Dealing with Recommendations in a Statistical Trust Model

Jianqiang Shi[1], Gregor v. Bochmann[2], and Carlisle Adams[2]

[1]Systems Science

[2]School of Information Technology and Engineering (SITE)

University of Ottawa

Ottawa, Ontario, Canada K1N 6N5

{jianqshi, bochmann, cadams}@site.uottawa.ca

Abstract. We previously developed a trust model in which an entity makes decisions in online interactions by using past behavior as a predictor of likely future behavior. However, that approach does not combine recommendations from different entities. This paper focuses on the problem of recommendation combination and detection of unfair recommendations. Our approach involves data analysis methods (Bayesian estimation, Dirichlet distribution), and machine learning methods (Weighted Majority Algorithm). We describe simulation experiments to illustrate the effectiveness and robustness of the methods and the resulting evolution of trust[*].

1 Introduction

Trust models have emerged as an important risk management mechanism in online environments. The basic idea is to let entities rate each other's performance during transactions. The aggregated ratings about a given entity are used to derive trust, which can assist an entity in deciding whether or not to transact with the given entity in the future. Furthermore, entities can exchange information regarding the aggregated ratings of a given entity via recommendations [8]. In such trust referral systems, it is then the summation of recommendations that plays an important role in decision making.

In trust referral systems, a fundamental problem is that the quality of recommendations is not guaranteed, in the sense that malicious entities could give unfair recommendations and/or the subjective evaluation criteria may be different. Because entities are distributed and autonomous, it is generally unreasonable to assume that there are universally accepted trustworthy authorities who can declare the trustworthiness of different entities [9]. Consequently, the requesting entities must rely on themselves for discerning the trustworthiness of recommenders and the quality of recommendations.

We developed a model of trust management based on Bayesian estimation and the Weighted Majority Algorithm (WMA). In WMA, the weight given to each recommender depends on the history and quality of the recommendations given by the recommender. The assumption is that recommenders with low weights are likely to give unfair recommendations and recommenders with high weights are likely to give fair recommendations. The weights are updated after each transaction. By assigning different weights, our proposal detects and protects against unfair recommendations. By using Bayesian estimation, our proposal integrates the subjective prior knowledge and the actual experience into the estimation of trust.

---

[*] This work was supported in part by a grant from the Ontario Research Network for Electronic Commerce (ORNEC).

The scheme that most closely resembles ours is the one by Yu and Singh [9] which is targeted at detecting and protecting against spurious ratings. Their approach involves an application of the WMA to the Dempster-Shafer belief function and aggregation. Agents in their approach can adaptively choose their neighbors through which agents can find witnesses (recommenders). The testimonies of these witnesses are collected and combined to improve prediction accuracy. Some simple models of deception are studied to show the prediction accuracy and the evolution of trust networks.

Whitby *et al*. [8] propose a statistical filtering technique for excluding unfair ratings in Bayesian Reputation Systems. The assumption in their approach is that unfair ratings can be recognized by their statistical properties only. They design an iterated filtering algorithm based on the Beta distribution for binary ratings systems to exclude the presumed unfair ratings. The simulation results demonstrated the filtering algorithm is most effective when a moderate (less than 30%) number of raters behave consistently unfairly.

Our proposal is different from the above two proposals. Unlike Yu and Singh's proposal [9], our proposal uses Bayesian estimation and integrates the subjective prior knowledge into the estimation of trust. Our proposal also extends univariate trust ratings to multivariate ratings [6]. Furthermore, we focus on recommendation combination, and deliberately ignore the problem of finding witnesses. With respect to Whitby *et al*.'s proposal [8], our proposal extends binary ratings systems to multinomial outcome systems using the Dirichlet distribution. Whitby *et al*. [8] propose to calculate the combined rating based on current recommendations only, while entities in our trust model maintain a set of weights correlated with past recommendations, and past experience. Our proposal also takes self-experiences into account to calculate the combined rating.

The rest of this paper is organized as follows. Section 2 summarizes the background information on Bayesian estimation, Dirichlet distribution, and the Weighted Majority Algorithm. Section 3 introduces our approach. Section 4 presents simulations and their results. Section 5 concludes the paper.

2 Background Information

2.1 Bayesian Estimation and Dirichlet Distribution

Bayesian statistics provide a conceptually simple process for updating uncertainty in the light of evidence. Initial beliefs about some unknown quantity are represented by a *prior* distribution. The *prior* distribution and information in the data are then combined to obtain the *posterior* distribution for the quantity of interest. The *posterior* distribution expresses the revised uncertainty in light of the data; in other words, an organized appraisal after consideration of previous experience [5]. The mathematical analysis leading to the expression for *posterior* distribution can be found in many text books and reports on probability theory, such as Heckerman [1], and we only present the results here.

In multinomial sampling, the observed variable X is discrete; having r ($r \geq 2$) possible states $x^1, \ldots, x^r$. The simple conjugate *prior* $p(\boldsymbol{\theta}|\xi)$ used with multinomial sampling is the Dirichlet distribution [1]:

$$p(\boldsymbol{\theta} \mid \xi) = Dir(\boldsymbol{\theta} \mid \alpha_1, \ldots, \alpha_r) \equiv \frac{\Gamma(\alpha)}{\prod_{k=1}^{r} \Gamma(\alpha_k)} \prod_{k=1}^{r} \theta_k^{\alpha_k - 1} \qquad (1)$$

where $\alpha = \sum\alpha_k$ and $\alpha_k>0$, $k=1,\ldots, r$ and $\Gamma(.)$ is the *Gamma* function[1]. The quantities $\alpha_1,\ldots, \alpha_r$ are often referred to as hyperparameters to distinguish them from the parameter $\boldsymbol{\theta}=\{\theta_1, \theta_2,\ldots, \theta_r\}$ (the parameter θ_1 can be given by $1 -\theta_2\ldots-\theta_r$), which correspond to physical probabilities of $x^1,\ldots, x^r$. The *sufficient statistics* for observation data set $\boldsymbol{D}=\{X_1=x_1,\ldots,X_n=x_n\}$ are $\boldsymbol{N}=\{n_1,\ldots,n_r\}$, where n_k is the number of times $X=x^k$ in $\boldsymbol{D}$ which consists of n independent identically distributed random outcomes. Here ξ is the background knowledge. The *posterior* distribution is

$$p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)=Dir(\boldsymbol{\theta}|\alpha_1+n_1,\ldots, \alpha_r+n_r). \quad (2)$$

Given this *posterior* distribution, the probability of x^k for the next observation is given by

$$p(X_{n+1}=x^k|\boldsymbol{D}, \xi) = E_{p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)}(\theta_k) = \int\theta_k\, Dir(\boldsymbol{\theta}|\alpha_1+n_1,\ldots, \alpha_r+n_r)d\boldsymbol{\theta} = (\alpha_k+n_k)/(\alpha+n). \quad (3)$$

We note that the explicit mention of the state of knowledge ξ is useful, because it reinforces the notion that probabilities are subjective. When set r=2, the results are suitable for binomial sampling.

2.2 Weighted Majority Algorithm

The Weighted Majority Algorithm (WMA) [4] deals with on-line prediction algorithms that learn according to the following protocol. Learning proceeds in a sequence of trials. In each trial, each algorithm of the pool makes a prediction and these predictions are fed to the *master algorithm*. The *master algorithm* then makes its prediction and receives a correct label, which it passes to the whole pool. In the update step, each algorithm's weight is multiplied by some factor that depends on its prediction in this trial.

If predictions are continuous then a variant WMC of WMA is used, which allows the predictions as well as the labels to be in [0, 1]. The term update-trial j refers to the jth trial in which an update step occurs. The *master algorithm* is applied to a pool of n algorithms, letting $x_i^{(j)}$ denote the prediction of the ith algorithm of the pool in update-trial j. Let $\lambda^{(j)}$ denote the prediction of the *master algorithm* in update-trial j, $\rho^{(j)}$ denote the label of update-trial j, and $w_1^{(j)},\ldots, w_n^{(j)}$ denote the weights at the beginning of update-trial j. All initial weights $w_i^{(1)},\ldots, w_n^{(1)}$ are positive. The prediction of the *master algorithm* is

$$\lambda^{(j)} = \sum_{i=1}^{n} x_i^{(j)}\times w_i^{(j)}/s^{(j)} \quad \text{where } s^{(j)}=\sum_{i=1}^{n} w_i^{(j)}. \quad (4)$$

The prediction $x_i^{(j)}$ is discounted by its relative weight $w_i^{(j)}/s^{(j)}$, which represents the "belief" of the *master algorithm* in the prediction.

In an update step of WMC each weight $w_i^{(j)}$ is multiplied by some factor F that depends on γ, $x_i^{(j)}$, and $\rho^{(j)}$: $w_i^{(j+1)}=F\times w_i^{(j)}$, where F can be any factor that satisfies

$$\gamma^{|x_i^{(j)}-\rho^{(j)}|} \le F \le 1-(1-\gamma)\,|\,x_i^{(j)} - \rho^{(j)}\,|\,. \quad (5)$$

The parameter γ is the factor by which weights are multiplied and is always in the range [0,1). The parameter γ measures how drastic the update is (the smaller γ, the more drastic the update). For simplicity, we keep γ constant for all trials [4].

---

[1] $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. If x is an integer n=1,2,3,…, then $\Gamma(n)=(n-1)!$.

In WMC, the absolute loss is used as a quantity that measures how far the prediction is from the correct label. If an algorithm predicts x in a trial with label ρ, then its loss in that trial is $|x-\rho|$; this definition applies both to algorithms in the pool and to the *master algorithm*.

WMA aims to design a *master algorithm* that uses the predictions of the pool to make its own prediction. Ideally the *master algorithm* should make not many more mistakes than the best algorithm of the pool, even though it does not have any *a priori* knowledge as to which of the algorithms of the pool make few mistakes for a given sequence of trials.

2.3 Similarity and Distance

For the binomial case, the prediction and label are scalars. The absolute loss in WMC is $|x-\rho|$. For the multinomial case, the prediction and label are multi-dimensional vectors. Intuitively, we use normalized Euclidean distance in multi-dimensional space to represent the absolute loss. Suppose prediction $\boldsymbol{x}$ and label $\boldsymbol{\rho}$ are r-dimensional vectors. The normalized Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{\rho}$ is

$$\text{normalized Euclidean distance} = \frac{|\mathbf{x}-\boldsymbol{\rho}|}{\sqrt{2}} = \frac{\sqrt{\sum_{i=1}^{r}(x_i-\rho_i)^2}}{\sqrt{2}}. \quad (6)$$

Therefore the absolute loss is in the range [0, 1], whatever the dimension of predictions and labels is.

3 Our Trust Model Integrated with Bayesian Estimation and WMA

In this section, we explain how to apply Bayesian estimation and WMA to our trust model. Our approach involves three major steps: 1) setting initial situational trust and *prior* hyperparameters; 2) building trust from self experience; 3) dealing with recommendations.

3.1 Setting Initial Situational Trust and *Prior* Hyperparameters

The basic idea in this section is to let the conjugate *prior* match the initial situational trust, which can be set from basic trust. In our previous paper [6], we model trust $T_\alpha(\beta)(x)$ as the distribution $D_\beta(x)$ over the space of possible outcomes X. We also propose a mapping function $S(x)$ to set initial situational trust $T_\alpha(\beta,\delta)(S(x))$ from basic trust $T_\alpha(x)$ with initial observation number N_{init}. For simplicity without loss of generality, we suppose there are r ($r\geq 2$) different outcomes $x^1,\ldots, x^r$ in this situation δ. In the Bayesian technique, the likelihood function is given by

$$p(X=x^k|\boldsymbol{\theta}, \xi) = \theta_k, \quad k=1,\ldots,r \quad (7)$$

where $\boldsymbol{\theta}=\{\theta_1, \theta_2,\ldots, \theta_r\}$ are the parameters (the parameter θ_1 can be given by $1-\theta_2\ldots-\theta_r$) which correspond to physical probabilities of $x^1,\ldots, x^r$. The simple conjugate *prior* used with multinomial sampling is the Dirichlet distribution in Equation 1, where $\alpha = \sum\alpha_k = N_{init}$ and $\alpha_k>0$, $k=1,\ldots,r$. The hyperparameters $\alpha_{1,\ldots,}\ \alpha_r$ can be assessed as follows.

$$\text{Initial situational trust } T_\alpha(\beta,\delta)(x^k) = p(X=x^k|\boldsymbol{\theta}, \xi) = \theta_k = \alpha_k/\alpha \quad k=1,\ldots,r. \quad (8)$$

Given these r equations, we can solve for $\alpha_{1,\ldots,}\ \alpha_r$. The size of N_{init} determines the strength of the prior beliefs. If this N_{init} is small, such as 2, just a few observations will be enough to take over prior

beliefs. On the other hand, if the N_{init} is large, such as 1000, then on the order of 1000 observations will be needed to significantly make the *posterior* distribution differ from the prior beliefs [3]. When N_{init} approaches 0, the *prior* distribution becomes noninformative (theoretically N_{init} can be any positive real number).

3.2 Building Trust from Self Experience

The basic idea in this section is to predict the distribution of the next observation based on self experience using Bayesian estimation. Suppose the requesting entity has experience denoted by an observation data set $\boldsymbol{D}=\{X_1=x_1,\ldots,X_n=x_n\}$ for which the *sufficient statistics* are $\boldsymbol{N}=\{n_1,\ldots,n_r\}$, where n_k is the number of times $X=x^k$ in $\boldsymbol{D}$. Given conjugate *prior* $p(\boldsymbol{\theta}|\xi)$ and data set $\boldsymbol{D}$, the *posterior* distribution $p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)$ and the probability distribution for the next observation $p(X_{n+1}=x^k|\boldsymbol{D}, \xi)$ are given by

$$p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)=Dir(\boldsymbol{\theta}|\alpha_1+n_1,\ldots, \alpha_r+n_r) \tag{9}$$

$$T_\alpha(\beta,\delta)(x^k) = p(X_{n+1}=x^k|\boldsymbol{D}, \xi) = E_{p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)}(\theta_k)=\int\theta_k\, Dir(\boldsymbol{\theta}|\alpha_1+n_1,\ldots, \alpha_r+n_r)d\boldsymbol{\theta} = (\alpha_k+n_k)/(\alpha+n) \tag{10}$$

For simplicity, we now use $E_{p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)}(\boldsymbol{\theta})$ to denote situational trust $T_\alpha(\beta,\delta)(x)$. Note that $E_{p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)}(\boldsymbol{\theta})$ is an r-dimensional vector. For each x^k, θ_k, we have $T_\alpha(\beta,\delta)(x^k) = E(\theta_k)$.

3.3 Dealing with Recommendations

When taking recommendations into account, the requesting entity uses the Weighted Majority Algorithm. The basic idea in this section is to discount each recommendation according to its relative weight, which will increase if the recommendation is fair and will decrease if the recommendation is unfair. Suppose there are f recommenders from which to query recommendations. Each recommender gives a recommendation $\boldsymbol{R}_i=\{\alpha_{i,1}+n_{i,1},\ldots, \alpha_{i,r}+n_{i,r}\}$ $(i=1,\ldots,f)$ which are hyperparameters of the *posterior* distribution $p(\boldsymbol{\theta}_i|\boldsymbol{D}_i,\xi_i)=Dir(\boldsymbol{\theta}_i|\alpha_{i,1}+n_{i,1},\ldots, \alpha_{i,r}+n_{i,r})$ based on the recommender's past experience. Note that it is impossible for the requesting entity to distinguish the subjective hyperparameters $\alpha_{i,1},\ldots,\alpha_{i,r}$ from the *sufficient statistics* $\boldsymbol{N}_i=\{n_{i,1},\ldots, n_{i,r}\}$. Since these recommenders are not 100% trustworthy, their recommendations are discounted according to their relative weights. We refer to these discounted recommendations as *equivalent samples*, which are treated as if they are the requesting entity's own experience. For the requesting entity, the *sufficient statistics* for the *equivalent sample* of recommendation $\boldsymbol{R}_i$ are $\boldsymbol{N}_i'= \boldsymbol{R}_i\times w_i/s$ where $s=\sum_{i=1}^{f} w_i$. The final *sufficient statistics* are $\boldsymbol{N}_m=\boldsymbol{N}+\sum\boldsymbol{N}_i'$ and the final *posterior* distribution of the requesting entity is:

$$p(\boldsymbol{\theta}_m|\boldsymbol{D}_m, \xi)=Dir(\boldsymbol{\theta}_m|\alpha_1+n_1+ \textstyle\sum(\alpha_{i,1}+n_{i,1})\times w_i/s,\ldots, \alpha_r+n_r+ \sum (\alpha_{i,r}+n_{i,r})\times w_i/s). \tag{11}$$

In the terminology of WMA, we say that $E(\boldsymbol{\theta}_i)$ is the recommender's algorithm (the expectation of $\boldsymbol{\theta}_i$ with respect to the distribution $p(\boldsymbol{\theta}_i|\boldsymbol{D}_i, \xi_i)$ in Recommendation $\boldsymbol{R}_i$), and $E(\boldsymbol{\theta}_m)$ is the *master algorithm* (the expectation of $\boldsymbol{\theta}_m$ with respect to the distribution $p(\boldsymbol{\theta}_m|\boldsymbol{D}_m, \xi)$).

In the update step of WMA, the requesting entity estimates the correct label $\boldsymbol{\rho}$ using $E(\boldsymbol{\theta})$ (the expectation of $\boldsymbol{\theta}$ with respect to the distribution $p(\boldsymbol{\theta}|\boldsymbol{D}, \xi)$ from the requesting entity's past experience). The requesting entity now can update each recommender's weight using the factor $F=1-(1-\gamma)|E(\boldsymbol{\theta}_i)-E(\boldsymbol{\theta})|/\sqrt{2}$, note that $E(\boldsymbol{\theta}_m)$, $E(\boldsymbol{\theta}_i)$ and $E(\boldsymbol{\theta})$ are r-dimensional vectors, and $|E(\boldsymbol{\theta}_i) - E(\boldsymbol{\theta})|$ is the Euclidean distance. The requesting entity uses $E(\boldsymbol{\theta})$ as the correct label $\boldsymbol{\rho}$, because there does not

exist a service that can provide a correct label $\boldsymbol{\rho}$. Therefore, for the requesting entity, the only trustworthy way to get this label is to predict $\boldsymbol{\rho}$ purely based on its own past experience.

3.4 Example

We take a car wash example to illustrate the previous three steps. To simplify our analysis, we assume that the outcome space is a 1-dimensional space, and the outcome (binomial) is either "good"(1) or "bad"(0). We can use a scalar quantity, the probability of outcome "good", $p(x=1)$, to represent trust. We assume the hyperparameters α_g, α_b of the prior distribution for every entity are 1 and 1, respectively, which implies that N_{init}=2 and the basic trust value is 1/2. This corresponds to two initial experiences with outcomes one "good" and one "bad". Car wash β is the service provider, and entity α is the service requestor.

Now entity α can calculate its initial situational trust as follows:

$$\begin{aligned} &\textit{Initial situational trust } T_\alpha(\beta,\text{"car wash"})(\text{"good"}) = p(x=1) = 1/2 \\ &\textit{Initial situational trust } T_\alpha(\beta,\text{"car wash"})(\text{"bad"}) = p(x=0) = 1/2 \end{aligned} \quad (12)$$

Suppose entity α has seven experiences (n=7) denoted by an observation data set $\boldsymbol{D}=\{X_1=x_1,\ldots,X_7=x_7\}$. The *sufficient statistics* for the data set $\boldsymbol{D}$ are $\boldsymbol{N}=\{2,5\}$, which means that two "good" outcomes and five "bad" outcomes have been actually observed. Now entity α can estimate its situational trust as follows:

$$\begin{aligned} T_\alpha(\beta,\text{"car wash"})(\text{"good"}) &= p(1) = (2+1)/(7+2) = 1/3 \\ T_\alpha(\beta,\text{"car wash"})(\text{"bad"}) &= p(0) = (5+1)/(7+2) = 2/3 \end{aligned} \quad (13)$$

Suppose entity α has two friends with weights w_1=0.2 and w_2=0.8. Friends give recommendations $\boldsymbol{R}_1=\{6,2\}$ ($p(1)$=0.75) and $\boldsymbol{R}_2=\{3,7\}$ ($p(1)$=0.3) respectively. Entity α can get the *equivalent samples* as follows:

$$\begin{aligned} \boldsymbol{N}_1' &= \{6,2\}*0.2/(0.2+0.8) = \{1.2,\ 0.4\} \\ \boldsymbol{N}_2' &= \{3,7\}*0.8/(0.2+0.8) = \{2.4,5.6\} \end{aligned} \quad (14)$$

With the *equivalent samples*, entity α can calculate the final *posterior* distribution as follows:

$$p(\boldsymbol{\theta}_m|\boldsymbol{D}_m, \xi) = Dir(\boldsymbol{\theta}_m|1+2+1.2+2.4, 1+5+0.4+5.6) = Dir(\boldsymbol{\theta}_m|6.6,12) \quad (15)$$

The situational trust can be estimated as follows:

$$\begin{aligned} T_\alpha(\beta,\text{"car wash"})(\text{"good"}) &= p(1) = 6.6/(6.6+12) = 0.355 \\ T_\alpha(\beta,\text{"car wash"})(\text{"bad"}) &= p(0) = 12/(6.6+12) = 0.645 \end{aligned} \quad (16)$$

Suppose after the transaction, the actual outcome is "bad", then entity α can update its *sufficient statistics* $\boldsymbol{N}=\{2,6\}$. Entity α estimates the correct label ρ=2/(2+6)=0.25. The updating factor F for friend 1 is F=1-0.5*|0.75-0.25|=0.75. The new weight w_1 is equal to 0.2*0.75 = 0.15. Similarly entity α can get the updating factor F for friend 2: F=1-0.5*|0.3-0.25|=0.975. The new weight w_2 is 0.8*0.975=0.78.

4 Simulations

4.1 Description of Simulation

We continue the previous car wash example. The purpose of the simulation is to illustrate the accuracy of the proposed recommendation system when the performance of the service provider may vary over

time. Our simulations also provide a comparison of our recommendation system with the Iterated Filtering Algorithm (IFA) proposed by Whitby *et al*. [8]. The simulation includes one car wash, which is the service provider, and 50 car owners, which are the service requestors. The simulation is divided into sessions, and each session is divided into transactions. After each session, the service provider adapts its performance *Perf*, which is the probability of outcome "good".

4.2 Forgetting Factor

Since the performance of the car wash changes over time, the basic assumption about a given outcome distribution for the actions of the entity, valid over all time, is not true any more. In this case, we must give different weights to the different experiences. It is desirable to give greater weight to more recent experiences. This can be achieved by introducing a forgetting factor τ where $0 \leq \tau \leq 1$. When a new experience yielding outcome x^k is observed, the sufficient statistics $\boldsymbol{N}$ will be updated as follows:

$$\begin{aligned} \boldsymbol{N} &:= \boldsymbol{N} \times \tau \\ n_k &:= n_k + 1 \end{aligned} \tag{17}$$

where the value of τ determines the weight of the past experience compared with the most recent experience. In our simulation, we set the forgetting factor $\tau = 0.7$.

4.3 Service Provider Behavior

Once the car wash has committed to a transaction, it will either provide a “good” service (with a probability equal to its performance) or a “bad” service (with a probability equal to 1 – performance). At the end of each session t, the performance for the next session (t+1) is chosen randomly as follows:

$$Perf_{t+1} = \begin{cases} Perf_t + delta & \text{with a probability equal to } 1/3 \\ Perf_t - delta & \text{with a probability equal to } 1/3 \\ Perf_t & \text{with a probability equal to } 1/3 \end{cases} \tag{18}$$

In the simulation, *delta* = 0.1. In addition, the performance $Perf_{t+1}$ is restricted to the range [0,1]. We note that this kind of service provider behavior is a Markovian random walk.

4.4 Service Requestor Behavior

Car owners are divided into three groups: (1) In the fair group, the car owners always give fair recommendations. (2) In the unfairly high group, each car owner selects one favorite car wash, and gives unfairly high recommendations with a probability equal to *Punfair*. (3) In the unfairly low group, each car owner selects one target car wash, and gives unfairly low recommendations with a probability equal to *Punfair*. In this paper, we only represent one car wash scenario. For multiple car wash scenarios, please refer to the thesis [7]. We note that there is no decision making process involved in one car wash scenario, while for multiple car wash scenarios, the decision making process is crucial [7].

4.5 Recommendation Model

We define three recommendation models in the simulation: (1) Fair recommendation, for each trust value x, the recommendation given is $y=x$. (2) Unfairly high recommendation, for each trust value x,

the recommendation is $y=x+a*(1-x)$. (3) Unfairly low recommendation, for each trust value x, the recommendation is $y=(1-a)*x$. Here the const a is an exaggeration coefficient ($0 \le a \le 1$), which represents the degree of unfairness. Variable x is the estimation of the performance of the car wash. In this paper, we only represent simulation results of fair and unfairly low recommendations. For the results of combination of the above three recommendation models, please refer to the thesis [7].

4.6 Simulation Results

The simulations are conducted to assess the effectiveness of our proposal in several scenarios. For the purpose of comparison, all scenarios are conducted in three different modes: WMA, IFA, and SIMPLE. In the WMA mode, the entities use our proposal to combine recommendations. In the IFA mode, the entities use IFA (Iterated Filtering Algorithm by Whitby *et al.* [8]) to filter out unfair recommendations. In the SIMPLE mode, which is the reference mode, the entities simply average all the recommendations.

All simulations are conducted over 2000 transactions (20 sessions of 100 transactions each). The simulation is based on a Java simulation package - *javaSimulation* [2]. Several initial parameters are defined as follows:

1. Each car owner has exactly 6 randomly selected recommenders.
2. For each car owner, we set the initial weights w_i=1 and γ=0.5 for WMA, and *quantile*=0.01 for IFA.
3. Initial performance of car wash 1 is 0.6.

The following figures show the average trust value and the average estimation error for car wash 1 in the first 2000 transactions. The number on the x-axis is the transaction number of the car wash. The number on the y-axis represents average trust value, performance, and average estimation error, which is defined as average trust value minus performance.

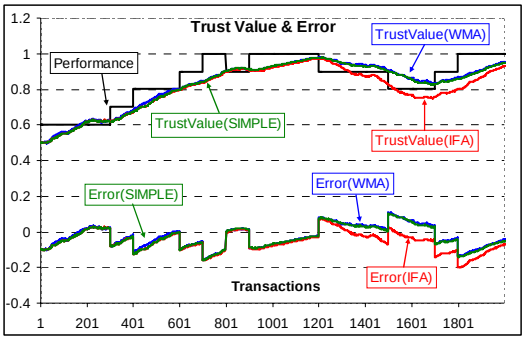

Fig. 1. Trust value, performance and error for car wash 1 with 0% unfair recommenders

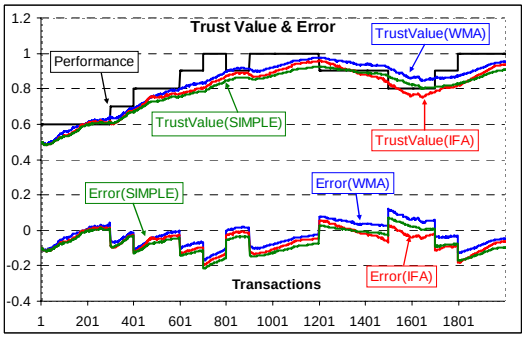

Fig. 2. Trust value, performance and error for car wash 1 with 20% unfair recommenders

The first simulation (Figure 1) is the basic scenario, in which there are no unfair recommenders. The second simulation (Figure 2) consists of 20% unfairly low recommenders with *Punfair*=100% and exaggeration coefficient a=0.875. The third simulation (Figure 3) consists of 40% unfairly low recommenders with *Punfair*=100% and exaggeration coefficient a =0.875.

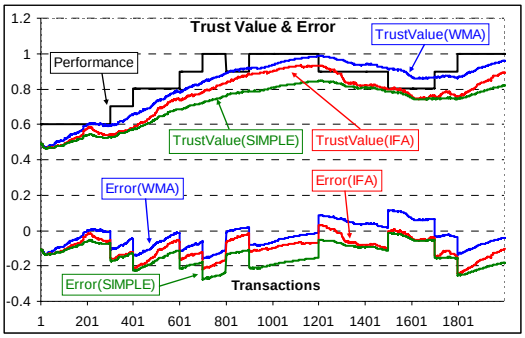

Fig. 3. Trust value, performance and error for car wash 1 with 40% unfair recommenders

Figure 1 shows that in the ideal scenario, the WMA and IFA algorithms are not helpful, since there are no unfair recommendations. Actually in transactions 1700-2000, the average estimation error in the IFA mode is worse than that in the SIMPLE mode, which implies that the IFA mistakenly filters out fair recommendations. Figure 2 and 3 show that both the WMA and the IFA can detect and avoid unfair recommendations to some degree. These figures directly illustrate how the trust value follows the changing performance in different modes.

4.7 Algorithm Effectiveness

The effectiveness of each algorithm is examined by simulating scenarios with three parameters:

1. The proportion of unfair recommenders
2. The probability that unfair recommenders give unfair recommendation (*Punfair*)
3. The degree of unfairness of unfair recommendations (exaggeration coefficient *a*)

For ease of comparison, we summarize the mean and standard deviation of average estimation error of WMA, IFA, and SIMPLE algorithms in different scenarios.

Table 1. Statistical information about average error (exaggeration coefficient *a*= 0.875, *Punfair*=1)

Unfairly low rate		0	0.2	0.4	0.6	0.8	1.0
mean	WMA	-0.0267	-0.0264	-0.0323	-0.0497	-0.1123	-0.3943
	IFA	-0.0546	-0.0610	-0.0987	-0.2624	-0.5854	-0.7203
	SIMPLE	-0.0310	-0.0661	-0.1412	-0.2394	-0.4475	-0.6415
std dev	WMA	0.0607	0.0629	0.0686	0.0734	0.0789	0.0772
	IFA	0.0558	0.0550	0.0592	0.0674	0.1509	0.1567
	SIMPLE	0.0611	0.0629	0.0639	0.0715	0.0950	0.1270

The following figure shows the mean and standard deviation of average estimation error of WMA, IFA, and SIMPLE algorithms during the first 2000 transactions in different scenarios described in Table 1. The number on the x-axis is the unfairly low rate (proportion of unfair recommenders). The number on the y-axis represents the mean and standard deviation of average estimation error. We note that the negative mean value of the average estimation error is due to the fact that the performance increases for most of the sessions.

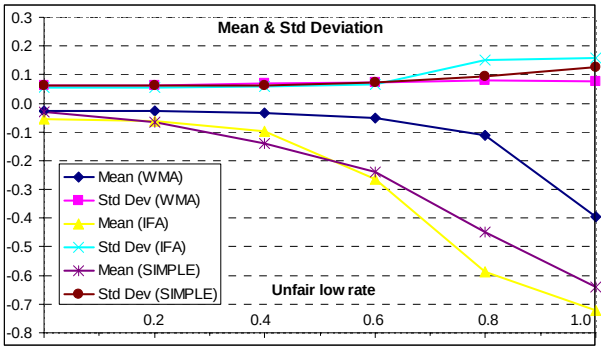

Fig. 4. Mean and standard deviation of average error of WMA, IFA and SIMPLE algorithm

It is clear in Figure 4 that the standard deviation of average estimation error of the three algorithms is not significantly different. However the standard deviation of the IFA algorithm increases faster than the others when unfairly low rate > 60%. Our proposal (WMA) has the least mean average estimation error.

Table 2 and Figure 5 show the effectiveness of the three algorithms in scenarios with exaggeration coefficient a= 0.875 and *Punfair*=0.25. Not surprisingly, all the algorithms increase their effectiveness compared with those in Table 1 and Figure 4. It is shown that IFA algorithm is the worst when unfairly low rate < 40%. WMA is the most effective algorithm in these scenarios.

Table 2. Statistical information about average error (exaggeration coefficient a= 0.875, *Punfair*=0.25)

Unfairly low rate		0	0.2	0.4	0.6	0.8	1.0
mean	WMA	-0.0267	-0.0256	-0.0302	-0.0430	-0.0630	-0.1187
	IFA	-0.0546	-0.0531	-0.0542	-0.0660	-0.1117	-0.1782
	SIMPLE	-0.0310	-0.0335	-0.0550	-0.0847	-0.1285	-0.1866
std dev	WMA	0.0607	0.0626	0.0672	0.0676	0.0668	0.0598
	IFA	0.0558	0.0562	0.0609	0.0605	0.0582	0.0601
	SIMPLE	0.0611	0.0635	0.0657	0.0655	0.0657	0.0623

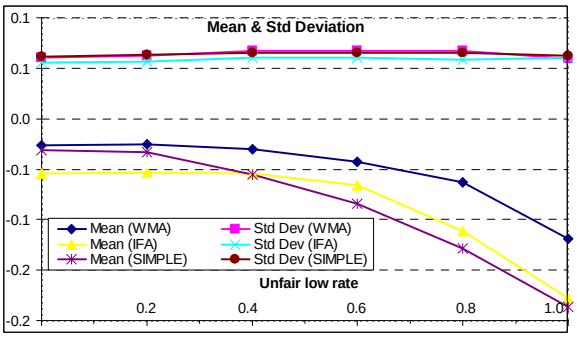

Fig. 5. Mean and standard deviation of average error of WMA, IFA and SIMPLE algorithm

The following two figures shows the effectiveness in scenarios with exaggeration coefficient a= 0.2, *Punfair*=0.25, and exaggeration coefficient a= 0.99, *Punfair*=1, respectively.

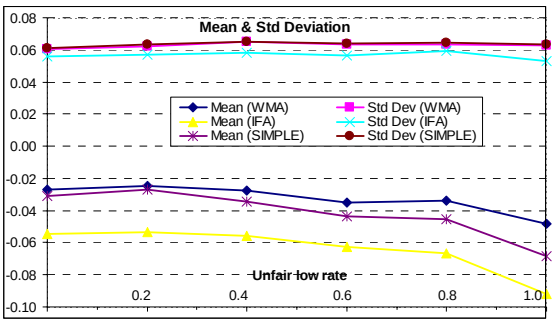

Fig. 6. Mean and standard deviation of average error (exaggeration coefficient *a*= 0.2 and *Punfair*=0.25)

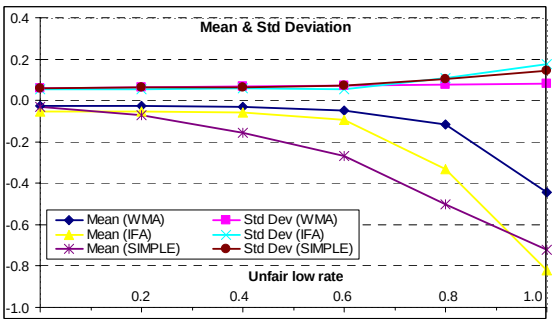

Fig. 7. Mean and standard deviation of average error (exaggeration coefficient *a*= 0.99 and *Punfair*=1)

Table 1 and Figure 4 show that the IFA algorithm is not effective compared with the SIMPLE algorithm in the scenario with exaggeration coefficient *a*= 0.875. In the scenario with low exaggeration coefficient and low *Punfair*, the IFA is particularly ineffective, even worse than the SIMPLE algorithm, which is illustrated by Figure 6. Figure 7 shows that the IFA algorithm is most effective in the scenario with very high exaggeration coefficient *a*= 0.99. To summarize, the IFA algorithm is very sensitive to the exaggeration coefficient. The higher the exaggeration coefficient, the more effective the IFA algorithm is. Regarding the average estimation error, our proposal (WMA) is better than IFA in all the above scenarios. More interestingly, our proposal is also much faster than IFA, since IFA consists of complex lower and upper *quantile* calculations. In our simulations, we also changed the parameter *quantile* of the IFA algorithm several times and found that the IFA algorithm produces the best results in most situations when the parameter *quantile* has the value 0.01.

5 Conclusion

This paper studies the problem of unfair recommendations. For simplicity, this work assumes that each entity has a fixed number of recommenders. Our approach integrates Bayesian estimation with a Weighted Majority Algorithm approach. We show that it is flexible and effective through simulations and comparisons with other algorithms. Our proposal shows great promise as a technique for improving the accuracy of trust estimation, and hence the fairness and robustness of trust models. In particular, our proposal is computationally efficient compared with the IFA algorithm.

Due to the intrinsic limitations of WMA, our trust model will become ineffective when either of the following conditions is satisfied: (1) The requesting entity has only one recommender, which makes the relative weight always equal to one, no matter what the recommendations are. (2) Among the requesting entity's recommenders, no one gives fair recommendations. These limitations can be solved by treating the requesting entity itself as a recommender. Therefore, the requesting entity at least has one recommender (itself) who always gives fair recommendations.

In our trust model, we studied 3 recommendation models. However, there may be another kind of unfair recommendation: flooded recommendation, in which the experience number is enlarged deliberately by the recommender in order to circumvent the Weighted Majority Algorithm. The flooded recommendation can be avoided by imposing an upper bound on experience number in recommendations. Any recommendation whose experience number exceeds the upper bound should be scaled to the upper bound.

6 References

1. David Heckerman: A Tutorial on Learning With Bayesian Networks. Technical Report, March 1995, http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-95-06
2. Keld Helsgaun: Discrete Event Simulation in Java. Technical Report, Roskilde University, Denmark http://www.akira.ruc.dk/~keld/research/ JAVASIMULATION/JAVASIMULATION-1.1/docs/Report.pdf
3. Richard Hughey, Anders Krogh: Hidden Markov models for sequence analysis: extension and analysis of the basic method. Computer Applications in the Biosciences (CABIOS), 12(2), 1996, pp. 95-107
4. Nick Littlestone, Manfred K. Warnuth: The Weighted Majority Algorithm. Information and Computation, 108(2), 1994, pp. 212-261
5. Stephane Pauquet: Bayesian Estimation. Course Notes, San Francisco State University, http://userwww.sfsu.edu/~efc/classes/biol710/bayes/a)-Bayesian-Estimation-web.html
6. Jianqiang Shi, Gregor v. Bochmann, Carlisle Adams: A Trust Model with Statistical Foundation. In proceedings of 2nd International workshop on Formal Aspects in Security and Trust, Toulouse, France, August 2004, pp.169-181
7. Jianqiang Shi: A Trust Model with Statistical Foundation. Master's Thesis, University of Ottawa, 2005
8. Andrew Whitby, Audun Jøsang, Jadwiga Indulska: Filtering Out Unfair Ratings in Bayesian Reputation Systems. In proceedings of the Workshop on Trust in Agent Societies, at the Autonomous Agents and Multi Agent Systems Conference (AAMAS2004), New York, July 2004
9. Bin Yu, Munindar P. Singh: Detecting Deception in Reputation Management. In proceedings of AAMAS2003, Melbourne, Australia, July 2003, pp. 73-80