A Robust Collaborative Filtering Algorithm Using Ordered Logistic Regression

Shanshan Zheng, Tao Jiang and John S. Baras
Institute for Systems Research
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD, 20742
Email:{sszheng,tjiang,baras}@umd.edu

***Abstract*—The Internet offers tremendous opportunities for information sharing and content distribution. However, without proper filtering and selection, the large amount of information may likely swarm the users rather than benefit them. Collaborative filtering is a technique for extracting useful information from the large information pool generated by interconnected online communities. In this paper, we develop a probabilistic collaborative filtering algorithm, which is based on ordered logistic regression and takes into account both similarities among the users and similarities among the items. We make inference with maximum likelihood and Bayesian frameworks, and propose a Markov Chain Monte Carlo based Expectation Maximization algorithm to optimize model parameters. The power of our proposed algorithm is its extensibility. We show that it can incorporate content and contextual information. More importantly, it can be easily extended to include the trustworthiness of users, thus being more robust to malicious data manipulation. The experimental results on a real world data set show that our proposed algorithm with the trust extension is robust under different types of attacks in recommendation systems.**

I. Introduction

The advancement of information technology has been rapidly integrating the physical world where we live and the online world that we rely on for retrieving and sharing information. Tremendous amount of information is being generated by web users everyday for sharing in various online communities or social networks. For example, Amazon users post their opinions on different sellers and products they have experienced, Netflix users provide their critics at movies they have watched, and Facebook users browse through their friends' spaces for helpful advices. Users could benefit greatly from peer provided information for better decision making, but due to the large amount and varying quality of the available information, it is necessary to filter out those unrelated or even misleading information and extract only the most useful information. Collaborative filtering is a promising technique to address this challenging problem.

In this paper, we present a probabilistic collaborative filtering algorithm based on ordered logistic regression and consider both the similarities among the users and the similarities among the items. To estimate the model parameters, we propose to use a Markov Chain Monte Carlo (MCMC) based Expectation Maximization algorithm. Our model is very flexible in the sense that it can be extended to include information other than users' ratings on items. For instance, content and contextual information can be naturally combined into the model to improve the prediction accuracy. In this paper, we are particularly interested in incorporating trust in the model. The widespread use of collaborative filtering and anonymity lead to user feedback of varying qualities. While some users express their true opinions faithfully, some may provide noisy ratings which can deteriorate the performance of collaborative filtering. Furthermore, there may exist malicious users who intentionally manipulate the ratings to achieve economic advantages or other purposes. It is believed that incorporating trust into collaborative filtering system greatly improves its robustness. Therefore, we extend our model to take into account the trustworthiness of different users and put different weights on the information provided by them. We also provide the performance of our algorithm against different types of malicious attacks.

The rest of this paper is organized as follows. We review existing collaborative filtering algorithms and previous work on trust in recommendation systems in Section II. The probabilistic model is proposed in Section III. Section IV provides our Markov Chain Monte Carlo based expectation maximization algorithm for estimation of model parameters. Then we discuss several possible extensions of our model in Section V, with focus on the one with the trust weights. Experimental results of our algorithm are presented in Section VI. Section VII draws the conclusions.

II. Related Work

A. Collaborative Filtering Algorithms

A number of collaborative filtering algorithms have been developed in the past, from memory based [1][2] to model based methods [3][4]. In memory based methods, one user's rating on a particular item is estimated as an weighted aggregation of the ratings of some other users (called neighbors) for the same item. This kind of methods usually requires all ratings, items and users stored in memory. In model based methods, the rating is predicated based on a model learned from training data. These modeling techniques include the generalized probabilistic latent semantic analysis [3], the maximum entropy model [5], combinations of multinomial mixture and aspects models [6], etc. Since the models can be trained offline, these algorithms usually scale better than the memory based algorithms for real world applications.

B. Trust in Recommendation Systems

Several existing work on collaborative filtering has considered utilizing trust to increase system robustness. For instance, Filmtrust [7] and MoteTrust [8] assume the trust values are explicitly expressed by users and use a breadth first search in the trust network to compute a prediction. The unknown rating is estimated by aggregating the ratings from other users weighted by their trust values. The assumption of the existence of an explicit trust network may limit their applications. O'Donovan et al. [9] proposed algorithms for computing trust on profile level and item level without such explicit trust networks. Profile level trust is the percentage of correct recommendations that this user has contributed. Item level trust is similarly defined on a specific item. But the two trust metrics are designed in such a way that they are only appropriate in simple memory based methods for recommendation. In contrast, our method defines a different trust metric that can be combined into the much more developed probabilistic approach.

III. Collaborative Filtering Model

A. Problem Statement And Notations

Collaborative filtering is a process of predicting the preference of one user based on the preferences of a group of users. We define $\mathcal{X} = \{X_1, \ldots, X_n\}$ as the set of users, $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$ as the set of items, and V as an $n \times m$ matrix with V_{ij} representing the rating of user X_i for item Y_j. Ratings in a collaborative filtering system may have different forms, such as binary ratings of agree/disagree or good/bad, and scalar ratings which consist of either numerical ratings or ordinal ratings [10]. In this paper, we are focusing on numerical ratings, where V_{ij} is either quantized to a small number of response levels or being null which means the rating is not available. Without loss of generality, we denote user $X_1 \in \mathcal{X}$ as the active user and define $\mathcal{Y}_p = \{Y_1, \ldots, Y_p\} \subset \mathcal{Y}$ as the subset of items for which the preferences of user X_1 are to be predicted. In other words, we are interested in the conditional probability $\Pr(V_{1i} = l | X_1, Y_i \in \mathcal{Y}_p)$, where $l \in \{1, \ldots, L\}$ is the possible response levels (ratings) of X_1.

B. Model Description

We consider collaborative filtering as an inference problem on a bipartite graph where the links represent the ratings from users to items. To begin with, user X_1 needs to determine the available ratings that are relevant for the predication. We say two ratings are *related* if they involve the same user or the same item. For example, V_{1i} and V_{1j} are related, because the more similar items Y_i and Y_j are, the closer V_{1i} and V_{1j} will be. Also V_{1i} and V_{ki} are related as they are ratings for the same item Y_i and the more alike users X_1 and X_k are, the closer V_{1i} and V_{ki} will be. In other words, user X_1 would need the part of the rating information from the bipartite graph as shown in Figure 1, where $\mathcal{Y}_r$ is the set of items that X_1 has rated and the items for which X_1 wishes to predict its preference, i.e., $\mathcal{Y}_r = \{Y_j : V_{1j} \neq \text{null}\} \cup \mathcal{Y}_p$, and $\mathcal{X}_r$ is the set of users who have rated some of the items in set $\mathcal{Y}_r$, i.e., $\mathcal{X}_r = \{X_i : \exists Y_j \in \mathcal{Y}_r, V_{ij} \neq \text{null},\ i \neq 1\}$.

Inspired by the latent space method in [11], we introduce unit-length *hidden vectors* representing user characteristics

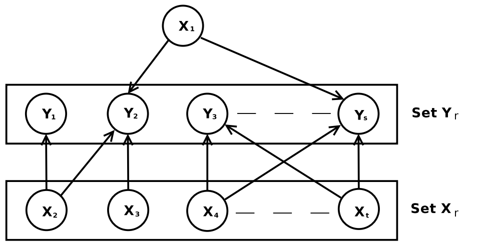

Fig. 1: Required Rating Informatin for X_1

and item characteristics, to model the dependencies among the ratings in Figure 1. We denote the characteristic vector for user X_i and item Y_j by $\boldsymbol{x_i}$ and $\boldsymbol{y_j}$ respectively. Since all information needed to predict V_{ij} can be modeled into $\boldsymbol{x_i}$ and $\boldsymbol{y_j}$, if we denote the model parameter as $\boldsymbol{\theta}$, the log-likelihood can be written as $\log \Pr(V | X_1, \mathcal{X}_r, \mathcal{Y}_r, \boldsymbol{\theta}) = \sum_{i,j} \log \Pr(V_{ij} | \boldsymbol{x_i}, \boldsymbol{y_j}, \boldsymbol{\theta})$, where $i \in \{i : X_i \in \mathcal{X}_r \cup \{X_1\}\}$ and $j \in \{j : Y_j \in \mathcal{Y}_r\}$. Furthermore, we model X_i's preference on Y_j to depend on the scalar product of the characteristic vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, i.e., we transform the users and the items to the same hidden characteristic space so they are directly comparable. We can imagine $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ as points on a sphere of unit radius, and they are better matched if the angle between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ is smaller. A convenient way to parameterize $\Pr(V_{ij} | \boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$ is by ordered logistic regression [12]. We define $\gamma_{ij}^l = \Pr(V_{ij} \leq l | \boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta}),\ l = 1, \ldots, L-1$, and let

$$\eta_{ij}^l = \log \frac{\gamma_{ij}^l}{1 - \gamma_{ij}^l} = \theta_l - \boldsymbol{x}_i \cdot \boldsymbol{y}_j, \tag{1}$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{L-1}]$. The reason to use the log-odds scale $\log \gamma_{ij}^l / (1 - \gamma_{ij}^l)$ on the cumulative probabilities γ_{ij}^l but not on the individual probabilities $\Pr(V_{ij} = l | \boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$, is to make the model invariant under the grouping of the adjacent response levels [12].

The model presented so far assumes that all users express their ratings on a common scale, i.e., when they have the same preference with respect to the items, they give the same ratings. However, different users may have different opinions upon the meanings of the ratings, e.g., a rating value of 5 out of 10 point scale may mean differently to different people. Our model can take into account this factor by adding one parameter for each user to allow user-level variability. More specifically, we assign each user characteristic vector $\boldsymbol{x}_i$ with a scalar parameter a_i and let the parameter η_{ij}^l in the model be $\theta_l - a_i \cdot \boldsymbol{x}_i \cdot \boldsymbol{y}_j$. Therefore, even if the angle between the characteristic vectors of two users is 0, they can still give different ratings for the same item, i.e., the one associated with larger a_i gives a higher rating. We name this process as user normalization, and will investigate its effect in Section VI.

IV. Maximum Likelihood Estimation

We use Maximum Likelihood (ML) estimation to estimate the model parameter $\boldsymbol{\theta}$ and the hidden characteristic vectors. We treat the hidden vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ and $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\}$ as missing data, then the Expectation Maximization (EM) algorithm provides a natural approach to obtain the

ML estimations for the parameters in the model. To make the problem tractable, we assume that the unknown characteristic vector $\boldsymbol{x}_i$ is drawn from a mixture of G multivariate normal distributions with spherical covariance matrices, i.e.,

$$\boldsymbol{x}_i \sim \sum_{g=1}^{G} \lambda_g \mathcal{N}_d(\mu_g, (\sigma_g)^2 I_d).$$

where $\mathcal{N}_d$ represents a d dimensional multivariate normal distribution and I_d is the d dimensional identity matrix. The choice of spherical covariance matrices is due to the fact that the likelihood is invariant to rotations of the characteristic space so the model should be invariant to rotations of the coordinate system. In order to reduce the effect of the prior on the posterior, we assume the following conjugate priors for the parameters λ_g, μ_g and σ_g,

$$\begin{aligned}(\lambda_1, \ldots, \lambda_G) &\sim \text{Dirichlet}(\nu_1, \ldots, \nu_G),\\ \mu_g &\sim \mathcal{N}_d(\mu_0, \omega_g^2 I_d),\ g = 1, \ldots, G,\\ \sigma_g^2 &\sim (\sigma_0)^2 \text{Inv}\chi^2_{\alpha_g},\ g = 1, \ldots, G,\end{aligned}$$

where ν_g, μ_0, ω_g, σ_0 and α_g are hyper-parameters to be specified, and $(\sigma_0)^2\text{Inv}\chi^2_{\alpha_g}$ is the scaled inverse chi square distribution with scale parameter $(\sigma_0)^2$ and freedom degree α_g. The item characteristic vector $\boldsymbol{y}_j$ is also assumed to be drawn from a mixture of Q multivariate normal distributions.

However, directly computing the conditional expectation of $\log(\Pr(V, X, Y|\boldsymbol{\theta}))$ is still complicated, although we have made the mixture Gaussian assumptions on $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$. We propose to use the Markov Chain Monte Carlo (MCMC) methods [13] to obtain an approximation of this expectation value. The EM algorithm with MCMC works as follows:

- *E-step with Markov Chain Monte Carlo:* draw a set of random observations $\{X, Y\}$ from their conditional distribution $\Pr(X, Y|V, \boldsymbol{\theta}^t)$ by a Markov Chain Monte Carlo method, given the observed data V and current value of parameter $\boldsymbol{\theta}^t$, then use these random observations to approximate the expectation of the log-likelihood $E_{X,Y|V,\boldsymbol{\theta}^t}[\log(\Pr(V, X, Y|\boldsymbol{\theta}))]$.
- *M-step:* obtain $\boldsymbol{\theta}_l^{t+1}$ by maximizing the expectation of the log-likelihood obtained in the E-step.

A. Expectation Step with Markov Chain Monte Carlo

The idea of the MCMC methods [13] is to construct a Markov chain that has the desired distribution as its *equilibrium distribution* and then use the states of the chain after a large number of steps as samples from the desired distribution. Gibbs sampling and Metropolis-Hastings (M-H) algorithm are two common MCMC methods.

1) Gibbs Sampling [13]: Gibbs sampling is usually applied when the joint distribution is not known explicitly, but the conditional distribution of each variable is easy to obtain. It generates a sample from the distribution of each variable in turn, given current values of other variables. It can be shown that under reasonably general conditions, the distribution of the samples converges to the target distribution.

2) Metropolis-Hastings Algorithm [13]: The M-H algorithm generates the Markov chain using a candidate-generating density $q(x'|x)$ and an acceptance probability $\alpha(x'|x)$. Assuming the target distribution is $f(x)$, when the Markov chain is at a point x^t, a value x' will be generated from $q(x'|x^t)$, and this value is accepted as the next state x^{t+1} with probability $\alpha(x'|x^t) = \dfrac{f(x')q(x^t|x')}{f(x^t)q(x'|x^t)}$. This algorithm requires only the candidate-generating density $q(x'|x)$ and the ratio $f(x')/f(x)$, thus it is popular in Bayesian applications where the normalization factor is difficult to compute.

In our case, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ and $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\}$ are the variables that should be sampled using the M-H algorithm since their posterior distributions contain normalization factors which are difficult to compute. Other variables have their full conditional distributions known so they can be sampled using Gibbs sampling. To implement the M-H algorithm, it is necessary to specify a suitable candidate-generating density. We use the suggestion from [14]: if $f(x)$ can be written as $f(x) \propto \phi(x)h(x)$, where $h(x)$ is a density that can be easily sampled and $\phi(x)$ is uniformly bounded, we can set $q(x'|x) = h(x')$. Since $\boldsymbol{x}_i|(\mu_g, \sigma_g, V) \propto \mathcal{N}_d(\boldsymbol{x}_i; \mu_g, \sigma_g^2 I_d)\prod_j \Pr(V_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$, we let the candidate-generating density for $\boldsymbol{x}_i$ to be $\mathcal{N}_d(\boldsymbol{x}'_i; \mu_g, (\sigma_g)^2 I_d)$, then the acceptance probability is $\prod_j \Pr(V_{ij}|\boldsymbol{x}'_i, \boldsymbol{y}_j, \boldsymbol{\theta}) / \prod_j \Pr(V_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$. The candidate-generating density and acceptance probability for $\boldsymbol{y}_j$ can be obtained in a similar way.

The algorithm in this E-step is as follows:

- Use Metropolis-Hastings algorithm to sample $\boldsymbol{x}_i^{t+1}$:
  - Sample $\boldsymbol{x}'_i$ from the conditional distribution
    $$\boldsymbol{x}'_i|(K^t_{x_i} = g, \mu^t_g, \sigma^t_g) \sim \mathcal{N}_d(\mu^t_g, (\sigma^t_g)^2 I_d)$$
  - Set $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}'_i$ with probability equal to
    $$\frac{\prod_j \Pr(V_{ij}|\boldsymbol{x}'_i, y^t_j, \boldsymbol{\theta}^t)}{\prod_j \Pr(V_{ij}|\boldsymbol{x}^t_i, \boldsymbol{y}^t_j, \boldsymbol{\theta}^t)}.$$
    Otherwise set $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}^t_i$.
- Use Metropolis-Hastings algorithm to sample $\boldsymbol{y}_j^{t+1}$ in a similar way.
- Update other parameters according to their corresponding posterior distributions using Gibbs sampling.

B. Maximization Step

In this step, we need to maximize the expectation of $\log(\Pr(V, X, Y|\boldsymbol{\theta}))$ with respect to $\boldsymbol{\theta}$. Since $\Pr(V, X, Y|\boldsymbol{\theta}) = \Pr(V|X, Y, \boldsymbol{\theta}) \Pr(X|Y, \boldsymbol{\theta}) \Pr(Y|\boldsymbol{\theta})$, where $\Pr(X|Y, \boldsymbol{\theta}) = \Pr(X)$, and $\Pr(Y|\boldsymbol{\theta}) = \Pr(Y)$, maximizing the expectation of $\log(\Pr(V, X, Y|\boldsymbol{\theta}))$ with respect to $\boldsymbol{\theta}$ is equal to maximizing the expectation of $\log(\Pr(V|X, Y, \boldsymbol{\theta}))$ with respect to $\boldsymbol{\theta}$. The expectation of $\log(\Pr(V|X, Y, \boldsymbol{\theta}))$ is obtained via the MCMC methods in the E-step. It can be shown that a unique maximizer of $\boldsymbol{\theta}$ exists for this expectation; details are omitted due to space limitations.

C. Algorithm Complexity

Next, we briefly analyze the complexity of this algorithm. In each iteration of the E-step, the Metropolis-Hastings algorithm has complexity $O(t)$ to sample a data point x_i and complexity

$O(s)$ to sample y_j, where t and s are the number of x_i's and y_j's, respectively. Assuming the length of the Markov chain is l, then it takes $O(lst)$ steps to obtain all the data points $\{x_1, \ldots, x_t\}$ and $\{y_1, \ldots, y_s\}$. The last step in the E-step to update other parameters using Gibbs sampling has complexity $O(d(s+t)l)$, where d is the number of components in the mixture Gaussian model. Since d is usually small compared to s or t, the complexity of the E-step is dominated by $O(lst)$. Following a similar analysis, it can be obtained that the complexity of the M-step is dominated by $O(st)$. Assuming the EM algorithms takes r iterations to converge, then the overall computational complexity of the MCMC-based EM algorithm is dominated by $O(lstr)$.

The value of r is usually small. Based on our experiments, $r = 15$ is enough for the EM algorithm to converge. However, the typical value of l is usually in the range of $10^4 \sim 10^6$. For large data sets with very large values of s and t, the proposed EM algorithm has high computational requirements. A straightforward approach to reduce the computational complexity of the algorithm is to use multiple Markov chains in the Metropolis-Hastings algorithm and distribute each chain to a separate machine for processing, so that the value of l can be reduced. But due to the burn-in process of the Markov chain, the smallest value of l is still on the order of 10^3. To further reduce computational complexity, we can distribute the whole input data over multiple processors. However, it is nontrivial to combine local processing on each processor to achieve a useful global solution. We will consider the distributed implementation of our algorithm in the future work.

V. Model Extensions

In this section, we discuss two extensions of our model. The first is to combine the content and contextual information into the model. The content information may include the attributes of users and items, the date associate with the rating, etc. The contextual information can refer to information such as under what condition the ratings were given, e.g., in the case of movie recommendation, when and with whom the user watched the movie. It's known that using content and contextual information can improve prediction accuracy for collaborative filtering [15]. We show below how to combine these information into our model. We use $\boldsymbol{c}_i^x$ to represent the content information of user X_i, $\boldsymbol{c}_j^y$ for the content information of Y_j, and $\boldsymbol{s}_{ij}$ for the contextual information when V_{ij} was given. Then the parameter η_{ij} in the model introduced in Section III is modified to be $\theta_l - \boldsymbol{x}_i \cdot \boldsymbol{y}_j - \boldsymbol{\alpha c}_i^x - \boldsymbol{\beta c}_j^y - \boldsymbol{\gamma s}_{ij}$, where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are the new parameters. This new model naturally takes into account the effect of the content and contextual information.

The second extension of the model considers the robustness of the collaborative filtering algorithm. It is known that in the collaborative filtering system a malicious user may insert multiple well-designed profiles under false identities to bias the recommendations for economic advantages or lower the accuracy of the system to make it useless. We introduce the concept of 'trust' [9] into the model to increase the robustness of our algorithm. By trust we refer to the reliability of a user to deliver accurate predictions.

1) Computational Model of Trust: When a user is involved in the collaborative filtering process, he/she is usually participating with a number of other users and it may not be possible to judge whether the user positively or negatively affects the final prediction. We use a 'leave-one-out' protocol to measure the relative contribution of a user on the prediction accuracy. Consider that user X_i is involved in predicting user X_j's rating for item Y_k. Assuming the real rating is V_{jk}. The estimated rating is $\hat{V}_{jk}$ if using the information from X_i, and $\tilde{V}_{jk}$ if excluding the information from X_i. Let $\hat{e}_{jk}^{(i)} = |\hat{V}_{jk} - V_{jk}|$ and $\tilde{e}_{jk}^{(i)} = |\tilde{V}_{jk} - V_{jk}|$, then we define the contribution score of user X_i for predicting V_{jk} as

$$s_{jk}^{(i)} = \begin{cases} 0, & \text{if } \tilde{e}_{jk}^{(i)} - \hat{e}_{jk}^{(i)} \le 0, \\ 1, & \text{if } \tilde{e}_{jk}^{(i)} - \hat{e}_{jk}^{(i)} > 0. \end{cases}$$

In other words, we consider the user providing a good recommendation if its information will push the estimated rating toward the true value, otherwise we consider the user providing a bad recommendation. The overall trust value of user X_i is then defined as $t_i = (\sum_{j,k} s_{jk}^{(i)})/N_i$, where N_i is the number of the prediction processes that X_i is involved.

2) Trust-Based Collaborative Filtering: A natural approach to incorporate trust into our collaborative filtering algorithm is to weight the individual likelihoods using the users' trust values. In the initial model, the likelihood of the model is the product of the individual likelihoods of the users who have been involved, i.e., $\Pr(V|X,Y,\boldsymbol{\theta}) = \prod_{ij} \Pr(V_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$. We define the trust-based likelihood to be $\Pr^T(V|X,Y,\boldsymbol{\theta}) = \prod_{ij} \Pr(V_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})^{t_i}$ with t_i being the trust value of user X_i . Then the log-likelihood can be written as $\log \Pr^T(V|X,Y,\boldsymbol{\theta}) = \sum_{i,j} t_i \log \Pr(V_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j, \boldsymbol{\theta})$, which is a weighted version of the log-likelihood in the initial model. Moreover, we can use the trust value to filter out the users who are untrustworthy and use only the information from the trustworthy users for prediction. Define a user set $\mathcal{X}_f = \{X_i : X_i \in \mathcal{X}_r \text{ and } t_i > t_0\}$, where t_0 is a predefined threshold value, then the likelihood of the model is computed only over the user set $\mathcal{X}_f \cup \{X_1\}$ and the item set $\mathcal{Y}_r$.

By employing the trust mechanism, we can adjust the relative contributions from different users, therefore the system can automatically detect malicious users as they consistently provide bad recommendations and make their contribution to future recommendations ineffective.

VI. Evaluation

The data set we used in the experiments is a subset from the Netflix data sets [16], which includes about $4,000$ ratings for 100 movies by 100 users. The rating scale in the data set is discrete, taking values from 1 to 5, with 5 representing the user likes the movie most. To evaluate the performance of our algorithm, we use the mean absolute error (MAE) $e_{MAE} = (\sum_{j=1}^{n} |\hat{V}_{ij} - V_{ij}|)/n$. The prediction experiment is carried out based on the Given N protocol [1] to evaluate the obtained prediction accuracy. To be more specific, we randomly select N ratings from user X_1 as the observed ratings and attempt to predict his remaining ratings. This protocol examines the performance of the algorithm when there are different amount of user X_1's information available.

Two baseline collaborative filtering algorithms [3] are implemented in our experiments for the purpose of comparison. In the first baseline method, the median of ratings over all other users for the item is taken as the predicted rating. The second baseline method is the well-known k-nearest neighbor method, in which the similarity between the user X_1 and X_i are computed using Pearson's correlation coefficient:

$$\mathrm{sim}_{X_1,X_i} = \frac{\sum_{j\in\mathcal{J}}(V_{ij}-\bar{V}_i)(V_{1j}-\bar{V}_1)}{\sqrt{\sum_{j\in J}(V_{ij}-\bar{V}_i)^2}\cdot\sqrt{\sum_{j\in\mathcal{J}}(V_{1j}-\bar{V}_1)^2}},$$

where $\mathcal{J}$ is the set of indexes for the items that X_1 and X_i have both rated and $\bar{V}_i$ is the average rating of user X_i over all items with indexes in set $\mathcal{J}$. After similarities are calculated, the k most similar users are selected as the neighborhood $\mathcal{M}$ of X_1. The predicted rating of user X_1 for the item Y_k is computed as

$$\hat{V}_{1k} = \bar{V}_1 + \frac{\sum_{X_i\in\mathcal{M}}(\mathrm{sim}_{X_1,X_i}\cdot(V_{ik}-\bar{V}_i))}{\sum_{X_i\in\mathcal{M}}|\mathrm{sim}_{X_1,X_i}|}.$$

A. Experimental Results

1) Performance of the Algorithm: We tried 3 different values of N, i.e., $N \in \{10, 20, 40\}$, in the Given N protocol to evaluate the performance of our algorithm introduced in Section III. In other words, the numbers of known ratings for user X_1 in the experiments are 10, 20 and 40 respectively. The characteristic space is set to be 2 dimensional and the mixture Gaussian models for $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ are assumed to have 2 components. Optimizing these two parameters should give better performance, which is part of our future work.

Figure 2 illustrates the performance of our algorithms. In what follows, model I refers to the model with characteristic vectors restricted to unit length while model II is the model with user normalization. The line graph represents the relative error reduction achieved by model II, compared to the second baseline method. We can see that the algorithm with user normalization performs the best. We also observe that all algorithms perform better given more known ratings of X_1.

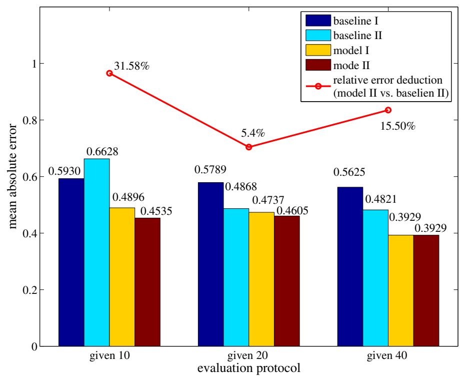

Fig. 2: Mean Absolute Error

2) Robustness of the Algorithm: In this section, we conduct experiments to evaluate the robustness of our extended model with the trust scheme. We test our algorithm under two basic attack models – the *random attack* and the *average attack*, which are introduced by Lam and Riedl [17]. These two attacks belong to the profile inject attack, in which the attacker injects maliciously designed user profiles into the system, in order to increase or decrease the system's predicated ratings on some target items for a given user or a group of users. We assume that the user profile created by the attacker are divided into three subsets P_t, P_v and P_ϕ. P_t is the set that contains the target items and their ratings. P_v contains randomly selected items that are not target items and their ratings. P_ϕ is the set containing the remaining items and these items are set as unrated . In the random attack, for each pair (Y_j, V_{ij}) in P_t, the attacker sets V_{ij} to be V_{max}, where V_{max} is the maximum possible rating value in the system. For each pair (Y_j, V_{ij}) in P_v, the attacker sets V_{ij} to be a value randomly drawn from a normal distribution $\mathcal{N}(\bar{V}, \bar{s})$, where $\bar{V}$ and $\bar{s}$ are the mean and standard deviation of ratings for all items. In the average attack, P_t is the same as in the random attack. But for each pair (Y_j, V_{ij}) in P_v, V_{ij} is set to a value randomly drawn from a normal distribution $\mathcal{N}(\bar{V}_j, \bar{s}_j)$, where $\bar{V}_j$ and $\bar{s}_j$ are the mean and standard deviation of ratings for item Y_j. Therefore, the average attack needs more information than the random attack, but it can increase the ratings of the target items more [18].

Before predicting the unknown ratings for user X_1 using the Given N protocol, we need to build the trust values for each user through a training phase. The settings for different attacks are classified by two parameters: the attack size, which is defined as the ratio of the number of malicious user profiles to the number of total user profiles, and the relative size of set P_t, which is defined as $f_1 = |P_t|/(|P_v|+|P_t|+|P_\phi|)$, where $|\cdot|$ represents the cardinality of a set. Figure 3 illustrates the mean trust values for the two types of attackers while the attack size is 2% and f_1 is are varying.

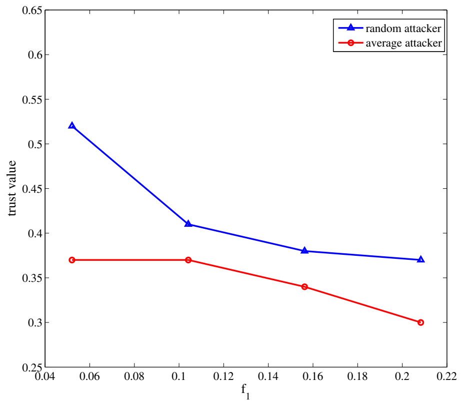

Fig. 3: Attacker's Trust Values for Different f_1

We can see that the average attacker usually has a lower trust value than the random attacker, which is consistent with the conclusion in [18]. Also, with the increase of f_1, the trust value of the attacker has a trend to decrease. The more targets the attacker aims at, the lower its trust values and the easier it to be detected. Further investigation on the experiments show that given one quarter of the users are malicious, the algorithm is still able to distinguish the attackers from normal users.

Figure 4 presents the histograms of the trust values for the normal users and the average attackers using our model. The result is obtained when the attack size is 3% and $f_1 = 0.3$. The height of the bar is the sum of the two distributions and the distribution for the average attacker is stacked on top of the distribution for normal users. We can see that the attacker consistently gets lower trust values than normal users. On the other hand, the two distributions are overlapped, which means that we may also exclude or put low weights on some normal users. However, it is reasonable to let a normal user have a low trust value when its information is not so helpful.

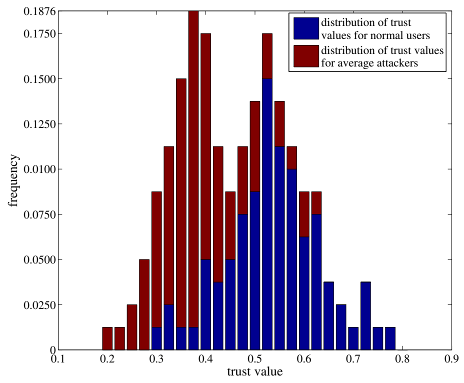

Fig. 4: Trust Values for Normal Users and Attackers

Figure 5 demonstrates the performance of our extended model with the trust schemes, i.e., model III and model IV. In model III, the ratings are weighted by the users' trust values, while in model IV, the ratings from users with low trust values are filtered out. The attacker setting is the same as that of Figure 4. As we expect, model IV performs the best.

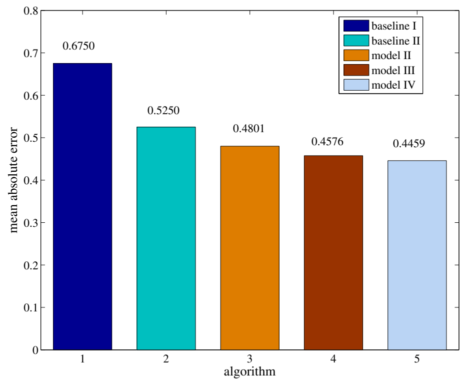

Fig. 5: Mean Absolute Error

VII. Conclusion

In this paper, we presented a probabilistic collaborative filtering algorithm based on ordered logistic regression. It takes into account similarities among the users and similarities among the items by introducing hidden characteristic vectors. The model can be easily extended to include the content and contextual information. We also propose a trust scheme to incorporate the trustworthiness of users into the model. Experiments on a real world data set shows promising performance of our algorithms.

Acknowledgement

Research supported by the Defense Advanced Research Projects Agency (DARPA) under award number 013641-001 for the Multi-Scale Systems Center (MuSyC), through the FRCP of SRC and DARPA, US Air Force Office of Scientific Research MURI award FA9550-09-1-0538, and Army Research Office MURI award W911-NF-0710287.

References

[1] J. Breese, D. Heckerman, and C. Kardie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 43–52.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, Chapel Hill, 1994, pp. 175–186.

[3] T. Hofmann, "Collaborative filtering via Gaussian probabilistic latent semantic analysis," in *Proceedings of ACM SIGIR'03*, Toronto, Canada, 2003.

[4] G. Shani, R. Brafman, and D. Heckerman, "An MDP-based recommender system," in *Proceddings of the 18th International Conference on Uncertainty in Artificial Intelligence*, Aug. 2002.

[5] D. Pavlov and D. Pennock, "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains," in *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS'02)*, 2002.

[6] H. Marlin, "Modeling user rating profiles for collaborative filtering," in *Proceedings of the 17th Annnual Conference Neural Information Processing Systems (NIPS'03)*, 2003.

[7] J. Golbeck, "Generating predictive movie recommendations from trust in social networks," in *Proceedings of the 4th International Conference on Trust Management*, 2006, pp. 93–104.

[8] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of ACM RecSys07*, 2007.

[9] J. O'Donovan and B. Smyth, "Trust in recommender systems," in *Proceedings of the 10th International Conference on Intelligent User Interfaces*, San Diego, CA, 2005, pp. 167–174.

[10] A. K. P. Brusilovsky and W. Nejdl, *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer Berlin, 2007.

[11] P. Hoff, A. Raftery, and M. Hancock, "Latent spcace approaches to social network analysis," *Journal of the American Statistical Association*, p. 1090, Dec. 2002.

[12] P. McCullagh and J. Nelder, *Generalized Linear Models*. Chapman and Hall, 1983.

[13] M. Jordan, *Learning in Graphical Models*. Kluwer Academic Pulishers, 1998.

[14] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *Journal of the American Statistical Association*, vol. 49, no. 4, pp. 327–335, Nov. 1995.

[15] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, Jun. 2005.

[16] Netflix Data Sets. [Online]. Available: http://www.netflixprize.com

[17] S. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International WWW Conference*, New York, NY, 2004.

[18] B. Mobasher, R. Burke, and R. Bhaumik, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology*, vol. 7, no. 4, Oct. 2007.