stability. We conclude that M-estimators do not add significant stability to recommendation; however the presented algorithm can outperform existing recommendation algorithms in its recommendation quality.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval—*Collaborative Filtering, robust statistics*; G.3 [**Probability And Statistics**]: Robust regression

General Terms

Algorithms

1. INTRODUCTION

Service providers on the World Wide Web operate in a cut-throat environment where even satisfied customers and growth do not guarantee continued success. As users become ever more proficient in their use of the web and are exposed to a wider range of experiences, they are becoming more demanding, and their definition of what constitutes good service is rapidly changing and being refined. Given the user population of the web, it is difficult to come up with a *one size fits all* approach. A successful mechanism to deal with the demands of such a heterogeneous user population is to modify contents, characteristics, or appearance of web based systems with respect to a specific user. This is referred to as *Personalization*, and is distinguished from customization by the use of implicit and assumed preferences.

To measure user perception of personalization and its effectiveness, surveys were conducted by Choicestream in 2004 and 2005. The results of the survey clearly point to the fact that customers realize the value of personalized content; they are willing to spend more effort and money to get a better service customized according to their individual preferences.

The popularity of Recommender Systems has attracted users with malicious intent to bias recommendations in their favor. Other users provide low quality ratings which deviate from the statistical assumptions made by various collaborative filtering algorithms. As a result, there is a danger of producing low quality or faulty outputs from recommender systems which may result in user loosing faith in the system. Recent research has revealed the vulnerability of similarity-based collaborative filtering. While recent algorithms [17, 16] are successful in identifying spam in collaborative filtering, it is desirable to develop algorithms which are robust to spam from the ground up. 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 detection. Robust statistical methods like M-estimators [11] that have been used successfully in statistics provide an alternative approach when data have abnormal entries, e.g. due to outliers.

In this work, we propose a matrix factorization algorithm based on robust M-estimators and compare it with various other algorithms. The resulting algorithm provides more stability against spam than previous approaches, but is outperformed by newer versions of SVD in robustness. However, the predictive performance of our proposed algorithm is better than other robust approaches like PLSA and the newly invented SVD based on Hebbian learning.

2. COLLABORATIVE FILTERING SPAM

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. One example is when a loosely organized group managed to trick the Amazon recommender into correlating a book titled *Six Steps to a Spiritual Life* (written by the evangelist Pat Robertson) with a book for gay men[1].

A lot of web-enabled systems provide free access to users via simple registration processes. 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 [13, 20]. *Profile injection attacks* add a few profiles (say 3% of the total profiles) which need to be identified and protected against. Such attacks have been referred to as *shilling* attacks, a specific form of spam. Profile injection 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 [20]. Research in the area of *shilling attacks* [20] has made significant advances in the last couple of years. Early work identified the threat of shilling attacks and the types of attack (*nuke* and *push*). Various attack strategies were then discovered and appropriate metrics were developed to measure the effectiveness of an attack. Attacks strategies include [18]:

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, and some popular items are rated with the maximum vote.

Note that Gaussian distributions $\mathcal{N}_{\mu,\sigma}$ have been used for generating 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 set of votes is used for random and bandwagon attacks, while the standard deviation of the each individual item is used for the average attack. [16] provide analytical proof for the average attack being the strongest attack[2].

Recent research in this area aimed at finding solutions to detecting profile injection attacks. The earliest spam detection algorithm based on features of spam profiles was invented by Chirita et al. [3]. 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. [18] compare their feature-based classification algorithm which performs significantly better than the Chirita algorithm by taking more features into account. The Mobasher et al. [18] algorithm trains a classifier given enough example spam and authentic profiles and is fairly accurate in detecting spam 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 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. [25] discusses these obfuscation strategies and their effect on detection precision. O'Mahony et al. [21] 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 [17, 16] provide highly accurate detection methods by exploiting the *group effect*: a property of spam users to work together. These recent methods are based on dimensionality reduction and detect spam with high accuracy. However, detection procedures can be applied only sparingly due to their highly computational and batch nature. Our approach in this paper is to use the insight gained in the design of detection procedures to create a robust alternative to SVD.

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

[2]Using exact item means is the actual strongest attack possible.

3. SVD AND ITS VARIATIONS

SVD stands for Singular Value Decomposition; it is a method of factorizing a matrix into two orthonormal matrices and a diagonal matrix. It stems from the Spectral Theorem [12]; SVD is a more general decomposition than Spectral decomposition since it is applicable to rectangular matrices as well. SVD factorizes a rectangular $n \times m$ matrix $\mathbf{D}$ as follows

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\,, \tag{1}$$

where $\mathbf{U}, \mathbf{V}$ are unitary normal matrices and $\mathbf{\Sigma}$ is a diagonal matrix of size $\mathrm{rank}(\mathbf{D}) \leq \min(m, n)$, where $\mathrm{rank}(\mathbf{D})$ 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 $k \leq \mathrm{rank}(D)$ (say $k = 10$), leading to an low rank approximation $\mathbf{D}_k$ of the matrix $\mathbf{D}$.

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

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. [1] provides more details on properties of SVD. The SVD is interesting in the context of many data analysis applications, since real-world data can often be approximated well by a few independent dimensions.

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) and the presence of a large number of zeros can make the computation of SVD very biased towards unobserved values. Initial applications of SVD to CF such as [23] tried to compensate for that by using the overall means for missing values. This approach, though more successful than previous approaches is highly biased towards the used means. In the last decade, there has been significant research on computation of SVD for large and sparse matrices. Significant work has been done in the design of *PROPACK*[3] and *SVDPACK*[4]. However, these approaches do not treat missing values in a principled fashion. [26] discusses the use of the Expectation Maximization [4] procedure to approximate SVD optimally in the log-likelihood sense. However, their approach requires a SVD to be performed at

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

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

each EM iteration, which cannot be scaled to large matrices, since it is improbable that any method which needs more than a few hundred iterations over the entire data can be scaled to large matrices with millions of rows.

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 [22] and calculates SVD by iterating through only observed values. The method has come into the limelight following its use in the *Netflix contest*[5] by a top-10 contestant named Simon Funk [24]. The advantage of this approach is that it uses a simple Hebbian learning rule which is easily expressed in "*two lines of code*" [24]. The method has been found to be highly accurate for CF and scales easily to a matrix with 8.4 billion potential values. Below we describe this approach in detail.

3.1 SVD using Hebbian learning

Gorrell [6] extends an existing method for eigen decomposition to non-symmetric matrices of arbitrary sizes. In her approach, 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

$$\mathbf{D}_{\text{rem}} = \mathbf{D} - \mathbf{u}_1\sigma_1\mathbf{v}_1^{\text{T}} \,, \tag{3}$$

The first eigen-value/vector 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 [9].

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

$$x_{\text{est}} = \hat{u}_i \cdot \hat{v}_j \,. \tag{4}$$

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

$$r(x) = x - x_{\text{est}} = x - \hat{u}_i \cdot \hat{v}_j \,, \tag{5}$$

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

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

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[7] modified this basic algorithm by introducing a weight decay regularization factor as follows [**?**] :

$$\triangle \hat{u}_i = \lambda(\hat{v}_j \cdot r(x) - \kappa \cdot \hat{u}_i) \,, \, \triangle \, \hat{v}_j = \lambda(\hat{u}_i \cdot r(x) - \kappa \cdot \hat{v}_j) \,, \tag{7}$$

where κ denotes the regularization strength. To ensure fewer iterations, he suggests the use of a base estimate using item averages (represented by $\bar{\mathbf{j}}$ for item j) and the average user offset. This gives a good estimate and reduces the number of iterations by a factor of 2.

$$x_{\text{base}}(i, j) = \bar{\mathbf{j}} + \underset{k: x_{i,k} \neq 0}{\text{AVG}} \left\{ x_k - \bar{\mathbf{k}} \right\} \,, \tag{8}$$

Further modifications included clipping the estimated value to the permissible range of values. Clipping makes this SVD approach particular to CF, where data values are discrete value bounded by a minimum and maximum rate (say 1–5).

$$\mathbf{D}_{ij}^{sf} = x_{base}(i, j) \, + \, \sum_k Clip(\hat{u}_{ik} \cdot \hat{v}_{jk}) \tag{9}$$

Where *Clip*() clips the value to between 1 and 5. Other modifications have also been proposed but have been found to have minor benefits. For the NetFlix dataset, $k = 25$–40 has been found optimal. The performance of this simple algorithm is surprisingly good: it performs up to 6% better on the Netflix dataset than the baseline set by NetFlix[8].. We have also experienced similar benefits in performance when running this algorithm on other datasets.

4. OTHER ALGORITHMS FOR CF

In addition to SVD, other CF algorithms have been widely deployed for collaborative filtering. We mention only two here (PLSA and k-NN) for a reason: our objective is to test to robustness of CF algorithms, and these two algorithms have been studied previously. The k-NN algorithm using Pearson's similarity is a popular approach, but was found highly susceptible to spam and serves as the baseline. PLSA on the other hand was found to provide strong robustness to spam [16, 19]. We later compare our proposed algorithm to these two approaches.

4.1 Probabilistic Latent Semantic Analysis

PLSA [8] 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 $\mathbf{z} \in Z = \{z_1, z_2, ..., z_K\}$ with each observation. In the context of collaborative filtering, each observation corresponds to a vote by a user to an item. The space of observations is normally represented as an $M \times N$ co-occurrence matrix (in our case, of M items) $\mathcal{Y} = \{y_1, y_2, .., y_M\}$ and N users $\mathcal{X} = \{x_1, x_2, .., x_N\}$. The aspect model can be described as the following generative model:

- select a data item y from $\mathcal{Y}$ with probability $P(y)$,
- pick a latent factor z with probability $P(z|y)$,
- generate a data item x from $\mathcal{X}$ with probability $P(x|z)$.

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

$$P(x, y|z) = \sum_z P(x, y, z) = \sum_z P(y|z)P(z|x)P(x) \tag{10}$$

This model is based on two independence assumptions: first, observation pairs (x, y) are assumed to be generated independently; second, 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

[5]www.netflixprize.com

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

[7]Brandyn Webb uses a pen name *Simon Webb* in his blog and has registered a NetFlix team with the same name

[8]`http://www.netflixprize.com`

usually interested in predicting the vote for an item for a given user, we are interested in the following conditional model:

$$P(y|x,z) = \sum_z P(y|z)P(z|x) \tag{11}$$

The process of building a model that *explains* a set of observations $(\mathcal{X}, \mathcal{Y})$ is reduced to the problem of finding values for $P(z), P(y|z), P(x|z)$ that maximize the (log)likelihood of the observations. The model parameters $P(z|u)$ and $P(y|z)$ are learnt using the *Expectation Maximization* [4] algorithm which is a standard procedure for latent variable methods. The iterative use of the EM algorithm leads to a conditional probability distribution for y, z which is optimal in the log-likelihood sense. [8] provides more details on the EM procedure for PLSA.

4.2 Pearson based k-NN

Basic collaborative filtering systems use a weighted reconstruction of the votes of users similar to the current user to predict the likely rating for a previously unrated item. Various improvements have been made to the basic mechanism of predicting votes using Pearson's correlation, but they mostly comply to the following scheme: assume the user database consists of a set of votes $v_{i,j}$ corresponding to the vote for user i on item j. The predicted vote for an active user a for item j, $p_{a,j}$ is a weighted sum of the votes of other (similar) users:

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i) \tag{12}$$

where $w(a,i)$ is the weight given to every user i from user active user a, and $\overline{v}_i$ and $\overline{v}_a$ are the average rating given by users i and a, and κ is a normalization factor.

Pearson's Correlation based Collaborative Filtering: The most popular k-NN CF algorithm uses a similarity measure called Pearson's Correlation. This is a standard measure in statistics, that is applied here with only a small modification: similarity is measured based only on items where votes are available for both users. Predicted votes $v(i,j)$ are computed as defined in Eq. (12) with similarity weights $w(a,i)$ defined as follows:

$$w_{\text{pearson}}(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}} \tag{13}$$

Various modifications to this scheme have been proposed in the literature (cf. [7]) which can lead to better coverage and higher accuracy. The principle behind these enhancements is better neighborhood selection and weighting similarity measures by the number of items that are co-voted by pairs of users.

5. ROBUST MATRIX FACTORIZATION

Matrix Factorization aims at learning a low rank approximation of a Matrix $\mathbf{D}$ under certain constraints. This technique is often applied in unsupervised learning from incomplete matrices, and is related to SVD. Formally, the problem is stated as follows: Given a matrix $\mathbf{D}$, find matrix factors $\mathbf{G}$ and $\mathbf{H}$ such that

$$\mathbf{D} \approx \mathbf{GH} \tag{14}$$

In general, MF can be applied to an $n \times m$ matrix to recover $\mathbf{G}_{n \times d}, \mathbf{H}_{d \times m}$, where $d \ll m, n$. Thus MF is a low rank approximation of $\mathbf{D}$ under some cost function. One such cost function is the Frobenius norm:

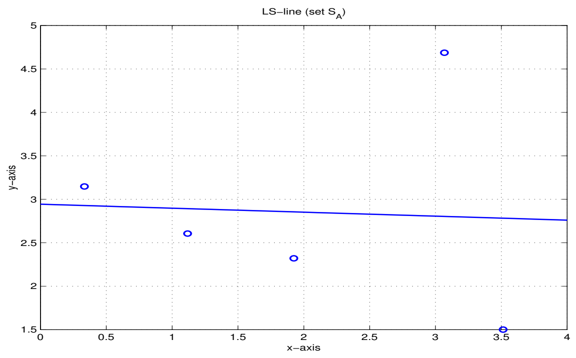

Figure 1: The effect of a single outlier on the least squares estimate.

$$||\mathbf{A} - \mathbf{B}||_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2} \tag{15}$$

Under this cost function, MF reduces to

$$\underset{\mathbf{G},\mathbf{H}}{\text{argmin}} \, ||\mathbf{D} - \mathbf{GH}||_F, \tag{16}$$

which is a formulation that is equivalent to the SVD, if singular values are absorbed appropriately into the left and right singular vectors.

$$\mathbf{D} = \mathbf{U\Sigma V}^T$$
$$\mathbf{D} = \mathbf{GH}, s.t.\ \mathbf{G} = \mathbf{U\Sigma}^{1/2},\ \mathbf{H} = \mathbf{\Sigma}^{1/2}\mathbf{V}^T$$

Under other cost functions, MF takes a slightly different form. Assume GH_{ij} is the (i,j)-th element of the matrix $\mathbf{GH}$. Then, for a real valued even function ρ, the MF problem is restated as

$$\underset{\mathbf{G},\mathbf{H}}{\text{argmin}} \sum_{ij} \rho(D_{ij} - GH_{ij}) \tag{17}$$

The formulation $r_{ij} = D_{ij} - GH_{ij}$ has also been used in the literature. r_{ij} is known as the *residual* of the fit. Clearly, if $\rho(0) = 0$, the above minimization has a lower bound of 0, when $\mathbf{D} = \mathbf{GH}$. The *least square* formulation corresponds to $\rho(x) = x^2/2$.

5.1 Robust approximation using M-estimators

In many real world scenarios, the observed matrix $\mathbf{D}$ is prone to erroneous values. In addition to some small noise, some values may be out of range, or unexpectedly different from the rest of the observations. Such values are typically called outliers; note that we are assuming outliers at a cell level, meaning individual observations D_{ij} might be faulty, with completely arbitrary values and random distribution of cells. Least squares estimates have been shown to be highly error prone to outliers [11]: even 1–2 erroneous values can completely disrupt the approximation. Fig. 1 shows the effect of one outlier on a linear least square estimator. A lot of research has been done in the last 35 years on the topic of robust regression. The theory suggests that minimizing the squared residual is not stable: instead a function of the residual should be minimized. This is done by the use of

M-estimators which use bounded real valued functions ρ

$$\underset{\theta}{\text{argmin}} \sum_{ij} \rho(r_{ij}) \tag{18}$$

where θ representing the model fitting parameters. Let us assume that ρ is a differentiable function, and its derivative is represented by ψ. The minimization of the above function w.r.t. the model parameters θ occurs when the derivative of the above equation is zero, i.e.

$$\sum_i \psi(r_i) \frac{\partial r_i}{\partial \theta} = 0 \ , \tag{19}$$

$\psi(x)$ is called the *influence function* of the M-estimator ρ and models the influence of a residual on the model fitting. It is postulated that robustness requires a bounded influence function. Clearly, ordinary least squares, which has an unbounded influence function ($\psi_{LS}(x) = x$) is non-robust following this criteria. To further simplify, let us define a weight function $w(x) = \psi(x)/x$. Then Eq. 19 becomes:

$$\sum_i w(r_i) r_i \frac{\partial r_i}{\partial \theta} = 0, \tag{20}$$

which is exactly the same condition required for solving the following iterative re-weighted least square problem:

$$\underset{\theta}{\text{argmin}} \sum_i w(r_i^{k-1}) {r_i}^2 \tag{21}$$

The final issue remaining is the choice of an M-estimator: various M-estimators have been described in literature, with Huber, Andrew and Tukey estimators being more popular. Huber's M-estimator [11] is recommended for general purposes and is characterized by the following weight function:

$$w(r) = \begin{cases} r \leq k & 1\,, \\ r > k & \frac{k}{|r|} \end{cases} \tag{22}$$

In Eq. (22), k is an appropriately chosen constant. For our application, we choose $k = 1.345$, a value reported to work well [11] for normally distributed data with std. dev. $\sigma = 1$[9]. The influence function of the Huber M-estimator is bounded by $|k|$. The Huber weight function also has distinct computation advantages over other M-estimators; its application results in a dampened effect of large errors, providing more robustness. In case of spam meant to cause large deviations in one item's predicted value, we expect robust regression to discourage large shifts and provide a moderate estimate.

5.2 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. Previous work shows that Matrix fitting problems can be performed in a similar manner using an Alternating fitting scheme. Assume we want to find the rank over factors $\mathbf{G}_1, \mathbf{H}_1$ as defined in Eq. 17, with the Huber M-estimator; higher rank estimates can be easily computed in a similar manner to SVD(see Sec. 3.1). For a rank 1 solution where $\mathbf{G}, \mathbf{H}$ are both vectors, the broad outline is as follows: first, we initialize $\mathbf{G}_1, \mathbf{H}_1$. Then we fix $\mathbf{G}_1$ and minimize the reweighed least square problem:

$$\underset{H_k}{\text{argmin}} \sum_{ij} w(D_{ij} - G_k H_k) \cdot (D_{ij} - G_k H_k)^2 \tag{23}$$

[9]Standard deviation in our dataset is 1.118 ($\sim$ 1)

This can be achieved by a fixed rate gradient decent algorithm, where updates are performed as follows:

$$G_i^{k+1} = G_i^k + \eta \cdot r_{ij}^k \cdot w(r_{ij}^k) \cdot H_j \ , \forall D_{ij} > 0 \tag{24}$$

Note that we use the function $r_i j$ to denote the residual at $D_{i,j}$. After a few iterations, $\mathbf{G}_1$ converges to a minimum. At this point, we switch $\mathbf{G}_1$ and $\mathbf{H}_1$, and minimize for $\mathbf{G}_1$. The above scheme is known as *Iteratively Re-weighted Least Squares* and was proven to converge to the rank-1 least squares estimate of the matrix [5]. For higher rank matrices, the above routine is repeated for the matrix $\mathbf{D}^k = \mathbf{D} - \mathbf{G}^k\mathbf{H}^k, k = 1, ..., d,$ to get a k-rank estimate. Algorithm 1 summarizes the above procedure.

Algorithm 1 Rank-1-estimate ($\mathbf{D}_{n\times n}$)

```
1: Initialize $\mathbf{G}^0, \mathbf{H}^0, k \leftarrow 1$.
2: Define $r_{ij} = D_{ij} - (G^k H^{k-1})_{ij}$.
3: Solve for $\underset{\mathbf{G}^k}{\text{argmin}} \sum_{ij} w(r_{ij})(r_{ij})^2$
4: Solve for $\underset{\mathbf{H}^k}{\text{argmin}} \sum_{ij} w(r_{ij})(r_{ij})^2$
5: $k \leftarrow k + 1$
6: Iterate steps 2, 3 and 4 till convergence.
7: $\mathbf{G}_1 = \mathbf{G}^k, \mathbf{H}_1 = \mathbf{H}^k$
```

Output: Matrices $\mathbf{G}_1, \mathbf{H}_1$

Algorithm 2 Rank-K-estimate ($\mathbf{D}_{n\times m}, K$)

```
1: Initialize $\mathbf{G} \leftarrow \mathbf{0}_{n\times k}, \mathbf{H} \leftarrow \mathbf{0}_{k\times m}, k \leftarrow 1$.
2: Define $\mathbf{D}_{rem}^k = \mathbf{D}$
3: while $k \leq K$ do
4:     $\mathbf{g}, \mathbf{h} \leftarrow$ Rank-1-estimate($\mathbf{D}_{rem}^k$)
5:     $\mathbf{G}(:, k) \leftarrow \mathbf{g}$, $\mathbf{H}(k, :) \leftarrow \mathbf{h}$
6:     $\mathbf{D}_{rem}^{k+1} \leftarrow \mathbf{D}_{rem}^k - \mathbf{G}(:, k)\mathbf{H}(k, :)\cdot$
7:     $k \leftarrow k + 1$
8: end while
```

Output: Matrices $\mathbf{G}, \mathbf{H}$, Residual error= $\|\mathbf{D}_{rem}^k\|$

Related Work

Robust statistics have been applied previously to SVD [15] using L-estimators. This approach uses Alternative least squares with a L_1 minimization. RANSAC based methods have also been developed for SVD [14]. There is plenty of work in the application of robust statistics to regression [10, 5] and least square estimates. However, all the above approaches for SVD have been designed with full matrices in mind. Moreover, the objective in the work above to deal with numerically large outliers. In our domain, the erroneous values are still in the permissible range; however their effect to cause large deviations which we want to guard against. The use of M-estimators for Matrix factorization is novel to the best of our knowledge; the RMF approach outline above is also meant to work with large and sparse matrices.

6. HYPOTHESIS & EXPERIMENTS

The aim of our work is to test whether robust statistical methods can be used to robustify collaborative filtering. The RMF method outlined in Sec. 5 should withstand profile injection attacks in order to be useful. To test this hypothesis, we apply RMF to CF data and compare the performance with the prediction accuracy after insertion of attack profiles. To insert attack profiles, we use the average attack model [19] and generate a certain percentage of profiles. These profiles collate to attack a single item, which is decided earlier. To choose items to attack, we use the following filter: an item which has not been voted by more than 5% of the user population and has an average vote of less than 3 (since our data set has votes between 1–5). We then vary the number of profiles inserted and the number of items voted by the spam user (filler size). All measurements of error are made on 10% of the original data which has not been used for training/prediction; this is called the *test set*. This methodology is standard and been used to measure the effectiveness of spam attacks previously [19, 18, 17, 16]. We apply the same procedure to PLSA, SVD, and k-NN for comparison.

In addition, we try a simple heuristic where we remove some of the user votes. Since extreme votes are the ones responsible for maximum deviation in case of attacks, we remove 10% of the highest and lowest votes of each person. We expect this heuristic to remove a large fraction of the votes on an attacked item from spam profiles, leading to a reduced prediction shift. Obviously, we expect the overall prediction accuracy to decrease for CF methods: however, it is possible that better methods can generalize well even from lesser data and not lose accuracy significantly.

6.1 Experimental Setup

To evaluate the performance of our proposed algorithms algorithm, we use the 1 million Movielens dataset which consists of 1,000,209 votes by 6040 users over 3952 movies and 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, Random attacks, as well as Gaussian and uniform noise.Since Average attacks tend to be the strongest, we present results only for them We use the generative models explained in [2] to generate these shilling profiles. A random 10% votes are removed from the dataset to create the test set; the training set then contains 900,209 votes to which spam votes are added. We add attack profiles with filler sizes of 3%, 5%, 7%, 10%, and 25%: the number of attack profiles ranges from 1% to 10%. Since adding a user profiles has a high human cost, we find addition of more profiles improbable.

6.2 Metrics Used

The task of evaluating predictions in collaborative filtering is easily described as the measurement of the *deviation* from observed values. Given that the user database can be compactly represented as a Matrix $\mathbf{X}$, with a user u_i forming a row with m items, the objective is to predict missing values in this matrix. Since only a small percentage of the matrix is observed, a portion of the observed data is artificially removed, and predicted using the remaining values. To measure the success of the prediction task, metrics which capture deviation from actual values are used. These include the *mean* and *root mean error*. An additional metric called the *ranking score* rates the ranking generated by the predicted user votes.

1. *Mean Average Error* $= \frac{1}{m}|p_v - a_v|$, where p_v is the predicted vote and a_v is the actual vote. The average is taken only over known values (assume the active user has provided m votes).

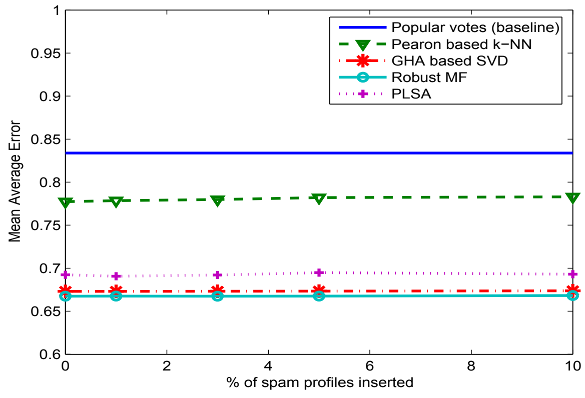

Figure 2: MAE of various CF algorithms compared to RMF measured over the testset: Attack profiles are inserted into the data and MAE is measured over the same testset. Interestingly, insertion of Gaussian spam does not have a significant effect on the overall MAE

2. *Root Mean Average Error* $= \sqrt{\frac{1}{m}|p_v - a_v|^2}$, where p_v is the predicted vote and a_v is the actual vote. The average is taken only over known values (assume the active user has provided m votes). This metric is useful in finding out the ability of a CF algorithm to generalize and highlights larger errors.

3. *MAE on attacked item* To measure the effect of the attack on prediction stability, we compute the mean average error of predicted votes of the attacked item in the test set. This is usually a small number of votes (say 40-100), and indicates the *real* shift in prediction. We prefer this over prediction shift, as it is difficult to compare multiple methods using prediction shift: a common baseline cannot be established, as the base performance of every method (before attack) is different. We measure the MAE after attack over multiple runs and present average results.

7. EXPERIMENTAL RESULTS

Our experiments show that the effect of targeted spam on the performance of various CF algorithms ranges from moderate to strong. The most robust algorithm turns out to be Webb's SVD, followed by RMF and PLSA (see Fig. 3 & 4). The k-NN is easily influenced even when we set the neighborhood size to be 5% of the entire user population (300 neighbors). This is due to two reasons: Spam users generated using the average attack can penetrate user neighborhoods very effectively; secondly, the attacked items chosen by us are voted on by very few users ($<$ 5%), therefore the votes of the spam users become highly authoritative. SVD on the other hand does not get influenced so easily since the factors representing a user and an item are learnt from the overall pattern. Since a significant portion of the user community seems to have a below-average opinion of the attacked item, the effect of spam is lesser than for k-NN. [16] discusses the impact of spam on PLSA and concludes that the stability of PLSA against spam is due to the soft-clustering nature. This applies to SVD as well since it is similar in nature to PLSA. The use of various CF specific optimizations such as clipping leads to a better fitting model. At large filer

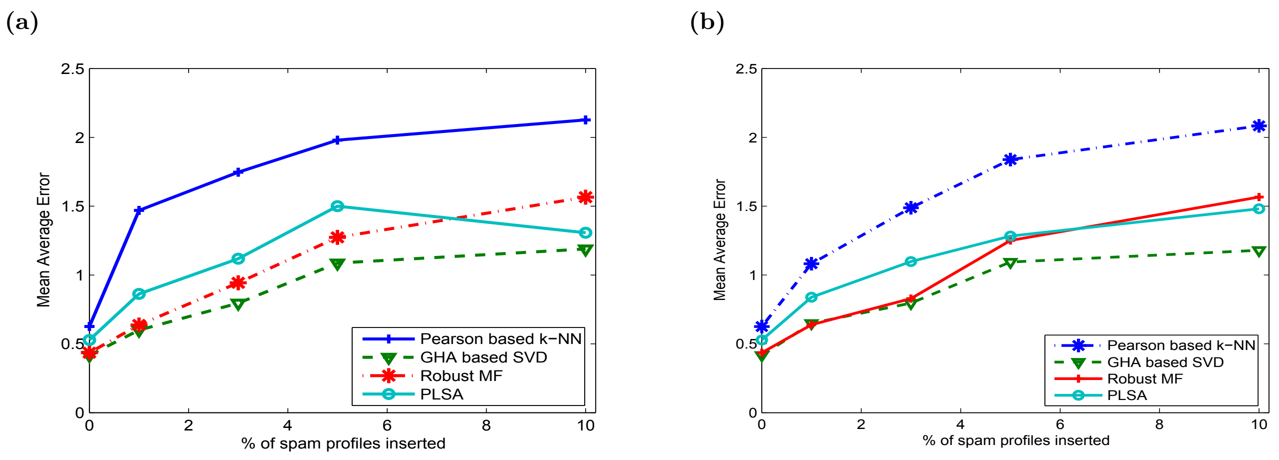

Figure 3: MAE of various CF algorithms on votes in the test set on the attack item a) with filler size=3%, b) with filler size=5%

sizes, k-NN appears to be more stable since the randomness in attack profiles lead to lower correlation; hence the effect of spammers is reduced. This trend has also been noted by previous research [19].

Our proposed Robust Matrix factorization algorithm also performs well in the face of moderate spam. Clearly, the effect of spam is low at small attack sizes, as the majority opinion is given more importance. However, once the number of votes by spammers are more than actual users, RMF starts treating the spammer's view as the majority opinion. The numbers also show that RMF is more tolerant to spam and model deviations than SVD and pLSA: the prediction accuracy of RMF is higher than any other method (see Fig. 2); this trend continues even in the face of spam attacks. Clearly, using robustness offers protection against minor departures from model assumptions.

Removing votes from data: An interesting trend appears when we remove 20% of the extreme votes from each user[10]: all collaborative filtering algorithms test show increased stability w.r.t. prediction shift. Table 1. shows that the accuracy of all methods over the test set votes of the attached item is increased by more than 10%. This clearly comes with a loss in the overall accuracy; however SVD and RMF do not suffer significant losses. The MAE of the SVD, RMF and PLSA remains close to the value without any vote removal, while gaining significant accuracy on the attached item. Particulatly notable is the performance of RMF which gains more than 25% in MAE, outlining how effective it is in learning trends from less and noisy data.

	All data		80% data	
	MAE	**MAE on Attacked item**	MAE	**MAE on Attacked item**
k-NN (1%)	0.7965	*1.4179*	0.8065(-1.2%)	*1.1014*(22.3%)
SVD (1%)	0.6731	*0.6669*	0.7018(-4.2%)	*0.5471*(17.9%)
RMF (1%)	0.6677	*0.6721*	0.6982(-4.5%)	*0.5836*(13.2%)
PLSA (1%)	0.6938	*1.1717*	0.7246(.4.4%)	*0.6840*(41.6%)
k-NN (3%)	0.7992	*1.5268*	0.8074(-1.0%)	*1.2178*(20.2%)
SVD (3%)	0.6733	*0.7625*	0.7013(-4.2%)	*0.6726*(11.8%)
RMF (3%)	0.6681	*0.8806*	0.6987(-4.6%)	*0.6523*(25.9%)
PLSA (3%)	0.6943	*1.1683*	0.7295(-5.1%)	*0.8455*(27.6%)
k-NN (5%)	0.8088	*1.6149*	0.8067(+0.2%)	*1.5198*(5.9%)
SVD (5%)	0.6737	*1.0882*	0.7004(-3.9%)	*0.9338*(14.2%)
RMF (5%)	0.6684	*1.2514*	0.6980(-4.4%)	*0.8759*(30.0%)
PLSA (5%)	0.6946	*1.4995*	0.7271(-4.7%)	*1.1900*(20.6%)
k-NN (10%)	0.8076	*1.8031*	0.8039(+0.4%)	*1.493*(17.2%)
SVD (10%)	0.6736	*1.2659*	0.6998(-3.9%)	*1.2811*(-1.2%)
RMF (10%)	0.6691	*1.5549*	0.6985(-4.4%)	*1.2310*(20.8%)
PLSA (10%)	0.6969	*1.2589*	0.7292(.4.7%)	*1.6346*(-29.8%)

Table 1: MAE of various CF algorithms on votes in the test set on the attack item, with filler size=7%. 20% of extreme votes has been removed for all users. We also report the MAE on the observed votes on the attacked item in the test set ($\sim 40 - 80$ votes)

8. CONCLUSIONS

This paper investigates the effectiveness of robust statistics in protecting against collaborative filtering spam. We present a new algorithm for Robust Matrix Factorization similar in spirit to SVD which is more stable to noisy data. Experimental results show that application of M-estimators does not add significant stability; modified SVD algorithms outperform RMF in robustness. However, 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 our dataset. However, the addition of robustness comes with a price: the RMF algorithm requires 4 times as much training time at SVD. This is a result of our training procedure which uses a fixed rate gradient descent approach; faster training can be achieved by using methods to accelerate gradient descent.

In addition, we have explored the effectiveness of vote sampling on stability and performance; removal of 20% of extreme votes leads to a significant increase in robustness for every method. Whiel some methods suffer from a significant loss in accuracy due to lesser data, SVD and RMF can generalize well even from reduced data and provide accurate prediction. Future work involves developing faster training procedures for RMF and developing algorithms which provide higher robustness against spam.

[10]Only users with more than 15 votes in the training test are selected for vote removal.

9. REFERENCES

[1] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *STOC*, pages 619–626, 2001.

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. 2006.

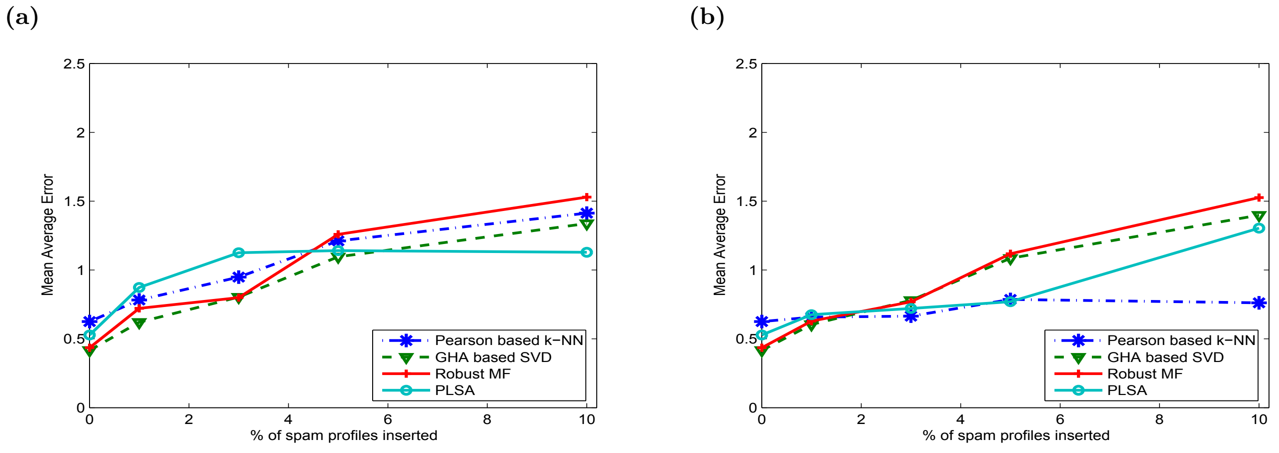

Figure 4: MAE of various CF algorithms on votes in the test set on the attack item a) with filler size=10%, b) with filler size=25%

[3] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of WIDM '05*, pages 67–74. ACM Press, 2005.

[4] 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.

[5] K. Gabriel and S. Zamir. Lower Rank Approximation of Matrices by Least Squares with Any Choice of Weights. *Technometrics*, 21(4):489–498, 1979.

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

[7] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, 2002.

[8] T. Hofmann. Latent Semantic Models for Collaborative Filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

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

[10] P. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[11] P. Huber. *Robust Statistics*. Wiley-IEEE, 2004.

[12] I. T. Jolliffe. *Principal Component Analysis (2nd Edition)*. Springer, 2002.

[13] 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. ACM Press, 2004.

[14] X. LI, Z. NING, and L. XIANG. Robust 3D Reconstruction with Outliers Using RANSAC Based Singular Value Decomposition. *IEICE Transactions on Information and Systems*, 88(8):2001, 2005.

[15] L. Liu, D. Hawkins, S. Ghosh, and S. Young. Robust singular value decomposition analysis of microarray data. *Proceedings of the National Academy of Sciences*, 100(23):13167–13172, 2003.

[16] B. Mehta. Unsupervised shilling detection for collaborative filtering. In *AAAI 2007: Proceedings of the 22nd Twenty-Second Conference on Artificial Intelligence*, AAAI Press, Vancouver, Canada, 2007.

[17] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and Propaganda: Detecting spam users in Collaborative Filtering. In *IUI '07: Proceedings of Intelligent user interfaces*, pages 14–21. ACM Press, 2007.

[18] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. 2006.

[19] B. Mobasher, R. D. Burke, and J. J. Sandvig. Model-based collaborative filtering as a defense against profile injection attacks. In *AAAI*, 2006.

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

[21] M. P. O'Mahony, N. J. Hurley, and Silvestre. Detecting noise in Recommender system databases. In *Proceedings of Intelligent User Interfaces (IUI'06)*, pages 109–115. ACM Press, 2006.

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

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

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

[25] 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.

[26] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *Proceedings of CEC '05*, pages 257–264, Washington, DC, USA, 2005. IEEE Computer Society.