Modeling Contextual Agreement in Preferences

Loc Do
School of Information Systems
Singapore Management University
hldo.2012@phdis.smu.edu.sg

Hady W. Lauw
School of Information Systems
Singapore Management University
hadywlauw@smu.edu.sg

ABSTRACT

Personalization, or customizing the experience of each individual user, is seen as a useful way to navigate the huge variety of choices on the Web today. A key tenet of personalization is the capacity to model user preferences. The paradigm has shifted from that of individual preferences, whereby we look at a user's past activities alone, to that of shared preferences, whereby we model the similarities in preferences between pairs of users (e.g., friends, people with similar interests). However, shared preferences are still too granular, because it assumes that a pair of users would share preferences across *all* items. We therefore postulate the need to pay attention to "context", which refers to the specific item on which the preferences between two users are to be estimated. In this paper, we propose a generative model for contextual agreement in preferences. For every triplet consisting of two users and an item, the model estimates both the prior probability of agreement between the two users, as well as the posterior probability of agreement with respect to the item at hand. The model parameters are estimated from ratings data. To extend the model to unseen ratings, we further propose several matrix factorization techniques focused on predicting agreement, rather than ratings. Experiments on real-life data show that our model yields context-specific similarity values that perform better on a prediction task than models relying on shared preferences.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Data Mining

Keywords

user preference; contextual agreement; generative model

1. INTRODUCTION

Users face a dizzying array of choices for almost any decision they make on the Web today, e.g., which movie to see, which book to purchase, which job to pursue next and when [40], which tag to use [26]. In exploring options, the limiting factor is often not affordability or availability, but rather the user's time and attention. Many Web platforms deal with this scarce resource through *personalization*, by focusing the user's attention on things most likely to be of interest.

In order to provide a personalized experience to each user, we first need to know the user's preferences. To some extent, this is provided by the user's own past activities, such as which Facebook posts she liked or disliked, or which products on Amazon she purchased, or which movie on Netflix she rated high or low. However, these preference signals are too *sparse*. They are expressed over a very limited number of items. For instance, most Netflix users would have assigned ratings to only tens of movies, as compared to the thousands of available movies. Hence, there is a need to extrapolate from these signals to build a more general preference model.

Most of the previous work in this area focus on modeling *individual preferences*. The aim is to derive user-specific models from preference data, e.g., ratings, which will help predict future adoptions or ratings by the user. There are several well-known classes of methods, such as aspect model [8, 9], matrix factorization [15], and content-based model [1, 31, 21] (see Section 2 for a more expansive review). These methods are still actively being researched, and their use is prevalent in industrial recommender systems.

Beyond individual preferences, there is also a significant body of work on the complementary issue of *shared preference* between pairs of users. For instance, neighborhood-based collaborative filtering systems [12, 33, 5] predict a user's rating on an item as the weighted average of ratings by her neighbors. Here, neighbors are other users with high similarity in preferences, and the weights are proportional to the degrees of similarity between a pair of users. Shared preference is useful, because some individuals may not have established a sufficiently long record of activities (e.g., ratings) for a reasonably accurate individual model to be built. However, the limited record may already be sufficient to infer her similarity to another user with a longer record or more accurate model, which can then be "borrowed" to help in the predictions for the former user. Alternatively, there may be extra information, e.g., social network, to induce preference sharing between friends [24, 22].

While shared preference is helpful, it also makes the implicit assumption that the similarity between two users applies equally to all items under consideration. More realistically, users have diverse preferences. Even a similar pair of users do not agree at all times. We therefore postulate the

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568006.

Movie	Rating by talyseon	Rating by youngchinq
Paranormal Activity	5	5
Payback	3	3
Coraline	5	5
Pan's Labyrinth	5	5
Memento	5	4
Gran Torino	5	4
The Hurt Locker	5	4
Jurassic Park III	3	2
Twilight	3	1
Inception	5	3
Daredevil	3	1
I Am Legend	4	2
Rosemary's Baby	5	2
The Day After Tomorrow	4	1
300	4	1
Moulin Rouge	5	2
Seven Pounds	4	1
The Dark Knight	5	1
The Last Samurai	5	1
Star Wars Episode III: Revenge of the Sith	5	1

Table 1: Epinions users *talyseon* and *youngchinq*

need to pay attention to "context", arguing that while a pair of users may agree in their preferences in one "context", they may disagree in a different "context". There are many ways to define "context". For instance, the product category or the time of day could each be a specific context. However, these definitions assume the presence of additional information in the data. To retain the most common framework in the literature, which is to rely on rating data alone, in this paper, by "context", we refer to each specific item. In other words, we are interested in the contextual agreeement of preferences between two users in the context of one item.

To illustrate this more clearly, we use a real-life example from Epinions.com, an online rating site for various products, e.g., movies (used in this example). In Table 1, we show the rating profiles of two users: *talyseon* and *youngchinq*, on twenty movies that both of them had rated. The ratings are from 1 (low) to 5 (high). The traditional approach of shared preference is to use these ratings to measure the overall similarity between the two users. Using Pearson's correlation, their similarity is 0.53. Using Cosine similarity, their similarity is 0.88. (See Section 2 for the definitions of these measures.) These similarities are considered high as Pearson ranges from -1 to 1, and Cosine ranges from 0 to 1.

On one hand, the two users do share some preferences. The top few movies in the list are movies that both assign high ratings to, which tend to be dramas and thrillers. On the other hand, a single similarity value cannot reveal the full picture of their preference sharing. The last few movies in the list are those that *talyseon* likes but *youngchinq* dislikes. These tend to be fantasy types (e.g., Star Wars III, Dark Knight). Therefore, agreement on preference should be seen in the context of individual items. For instance, we say that *talyseon* and *youngchinq* agree on their preference in the context of "Paranormal Activity" movie. However, they disagree in the context of "The Last Samurai" movie.

Problem. Given a set of users (e.g., u), a set of items (e.g., i), and some ratings by users on items (e.g., r_{ui}), we seek to model the contextual agreement between a pair of users u and v on a specific item i (collectively denoted as a triplet $\langle u, v, i \rangle$). Instead of just another similarity measure, we model this contextual agreement as a probability measure, with a binary random variable y_{uvi} with two outcomes: agreement ($y_{uvi} = 1$) or disagreement ($y_{uvi} = 0$).

One key observation is that the observed rating values, e.g., r_{ui} and r_{vi}, provide signals of the agreement or disagreement between u and v on item i. To represent this more succintly, we derive a quantity x_{uvi}, which is a function of r_{ui} and r_{vi}, i.e., $x_{uvi} = \mathcal{F}(r_{ui}, r_{vi})$, through some function $\mathcal{F}$ (to be defined later). Our problem can thus be restated as estimating the probability of agreement $\mathrm{P}(y_{uvi} = 1|x_{uvi})$.

This gives rise to two sub-problems. The first is the probabilistic modeling of $\mathrm{P}(y_{uvi} = 1|x_{uvi})$. This is akin to probabilistic clustering, whereby we seek to decide the latent "class" y_{uvi} using the "feature" x_{uvi}. Therefore, we adopt a generative modeling based on Gaussian mixtures, which has been applied to other unsupervised clustering problems. We call this model *Contextual Agreement Model* or *CAM*.

The second sub-problem is that not all x_{uvi}'s are observed, arising directly from not having observed all possible ratings. For "unseen" triplets, where either r_{ui} or r_{vi} is unobserved, we need to predict $\hat{x}_{uvi}$ (the hat indicates predicted, rather than observed). For this, we adopt the framework of matrix factorization. The key to our approach is the minimization of a novel objective function based on optimizing for agreement, rather than rating, prediction. We call this method *Differential Probabilistic Matrix Factorization* or *DPMF*.

Application. The probability of contextual agreement allows for a better estimation of the contextual similarity between a pair of users on a specific item. This will come in useful in several potential applications. First, as we will explore in Section 3, the agreement probability can be used in calibrating the similarities between neighbors in an item-specific manner for a neighborhood-based recommender system to derive a rating prediction. Second, it can support a more targeted social recommendation. When a user wants to recommend an item to her friends, instead of sharing with all friends, the contextual agreement probability can identify the subset of friends most in agreement on the item. Third, the model may be useful in a study of prevalence of agreement in different communities, product categories, etc.

Scope. While our work is related to recommender systems, our focus is on modeling preferences, and not on rating prediction. The reader may also surmise that a similar contextual agreement framework may apply to triplets involving a user and two items. This is indeed the case, but to maintain focus, we will discuss only triplets involving two users and an item. As input, we assume only ratings data, and not other information such as categories or ontologies [28]. We also assume that ratings are truthful and reflective of user preferences (and not artefacts of dishonesty or fraud [6]), which we believe is true for a vast majority of users.

Contributions. *First*, as far as we know, we are the first to propose modeling item-specific context in estimating the agreement between a pair of users on an item. *Second*, to realize this modeling, in Section 4, we develop a probabilistic generative model, called *CAM*, based on Gaussian mixtures. We enforce a monotonicity property that results in a specific parameter constraint, and describe how to learn the constrained parameters with Expectation Maximization. *Third*, to extend this model to unseen triplets, in Section 5, we outline how several matrix factorization methods can be applied. We also propose a new method, called

DPMF, with a novel objective function that minimizes errors in rating differences, and describe its gradient descent learning algorithm. *Fourth*, in Section 6, we validate these models comprehensively on three real-life, publicly available rating datasets, showing how well the model parameters are learned, and how they improve upon shared preference models in a neighborhood-based rating prediction task.

2. RELATED WORK

In the following, we survey related work on modeling preferences, first focusing on individual users, and then on similarities between users, and finally on the role of context.

Individual preference. Most works on modeling individual preference are found in model-based recommender systems [1]. The main step is to construct a preference model for each user, which is then used to derive predictions. Here, we review three popular modeling choices. The first is *aspect model* [8, 9]. A user u's preference is modeled as a probability distribution $\{\mathrm{P}(z_k|u)\}_{k=1}^{K}$ over K latent aspects. Each aspect z_k is associated with a distribution over items i to be adopted, i.e., $\mathrm{P}(i|z_k)$, or over ratings r, i.e., $\mathrm{P}(r|z_k, i)$.

The second is *matrix factorization-based model* [15]. User u's preference is modeled as a column vector S_u in a K-dimensional latent space. Each item i is also associated with a rank-K column vector Q_i. The rating prediction $\hat{r}_{ui}$ by u on i is given by ${S_u}^T Q_i$. There are different factorization methods [18, 39, 17, 25], which vary in their objective functions, including several probabilistic variants [29, 34].

The third is *content-based model* [1, 31, 21]. User u's preference is modeled as a content vector whose dimensionality is the vocabulary size (e.g., $tf \cdot idf$ vector), derived from the content (e.g., meta-data, text) of items that u likes.

Shared preference. Modeling sharing of preferences is mostly found in neighborhood-based recommender systems [11]. One approach is based on *similarity*. For user-based collaborative filtering (CF) [12], the similarity w_{uv} is between a pair of users u and v. The higher w_{uv} is, the more u and v share their preferences. The most common similarity measures in the literature are Pearson's correlation coefficient [33], and vector space or Cosine similarity [5]. Given that $\mathbf{r}_u$ and $\mathbf{r}_v$ represent vectors of ratings, $\{r_{ui}\}$ by u and $\{r_{vi}\}$ by v, on a set of items $\{i\}$, Pearson is determined as in Equation 1 (where $\bar{r}_u$ and $\bar{r}_v$ are average ratings), and Cosine as in Equation 2. Correspondingly, for item-based CF [35, 19], the similarity is between a pair of items.

$$w_{uv}^{pearson} = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}} \tag{1}$$

$$w_{uv}^{cosine} = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u|| \times ||\mathbf{r}_v||} \tag{2}$$

Another approach to model sharing of preferences is to exploit existing *structures*. For example, in a social network, each relationship (e.g., friends or follower-followee) is seen as inducing sharing of preferences between the two users [24, 22]. Some exploit the taxonomy structure to induce sharing between items in the same category [36, 2, 13, 27, 14].

Context. Most of the work discussed above base their approaches on the dyad of user-item pair. In some cases, additional information or "context" may be available, i.e., rather than pairs $\langle u, i \rangle$, we observe triplets $\langle u, i, c \rangle$ where c refers to some context. There are different approaches to dealing with triplets. One approach is to break a triplet into multiple binary relations, e.g., friend-user-item into user-friend and user-item such as done in [23, 38, 37] for rating-cum-link prediction. [41, 20] suggest partitioning dyads into clusters based on context, and then learning a separate model for each cluster. Another approach is tensor factorization, such as done in [10] for cross-domain rating prediction. Yet another approach, such as ours, is to model triplets directly. Differently from [32, 30, 16] targeting user-item-item triplets for personalized ranking of items (asymmetric), we target user-user-item triplets to model agreement.

3. OVERVIEW

Notations. The universal set of users is denoted as $\mathcal{U}$, and we use u or v to refer to a user in $\mathcal{U}$. In turn, we use i or j to refer to an item in the universal set of items $\mathcal{I}$. The rating by u on i is denoted as r_{ui}. The set of all ratings observed in the data is denoted R. We seek to model user-user-item triplets $\langle u, v, i \rangle$. The universal set of triplets comprises $\mathcal{U} \times \mathcal{U} \times \mathcal{I}$, excluding triplets involving the same users, e.g., $\langle u, u, i \rangle$. Each triplet $\langle u, v, i \rangle$ is associated with two quantities (modeled as random variables): x_{uvi} and y_{uvi}, which are essential to our probabilistic modeling.

The variable $x_{uvi} \in \mathbb{R}$ is real-valued. It represents the indicator of agreement between u and v on i, some of which are observed in the data. The closer x_{uvi} is to 0, the more likely it is that u and v agree on i. If $x_{uvi} \ll 0$ or $x_{uvi} \gg 0$, then disagreement is more likely. x_{uvi} can be expressed as a function of ratings, i.e., $x_{uvi} = \mathcal{F}(r_{ui}, r_{vi})$. While there are many possible definitions of $\mathcal{F}$, in this paper, we simply use the *rating difference* between two users on the same item, as shown in Equation 3. This choice of function also implies the symmetry of $x_{uvi} = -x_{vui}$.

$$x_{uvi} = r_{ui} - r_{vi} \tag{3}$$

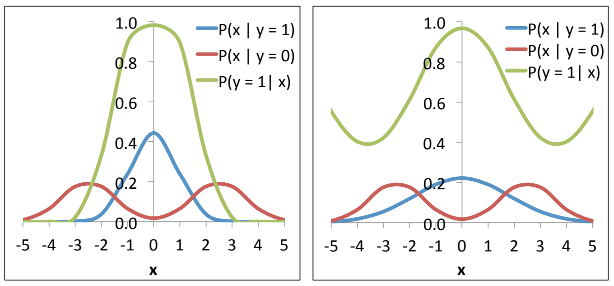
The second variable $y_{uvi} \in \mathcal{Y} = \{0, 1\}$ is binary. $y_{uvi} = 1$ represents the event of agreement between u and v on their preference for i. $y_{uvi} = 0$ is the event of disagreement. These events are latent, and never observed. They are to be estimated from the observed x_{uvi}'s. The closer x_{uvi} is to 0, the more likely we expect $y_{uvi} = 1$. The further x_{uvi} is away from 0, the more likely we expect $y_{uvi} = 0$.

Problem Formulation. Given ratings data R, and the above x_{uvi} definition, we seek to estimate the probability $\mathrm{P}(y_{uvi}|x_{uvi})$ for all triplets. Not all x_{uvi}'s can be observed. x_{uvi} is not observed if either $r_{ui} \notin R$ or $r_{vi} \notin R$. This gives rise to two sub-problems. The first is how to estimate $\mathrm{P}(y_{uvi}|x_{uvi})$ given the observed x_{uvi} values. The second sub-problem is how to predict the un-observed $\hat{x}_{uvi}$ values.

For the first sub-problem, we propose the probabilistic *CAM* model in Section 4. Since y_{uvi} is latent, it is not possible to employ discriminative modeling. We therefore turn to generative modeling, by representing x_{uvi} as a random variable, whose generative process is related to y_{uvi}. Our approach is thus to model the joint probability $\mathrm{P}(y_{uvi}, x_{uvi})$. The conditional probability $\mathrm{P}(y_{uvi}|x_{uvi})$ can afterwards be estimated from the joint probabilities as follows:

$$\mathrm{P}(y_{uvi}|x_{uvi}) = \frac{\mathrm{P}(y_{uvi}, x_{uvi})}{\sum_{y'_{uvi} \in \mathcal{Y}} \mathrm{P}(y'_{uvi}, x_{uvi})} \tag{4}$$

The second sub-problem is how to predict the unseen $\hat{x}_{uvi}$. We will then use the predicted $\hat{x}_{uvi}$ with the parameters



a. Well-behaved Parameters (θ = <0.7, 0, 0.9, 2.5, 1>)

b. Errant Parameters (θ = <0.7, 0, 1.8, 2.5, 1>)

Figure 1: Distributions of $\mathrm{P}(x|y)$ and $\mathrm{P}(y|x)$

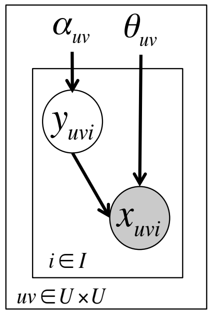


Figure 2: Plate Diagram for CAM

learned in the first sub-problem, to estimate $\mathrm{P}(y_{uvi}|\hat{x}_{uvi})$. Our key insight is that the $\hat{x}_{uvi}$'s are not independent from one another. All triplets involving the same item i or the same user pair (u,v) will share some dependency. Furthermore, the triplet should model the interaction of users and items. Our approach is to model the generation of x_{uvi} based on user- or item-specific parameters so as to generate/predict unseen $\hat{x}_{uvi}$ through matrix factorization in Section 5. The framework can accommodate different predictive methods. Indeed we outline several potential methods, including a new proposed method called *DPMF*.

Application. One application of the agreement probabilities is as a similarity value in a neighborhood-based collaborative filtering (CF). User-based CF [11] exploits the similarities between users to predict unseen ratings. Adopting the same rating prediction framework, we can use the contextual agreement to weigh the contributions of neighbors. To predict an unseen rating $\hat{r}_{ui}$, we use Equation 5, which is the weighted average of ratings on i by u's neighbors. Neighbor v can be any user, weighted by w_{uvi}.

$$\hat{r}_{ui} = \frac{\sum_{v \neq u, r_{vi} \neq \phi} w_{uvi} \times r_{vi}}{\sum_{v \neq u, r_{vi} \neq \phi} w_{uvi}} \tag{5}$$

In our case, we use $w_{uvi} = \mathrm{P}(y_{uvi} = 1|\hat{x}_{uvi})$, which is specific to every item i. In Section 6, we will compare this to the traditional case of shared preference, where the weight w_{uvi} is set to the similarity between u and v, which is then applied to all items. The most popular similarity functions are Pearson's coefficient [33] and Cosine similarity [5]. This comparison is fair as both approaches are given exactly the same set of ratings to use, but differ only in the relative weights of the ratings. Note that in this application, our objective is not to propose a new rating prediction algorithm, but rather to illustrate the utility of contextual agreement, and enable comparison to appropriate baselines.

4. CONTEXTUAL AGREEMENT MODEL

4.1 Generative Model

Given the observed x_{uvi}'s, we want to estimate the probability distribution of contextual agreement $\mathrm{P}(y_{uvi}|x_{uvi})$. When the context is clear, we simplify the notations for y_{uvi} and x_{uvi} to y and x respectively. Because y is latent, we estimate the conditional probability $\mathrm{P}(y|x)$ from the joint probability $\mathrm{P}(y,x)$. In a generative modeling framework, we decompose $\mathrm{P}(y,x)$ into $\mathrm{P}(x|y)\mathrm{P}(y)$. $\mathrm{P}(y)$ corresponds to the prior probability of agreement between u and v on i. $\mathrm{P}(x|y)$ is the likelihood that x has been generated from y.

The prior of agreement $\mathrm{P}(y)$ is the base level of agreement between u and v before seeing the item i. Given that there are two probable events, i.e., agreement $(y=1)$ and disagreement $(y=0)$, we model this as a Bernoulli process with a parameter α. In other words, the prior of agreement is $\mathrm{P}(y=1)=\alpha$, and of disagreement is $\mathrm{P}(y=0)=1-\alpha$.

In the event of agreement $(y=1)$, x will be generated according to a probability $\mathrm{P}(x|y=1)$. Because x is real-valued, and we expect that its values in the event of agreement will cluster together, we model the generation of x as a Gaussian, with an underlying mean μ_1 and variance σ_1^2. As mentioned in Section 3, the closer is x_{uvi} to 0, the more likely it is that u and v agree on i. Therefore, we make a simplifying step, and set $\mu_1 = 0$. We learn σ_1 from data. The blue curve in Figure 1(a) illustrates the probability density function (p.d.f.) of $\mathrm{P}(x|y=1)$, which is a Normal distribution centered at $\mu_1 = 0$ (in this example, $\sigma_1 = 0.9$).

In the event of disagreement $(y=0)$, x will be generated according to to a probability $\mathrm{P}(x|y=0)$. Since $x \gg 0$ or $x \ll 0$ indicates disagreement, the mean of this Gaussian should be away from 0. Due to the symmetric property $x_{uvi} = -x_{vui}$, we model this as a bimodal distribution, such as an equally-weighted mixture of two Gaussians with positive mean at μ_0 and negative mean $-\mu_0$, and a variance of σ_0^2. The red curve on Figure 1(a) illustrates the bimodal p.d.f. of $\mathrm{P}(x|y=0)$ (in this example, $\mu_0 = 2.5$, $\sigma_0 = 1$).

$\mathrm{P}(y|x)$ can therefore be expressed in terms of these components as shown in Equation 6. The green curve on Figure 1(a) illustrates the "decision function" or the p.d.f. of $\mathrm{P}(y=1|x)$, estimated from the respective prior $\mathrm{P}(y)$ and likelihood $\mathrm{P}(x|y)$. As expected, $\mathrm{P}(y=1|x)$ is highest when $x \approx 0$. As x moves away from 0, the probability of agreement decreases, which fits the modeling objective.

$$\mathrm{P}(y|x) = \frac{\mathrm{P}(x|y)\mathrm{P}(y)}{\sum_{y' \in \mathcal{Y}} \mathrm{P}(x|y')\mathrm{P}(y')} \tag{6}$$

Generative Process. We now describe the full generative process for a set of observed triplets $X = \{x\}$.

For every triplet $x \in X$:

1. Draw an outcome for $y \in \{0,1\}$:

$$y \sim Bernoulli(\alpha)$$

2. Draw an outcome for $x \in \mathbb{R}$:

(a) In the event of agreement, i.e., $y = 1$:

$$x \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

(b) Else, in the event of disagreement, i.e., $y = 0$:

$$x \sim \frac{1}{2}\mathcal{N}(\mu_0, \sigma_0^2) + \frac{1}{2}\mathcal{N}(-\mu_0, \sigma_0^2)$$

Based on this generative process, the distribution of x can be expressed as a mixture of three Gaussians with weights α, $\frac{1-\alpha}{2}$, and $\frac{1-\alpha}{2}$ respectively, as shown in Equation 7.

$$x \sim \alpha\mathcal{N}(\mu_1, \sigma_1^2) + \frac{1-\alpha}{2}\mathcal{N}(\mu_0, \sigma_0^2) + \frac{1-\alpha}{2}\mathcal{N}(-\mu_0, \sigma_0^2) \quad (7)$$

Parameters. For the above generative process, the set of parameters can be encapsulated by $\theta = \langle \alpha, \mu_1, \sigma_1, \mu_0, \sigma_0 \rangle$. The question arises whether there is a unique θ for every triplet $\langle u, v, i \rangle$. Because θ is a distributional parameter, it is not feasible to estimate θ from a single observation of x. Another approach is to tie together the parameters of a group of triplets. In this paper, we propose to tie the parameters of triplets corresponding to each pair of users. In other words, there is a specific θ_{uv} for each pair of users u and v that applies to all items. As shown in the plate diagram in Figure 2, α_{uv} and θ_{uv} are within the plate of each pair of users. For clarity, we draw α_{uv} separately to show that y_{uvi} only depends on α_{uv}, although $\alpha_{uv} \in \theta_{uv}$. x_{uvi} is shaded, because it is observed.

4.2 Monotonicity Property

We would like to model $\mathrm{P}(y = 1|x)$ that increases as $x \to 0$, and decreases as $x \to \infty$ or $x \to -\infty$. We refer to this as the monotonicity property of the conditional probability of agreement. This monotonicity property does not always hold for any or all parameter settings. There are errant parameter settings that may cause this property to be violated. As an example, in Figure 1(b), we show a case where $\mathrm{P}(y = 1|x)$ (the green curve) initially decreases as x goes away from zero, but as x continues moving away, it starts to increase again. This is not intuitive, as it suggests that the probability agreement is very high even as $x \to \infty$.

To enforce the monotonicity property, we propose introducing some constraint to the parameters of the Gaussian mixtures. By expanding Equation 6 according to the generative process, we can express the p.d.f. of $\mathrm{P}(y = 1|x)$ as in Equation 8. Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the p.d.f. of Normal distribution, i.e., $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$.

$$\mathcal{G}(x) = \frac{\alpha\mathcal{N}(x; 0, \sigma_1^2)}{\alpha\mathcal{N}(x; 0, \sigma_1^2) + \frac{1-\alpha}{2}\mathcal{N}(x; \mu_0, \sigma_0^2) + \frac{1-\alpha}{2}\mathcal{N}(x; -\mu_0, \sigma_0^2)} \quad (8)$$

Because the p.d.f $\mathcal{G}(x)$ is continuous and differentiable, one way to ensure that monotonicity holds is to constrain the gradient of $\mathcal{G}(x)$ to be negative for all $x > 0$, as shown in Equation 9. Note that due to the symmetric property of the Gaussian mixtures, it is sufficient to enforce this monotonicity for $x > 0$, as the other case $x < 0$ is met by default.

$$\frac{\partial \mathcal{G}(x)}{\partial x} < 0, \text{for all } x > 0 \quad (9)$$

By taking the derivative of $\mathcal{G}(x)$ with respect to x, Equation 9 can be reduced into the inequality in Equation 10.

$$\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) > 0 \quad (10)$$

This inequality still contains the variable x. We need to reduce it to an inequality involving only the parameters. We discover a simple constraint that meets that objective.

Proposition 1. *The constraint $\sigma_1 < \sigma_0$ ensures that Equation 10 always holds for any $x > 0$.*

Proof. Let us first consider the first additive term in LHS of Equation 10, i.e., $\exp\{\frac{4x\mu_0}{2\sigma_0^2}\}(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2})$. Because x, μ_0, and σ_0 are all positive, we have $\frac{4x\mu_0}{2\sigma_0^2} > 0$. In turn, we have $\exp\{\frac{4x\mu_0}{2\sigma_0^2}\} > 1$. Because $\sigma_1 < \sigma_0$, we also have $(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}) > 0$. We can therefore take Step 1 in Equation 11. From Step 1, we can go to Step 2 by a simple addition of the terms. Finally, because $x > 0$, and $\sigma_1 < \sigma_0$, we have $2x(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0}) > 0$ in Step 3, which concludes the proof.

$$\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) \quad (11)$$

$$\geq \left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) \quad \text{(Step 1)}$$

$$= 2x\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0}\right) \quad \text{(Step 2)}$$

$$> 0 \quad \text{(Step 3)}$$

We have shown that with the constraint of $\sigma_1 < \sigma_0$, Equation 9 holds, guaranteeing the monotonicity property for $x > 0$ (and simultaneously for $x < 0$). This constraint $\sigma_1 < \sigma_0$ is also intuitive, as when two users are agreeing their rating difference is likely to be small and not vary as widely as when they are disagreeing.

4.3 Parameter Estimation

Parameter estimation deals with learning the parameters θ that best "describes" the observed data $X = \{x\}$. Because every x is assumed to have been generated independently in the generative process, the likelihood can be expressed as the joint probability shown in Equation 12.

$$\mathrm{P}(X|\theta) = \prod_{x \in X} \mathrm{P}(x|\theta) \quad (12)$$

The strategy employed in this paper is to find the parameters that maximize the likelihood of observing X. Due to the presence of constraints, the objective is to also find θ that meets the constraints, as shown in Equation 13. The first constraint ensures the mixture weights of the Gaussians sum to 1, by setting the mixture weights to $\alpha_1 = \alpha$ and $\alpha_0 = 1 - \alpha$ respectively. The second constraint ensures the monotonicity of $\mathrm{P}(y = 1|x)$ by setting $\sigma_1 < \sigma_0$.

$$\arg\max_{\theta} \mathrm{P}(X|\Theta),$$
$$\text{subject to: } \alpha_0 + \alpha_1 = 1, \text{and } \sigma_1 < \sigma_0 \quad (13)$$

To maximize the likelihood, we can equivalently maximize the log-likelihood. As it is a constrained optimization problem, we employ the use of Lagrangian multipliers [4] to enforce the constraint. In Equation 14, we show the updated log-likelihood function $\mathcal{L}$. Both λ_α and λ_σ are Lagrangian multipliers for the constraints. We also introduce a slack variable s^2, whose positive value ensures that $\sigma_1 < \sigma_0$.

$$\mathcal{L} = \sum_{x \in X} \ln \mathrm{P}(x|\theta) + \lambda_\alpha(\alpha_1 + \alpha_0 - 1) + \lambda_\sigma(\sigma_0 - \sigma_1 - s^2) \quad (14)$$

To learn the parameters that maximize the log-likelihood function $\mathcal{L}$, we turn to Expectation Maximization (EM) algorithm [3]. It can be shown that the derivation of $\mathcal{L}$ with respect to each parameter leads to the following computations in the E-step and M-step.

In the **E-step**, we compute the following quantities (to be used in the next M-step):

- $c(x) = \frac{1-\alpha}{2\mathrm{P}(x|\Theta)}(\mathcal{N}(x| - \mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_0, \sigma_0^2))$
- $d(x) = \frac{\alpha \mathrm{P}(x|y=1)}{\mathrm{P}(x|\Theta)}$
- $e_1(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)}\mathcal{N}(x| - \mu_0, \sigma_0^2)$
- $e_2(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)}\mathcal{N}(x|\mu_0, \sigma_0^2)$

In the **M-step** we compute μ_0, σ_1, σ_0 and s.

- $\mu_0 = \frac{1}{C}\sum_{x \in X}(e_1(x) - e_2(x))x$, where $C = \sum_{x \in X} c(x)$
- $\alpha = \frac{1}{|X|}\sum_{x \in X} d(x)$
- $\sigma_1^2 = \frac{1}{D}\sum_{x \in X} d(x) \cdot x^2$, where $D = \sum_{x \in X} d(x)$
- $\sigma_0 = (\frac{1}{E}\sum_{x \in X}(e_1(x) \cdot (x+\mu_0)^2 + e_2(x) \cdot (x-\mu_0)^2))^{-\frac{1}{2}} + \sigma_1$, where $E = \sum_{x \in X}(e_1(x) + e_2(x))$

Once the parameters are learned, we can make inferences for the posterior probability of agreement $\mathrm{P}(y = 1|x)$, based on Equation 6, and substituting the learned parameters θ.

5. RATING DIFFERENCE PREDICTION

While *CAM* could explain the distributive properties of x_{uvi}'s and provide an estimation of the contextual agreement probability $\mathrm{P}(y_{uvi}|x_{uvi})$, it assumes that x_{uvi} is known. This is true only for a relatively small subset of triplets. In order to extend the model to unseen triplets, we need to estimate the unseen $\hat{x}_{uvi}$ from ratings data. Inspired by previous work on recommender systems, we adopt an approach based on matrix factorization. While related, our problem is different from traditional recommender systems in two ways. First, the object of interest is a triplet $\langle u, v, i\rangle$, instead of a pair $\langle u, i\rangle$. Second, the value to be estimated x_{uvi} is *rating difference* (see Equation 3), instead of ratings.

We outline three matrix factorization approaches to solve this problem. The first, *PMF*, is an existing approach repurposed for our problem. The second, *PPMF*, is a modification. The third, *DPMF*, is a new proposed method.

5.1 Probabilistic Matrix Factorization (PMF)

One way to predict $\hat{x}_{uvi}$ is to first predict $\hat{r}_{ui}$ and $\hat{r}_{vi}$, and subsequently taking their difference. As a representative of this approach, we employ the *Probabilistic Matrix Factorization* or *PMF* [29]. The set of ratings R can be represented as a matrix of size $|\mathcal{U}| \times |\mathcal{I}|$, where each element corresponds to a rating r_{ui}. This matrix is incomplete, and the goal is to fill up the missing entries with predicted $\hat{r}_{ui}$. The approximation uses two rank-K matrices $S \in \mathbb{R}^{K \times |\mathcal{U}|}$ and $Q \in \mathbb{R}^{K \times |\mathcal{I}|}$.

Let S_u be a column vector in S for user u. Let Q_i be a column vector in Q for item i. *PMF* places zero-mean spherical Gaussian prior distributions on S_u and Q_i (with standard deviations $\varphi_{\mathcal{U}}$ and $\varphi_{\mathcal{I}}$) to control the complexity of the parameters, i.e., $S_u \sim \mathcal{N}(\mathbf{0}, \varphi_U^2\mathbf{I})$ and $Q_i \sim \mathcal{N}(\mathbf{0}, \varphi_I^2\mathbf{I})$. The plate diagram of PMF is shown in Figure 3(a). It shows how ratings are generated by the parameters S_u and Q_i. Each $\hat{r}_{ui}$ is assumed to be drawn from a Gaussian distribution centered at ${S_u}^T Q_i$ with variance γ^2 (Equation 15).

$$\hat{r}_{ui} \sim \mathcal{N}({S_u}^T Q_i, \gamma^2) \quad (15)$$

Parameter estimation is by maximizing the log-posterior distribution over item and user vectors with hyper-parameters, which is equivalent to minimizing the sum of squared-errors function in Equation 16. $\mathbb{I}_R(u, i)$ is an indicator function of whether u has rated i. Equation 16 contains two components. The first summand is the fitting constraint, while the rest constitutes the regularization. The fitting constraint keeps the model parameters fit to the training data whereas the regularizers avoid overfitting, making the model generalize better [7]. λ_U, λ_I are the regularization parameters.

$$E = \frac{1}{2}\sum_{u \in \mathcal{U}}\sum_{i \in \mathcal{I}} \mathbb{I}_R(u, i)(r_{ui} - {S_u}^T Q_i)^2 + \frac{\lambda_U}{2}\sum_{u \in \mathcal{U}} ||S_u||^2 + \frac{\lambda_I}{2}\sum_{i \in \mathcal{I}} ||Q_i||^2 \quad (16)$$

The estimation is done using gradient descent [29], with the following gradients. Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_u}^T Q_i - {S_v}^T Q_i$.

$$\frac{\partial E}{\partial S_u} = -(r_{ui} - {S_u}^T Q_i)Q_i + \lambda_U S_u \quad (17)$$
$$\frac{\partial E}{\partial Q_i} = -(r_{ui} - {S_u}^T Q_i)S_u + \lambda_I Q_i \quad (18)$$

5.2 Pairwise PMF (PPMF)

One potential issue with the previous approach using *PMF* is the indirection of going through ratings, instead of predicting $\hat{x}_{uvi}$ directly. The second approach is to instead fit another matrix $\mathbb{X}$, of size $|\mathcal{U} \times \mathcal{U}| \times |\mathcal{I}|$. Each row corresponds to a pair of users uv. Each column relates to an item i. Each element x_{uvi} is the rating difference $r_{ui} - r_{vi}$.

To approximate $\mathbb{X}$, we associate each user pair with a rank-K vector S_{uv}, and each item with Q_i. To generate $\hat{x}_{uvi}$, we draw it from a Normal distribution, as in Equation 19.

$$\hat{x}_{uvi} \sim \mathcal{N}({S_{uv}}^T Q_i, \gamma^2) \quad (19)$$

We call this approach *Pairwise PMF* or *PPMF*. The plate diagram is shown in Figure 3(b), which clearly illustrates the difference from *PMF*. In *PPMF*, the observations (shaded) are x_{uvi}'s, instead of ratings. The objective function of *PPMF* is specified in Equation 20.

$$\begin{aligned} E = &\frac{1}{2}\sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v}\sum_{i \in \mathcal{I}} \mathbb{I}_R(u, v, i)(x_{uvi} - {S_{uv}}^T Q_i)^2 + \\ &\frac{\lambda_U}{2}\sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v} ||S_{uv}||^2 + \frac{\lambda_I}{2}\sum_{i \in \mathcal{I}} ||Q_i||^2 \end{aligned} \quad (20)$$

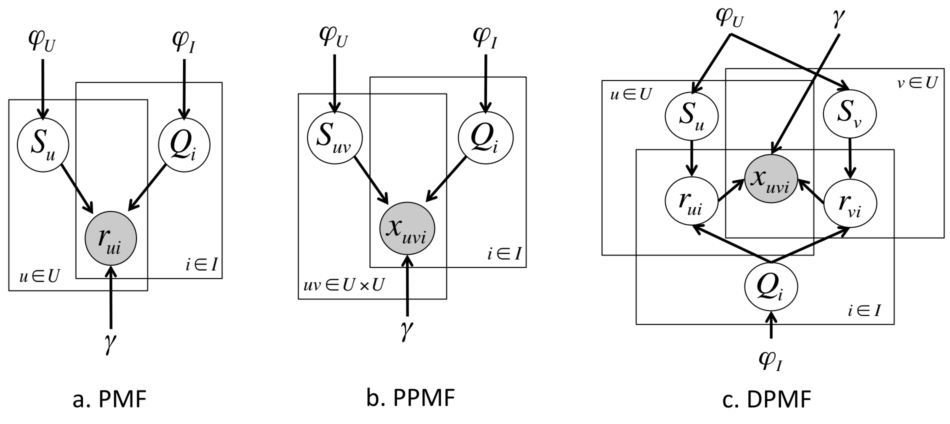


Figure 3: Plate Diagrams: Matrix Factorization Models for Rating Difference Prediction

The estimation is done using gradient descent, with the following gradients.

$$\frac{\partial E}{\partial S_{uv}} = -(x_{uvi} - {S_{uv}}^T Q_i)Q_i + \lambda_U S_{uv} \tag{21}$$

$$\frac{\partial E}{\partial Q_i} = -(x_{uvi} - {S_{uv}}^T Q_i)S_{uv} + \lambda_I Q_i \tag{22}$$

Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_{uv}}^T Q_i$.

5.3 Differential PMF (DPMF)

While *PPMF* estimates $\hat{x}_{uvi}$ directly, it suffers from two design issues. First, it blows up the number of parameters, as we now have to learn the S_{uv} for every pair, instead of every user. Second, it assumes that the vectors S_{uv} and $S_{uv'}$ are independent, even as they share the same user u.

To address these deficiencies, we propose a new factorization model, which we call *Differential Probabilistic Matrix Factorization* or *DPMF*. The plate diagram is shown in Figure 3(c). In this approach, we will still associate each user u with a latent vector S_u, and each item i with Q_i. The key distinction is that we consider ratings to be latent, and fit the rating difference x_{uvi} directly. In other words, $\hat{x}_{uvi}$ is a draw from the following Normal distribution (Equation 23).

$$\hat{x}_{uvi} \sim \mathcal{N}({S_u}^T Q_i - {S_v}^T Q_i, \gamma^2) \tag{23}$$

The objective function of *DPMF* in Equation 24 shows that we fit the prediction $\hat{x}_{uvi} = {S_u}^T Q_i - S_v^T Q_i$ to the observation $x_{uvi} = r_{ui} - r_{vi}$.

$$E = \frac{1}{2} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}, v \neq u} \sum_{i \in \mathcal{I}} \mathbb{I}_R(u,v,i)((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))^2 + \frac{\lambda_U}{2} \sum_{u \in \mathcal{U}} ||S_u||^2 + \frac{\lambda_I}{2} \sum_{i \in \mathcal{I}} ||Q_i||^2 \tag{24}$$

Estimation by gradient descent uses the gradients below.

$$\frac{\partial E}{\partial S_u} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - S_v^T Q_i))Q_i + \lambda_U S_u \tag{25}$$

$$\frac{\partial E}{\partial S_v} = ((r_{ui} - r_{vi}) - ({S_u}^T Q_i - S_v^T Q_i))Q_i + \lambda_U S_v \tag{26}$$

$$\frac{\partial E}{\partial Q_i} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - S_v^T Q_i))(S_u - S_v) + \lambda_I Q_i \tag{27}$$

6. EXPERIMENTS

Our objective in the experiments are three-fold. First, we investigate the learning of *CAM*. Second, we study the effectiveness of different methods in predicting rating differences. Third, we test the combined model against baselines on an evaluative rating prediction task. In addition, we include a case study to better illustrate the workings of *CAM*. Our focus here is on effectiveness, rather than on computational efficiency. We will briefly comment on the runtime of the learning algorithms in the appropriate sections.

6.1 Experimental Setup

Datasets. We conduct experiments on three real-life, publicly available rating datasets, namely: Ciao[1], Epinions[1], and Flixster[2]. Flixster contains ratings on movies. Ciao and Epinions both contain ratings on various categories such as books, electronics, movies, etc. We deliberately do not split the ratings by category to see if the model can contextualize the ratings per item basis without this information. Ratings are normalized into a 5-point scale. In all cases, only *ratings* (and not other information) are used in learning.

We pre-process the raw data as follows. First, we retain only pairs of users who have co-rated at least 20 items. This is to ensure that there is sufficient data to learn the model parameters reasonably accurately. For each co-rated item, we derive x_{uvi} from $r_{ui} - r_{vi}$. In addition, since Flixster has timestamps, we decide to split the ratings into four annual subsets: 2006-2009, and retain only user pairs who exist in all four subsets. This is to see if the results will be consistent across subsets of the data. The data sizes are shown in Table 2. After pre-processing, all the datasets are still sizeable, with thousands of users/items, and tens to hundreds of thousands rating differences.

Training vs. Testing. For each data set we create two types of training/testing data. For Sections 6.2 and 6.3, we work with rating difference triplets x_{uvi}'s. We split the observed triplets X into two subsets: 80% training set X_{train} and 20% testing set X_{test}. We average all the experimental results across 30 such folds (created by random sampling). For Section 6.4, we work with user-item ratings r_{ui}'s. To form the corresponding training set for ratings R_{train} for

[1]`http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm`
[2]`http://www.cs.ubc.ca/~jamalim/datasets/`

Dataset	Original			Preprocessed		
	Users	Items	Ratings	User pairs	Items	Rating Differences
Ciao	10,980	112,832	301,534	3,312	7,425	91,277
Epinions	127,771	331,642	1,185,975	10,997	24,453	369,998
Flixster	147,612	48,794	8,196,077	-	-	-
- Flixster06				1,682	3,421	307,044
- Flixster07				1,682	3,642	106,312
- Flixster08				1,682	3,018	65,210
- Flixster09				1,682	2,127	44,863

Table 2: Datasets

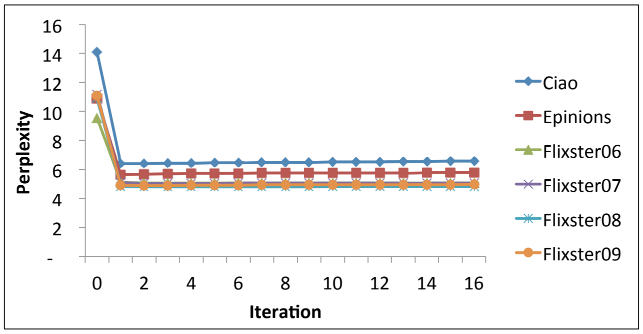


Figure 4: Perplexity of *CAM* on Testing Set

each X_{train}, we "decompose" each x_{uvi} into the original r_{ui} and r_{vi}. Similarly, R_{test} is created from X_{test}, but with an additional step of removing any rating that also exists in R_{train}. Since there are 30 samples for X_{train} and X_{test}, correspondingly there are 30 samples for R_{train} and R_{test}.

6.2 Contextual Agreement Model

Perplexity. First, we study the parameter learning for *CAM*. As mentioned in Section 4, there is a set of parameters θ_{uv}, for every pair of users. One measure of effectiveness for a probabilistic model is *perplexity*, or the ability of model parameters learned from training data (X_{train}) to fit the testing data (X_{test}). Perplexity is measured as $\exp\{-\frac{1}{N}\sum_{m=1}^{N}\log p(x_m)\}$, where N is the number of triplets in the held-out testing data (X_{test}), and $p(x_m)$ is the likelihood of observing the value of a triplet x_m based on the parameter θ. If a model is well-trained, the perplexity will be lower as it gets better at generalizing over the held-out data. To investigate if this is the case, in Figure 4, we plot these perplexity values (averaged over 30 folds each). For each dataset, we measure the perplexity of learned model parameters after every iteration of the EM algorithm. The perplexity decreases quickly in the first few iterations, and then stabilizes. As the EM algorithm converges quickly in improving the fitness of the model parameters to the training data, it also improves the fit with the held-out data.

Distribution of Agreement Prior. To get some sense of the learned parameters, we also inspect the distribution of parameter α_{uv}'s (for different user pairs). This parameter is the prior probability of agreement $\mathrm{P}(y_{uvi} = 1)$ for a pair of users u and v. We show the distribution as a series of white box plots in Figure 5. It shows that in all six datasets, there are diverse types of users. Some user pairs tend to agree ($\alpha \to 1$) while others tend to disagree ($\alpha \to 0$). Most users are somewhere in between. The median hovers around 0.6. In most datasets, the inter-quartile range around the median is 0.3 to 0.4. This result supports our intuition that

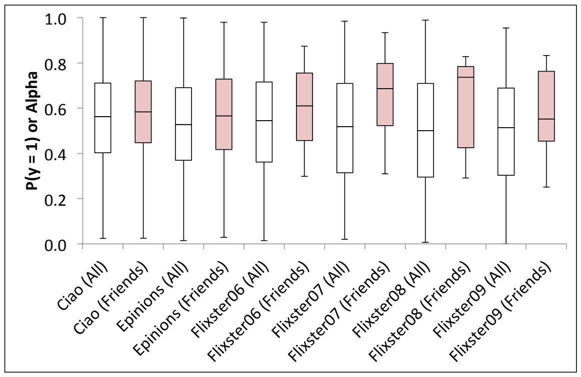


Figure 5: Distribution of $\mathrm{P}(y_{uvi} = 1)$ or α_{uv}

user pairs do not agree all the time. Most will have some disagreements, and therefore it is important to contextualize their agreement on per item basis. Note that this, as well as the earlier conclusion, generally holds for all the annual subsets of Flixster datasets.

Friendship. Since the datasets also contain the social network links among users, we also test the frequently made hypothesis that friendship or trust relationship can help in learning the preferences of users [24, 22]. In the same Figure 5, we draw the distributions of α_{uv}, narrowing down the population to only those user pairs sharing friendship or trustor-trustee relationship. These are drawn as red box plots. One observation is that friendship does contain some information. The comparison of every pair of white (all pairs) vs. red (friends-only) box plots, show that friends have greater agreement in general. This is especially evident in the Flixster datasets. However, another interesting observation is that even some friends disagree a lot, as shown by the lower whiskers of the box plots. Hence, just because a pair of users are friends, it does not mean they always agree. Therefore, it is helpful to know the context of agreement.

The EM learning algorithms are relatively efficient. For each fold, the parameters for all user pairs can be learned in 1 to 4 minutes on an Intel(R) Xeon(R) Processor E5-2667 2.90GHz machine.

6.3 Rating Difference Prediction

We study the efficacy of different matrix factorization methods outlined in Section 5 (*PMF*, *PPMF* and *DPMF*) in deriving good predictions for unseen triplets. *PPMF* and *DPMF* are trained on X_{train}, while *PMF* is trained on the corresponding R_{train}, all using the same parameter choices as in the original paper for PMF [29] (learning rate = 0.005, number of latent factors = 30, regularization coefficient = 0.002). All three are tested on the same X_{test}.

For every triplet x_{uvi} in the test set X_{test}, we derive a prediction $\hat{x}_{uvi}$ using each method, and compare the accuracy of their predictions in terms of root mean squared error commonly used in matrix factorization. $RMSE_{diff}$ is defined in Equation 28. Lower value indicates better performance.

$$RMSE_{diff} = \sum_{x_{uvi} \in X_{test}} \sqrt{\frac{(\hat{x}_{uvi} - x_{uvi})^2}{|X_{test}|}} \qquad (28)$$

Vary Epochs. In Figure 6, we plot the $RMSE_{diff}$ at different epochs. One epoch corresponds to a full iteration

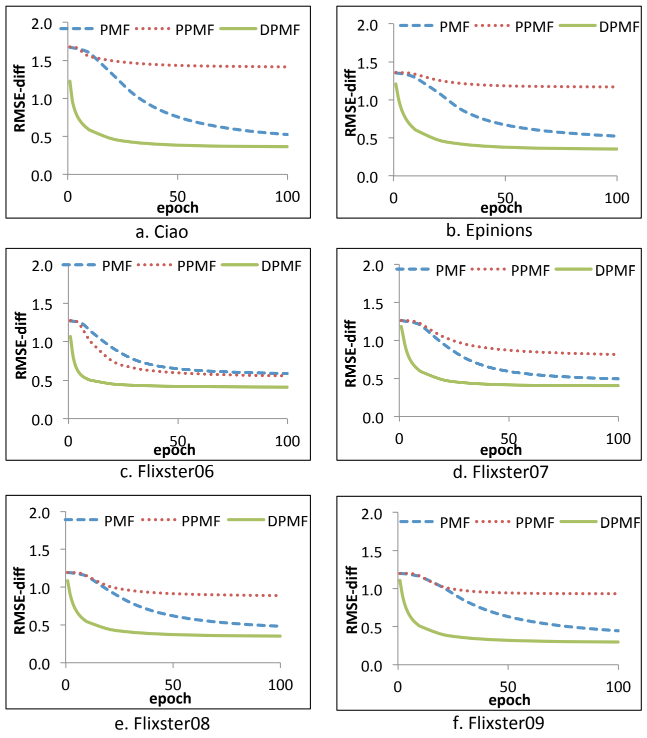


Figure 6: PMF vs. PPMF vs. DPMF ($RMSE_{diff}$)

over the whole training set. For all, the error goes down with the epochs, and eventually converges. *DPMF* performs the best in two respects. First, its converged error is the lowest of the three, followed by *PMF*, and *PPMF* (worst). Second, it achieves convergence much faster (by 30 epochs). Although by 100 epochs, *PMF* narrows down the error gap somewhat, it converges very slowly, requiring more epochs.

We hypothesize that this is due to the differences in the objective functions. *PMF* tries to make its prediction as close to the observed rating as possible, without consideration on the level of difference between ratings. For example, suppose users u and v give ratings of 4 and 1 respectively to the same item in the test set. If the predicted ratings are 4.5 and 0.5, these are close enough to the actual ratings (4 and 1). However, in terms of the rating *difference*, it has widened from $4-1=3$ to $4.5-0.5=4$. In contrast, *DPMF* tries to fit the rating difference directly, for instance by predicting 4.5 and 1.5, which has the same error in terms of rating, but zero error in terms of rating difference.

We perform one-tailed t-test with 0.01 significance level on the $RMSE_{diff}$ values of *PMF* and *DPMF* over different epochs. The result confirms that the outperformance by *DPMF* over *PMF* is statistically significant.

Vary Latent Factors. We conduct a separate experiment on *DPMF* on different numbers of latent factors K. The $RMSE_{diff}$ at 100 epochs are shown in Table 3. It shows that by around $K=30$, the errors have converged. There is no significant gain by running higher latent factors (which will make the learning algorithms slower). Subsequently, we will use *DPMF* in conjunction with *CAM* with the same parameter settings ($K=30$, 100 epochs) .

The gradient descent learning algorithms are also efficient. For all three methods, the parameters can be learned within

Dataset	Number of latent factors K					
	10	20	30	40	50	100
Ciao	0.87	0.43	0.36	0.36	0.35	0.34
Epinions	0.77	0.45	0.35	0.34	0.33	0.32
Flixster06	0.78	0.55	0.41	0.33	0.29	0.23
Flixster07	0.65	0.47	0.40	0.38	0.37	0.35
Flixster08	0.62	0.42	0.35	0.34	0.33	0.32
Flixster09	0.58	0.35	0.30	0.29	0.28	0.28

Table 3: DPMF: Vary Latent Factors ($RMSE_{diff}$)

1 minute for each fold on the same Intel(R) Xeon(R) Processor E5-2667 2.90GHz machine.

6.4 Application: Collaborative Filtering

Here, we use the model parameters of *CAM*, combined with the rating difference predictions by *DPMF* to generate contextual agreement probabilities $w_{uvi} = \mathrm{P}(y_{uvi}=1|\hat{x}_{uvi})$. These probabilities are used as similarity in neighborhood-based collaborative filtering, as outlined in Section 3.

In the rating prediction task, for every rating $r_{ui} \in R_{test}$, we predict $\hat{r}_{ui}$ as a weighted average of neighbors' ratings in R_{train}. The accuracy of rating prediction is measured by $RMSE_{rating}$ defined in Equation 29.

$$RMSE_{rating} = \sum_{r_{ui} \in R_{test}} \sqrt{\frac{(\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}} \qquad (29)$$

Contextual vs. Shared. First, we compare the efficacy of item-specific contextual agreement (labeled *CAM-DPMF*) as compared to baselines relying on shared preference that applies to all items of the same user pair, as measured by Pearson and Cosine functions (see Section 2).

The prediction accuracies in terms of $RMSE_{rating}$ are listed in Table 4. For each dataset, we indicate with an '$*$' the best method with the lowest error, which is significantly different from the second-best (using t-test significance test at 0.01 significance level). For all of the datasets, *CAM-DPMF* has the lowest errors. For Ciao, *CAM-DPMF* has a lower error than Cosine or Pearson, but not statistically significant at $p=0.01$. As all the comparative methods work with exactly the same set of ratings, the only difference is how each method weighs the contribution of each rating. This result shows that paying attention to context, as *CAM-DPMF* does, helps to gain a lower prediction error.

Combination vs. Components. *CAM-DPMF* uses a combination of *CAM*'s model parameters and *DPMF*'s predicted rating differences. To show that this joining of the two components is really necessary, it is instructive to see how each respective component performs on the same task. We therefore construct two more baselines based on each component respectively. The first, called *CAM*-α, uses the α_{uv} of each pair as a non-contextual similarity value in Equation 5. The second is the factorization model *DPMF* described in Section 5. We also include *PMF* for completeness. We use the users' and items' parameters S_u and Q_i to predict unobserved ratings $\hat{r}_{ui}$. Table 5 shows a comparison between the combined approach *CAM-DPMF* and the two components, *CAM*-α and factorization models on the rating prediction task. In five out of six datasets, *CAM-DPMF* has a lower error than both components. One exception is Flixster06, where *PMF* performs slightly better. Interestingly, *DPMF* performs very badly on its own. This

Dataset	CAM-DPMF	Shared Preference	
		Cosine	Pearson
Ciao	1.110*	1.119*	1.118*
Epinions	1.141*	1.180	1.180
Flixster06	1.084*	1.144	1.143
Flixster07	1.011*	1.060	1.058
Flixster08	1.051*	1.081	1.079
Flixster09	1.087*	1.148	1.146

Table 4: Versus Shared Preference (statistically significant best-performing entries are asterisked)

Dataset	CAM-DPMF	CAM-α	DPMF	PMF
Ciao	1.110*	1.129	4.181	1.183
Epinions	1.141*	1.198	4.075	1.194
Flixster06	1.084	1.150	3.446	1.046*
Flixster07	1.011*	1.073	3.532	1.073
Flixster08	1.051*	1.095	3.617	1.095
Flixster09	1.087*	1.152	3.595	1.152

Table 5: Versus Model Components ($RMSE_{rating}$)

is because it is optimized for predicting rating differences, and not ratings. The results emphasize the improvement of *CAM-DPMF* over shared preference comes from the complementary combination of both components, and not from the sole contribution of either one.

6.5 Case Study

To illustrate the workings of *CAM*, we now show a case study drawn from the Epinions dataset, involving the same pair of users as in Section 1. Table 6 shows the ratings of user u (*talyseon*) and v (*youngching*) on twenty movies.

Based on these ratings, the *CAM* parameters for this pair are as follows: $\alpha = 0.40$, $\mu_0 = 2.9$, $\sigma_0 = 0.83$, $\sigma_1 = 0.81$. The relatively low α suggests that this pair do not always agree. That $\mu_0 = 2.9$ suggests that when they disagree their rating difference is around 3. This is evident from the fourth column labeled $|x_{uvi}|$, which tracks their rating differences. The lower half of the table shows rating differences around 3, suggesting that these are movies the pair disagree on.

CAM uses these parameters to estimate the contextual probability of agreement shown in the fifth column. As expected, the contextual probability of agreement is high (close to 1) for the movies at the upper half of the table (where rating differences are low), and is low (close to 0) for the movies at the lower half. In contrast to the item-specific agreement produced by *CAM*, the baselines Pearson and Cosine each assign a single similarity value that applies to all items, inadequately describing the nature of agreement between users.

To see that such cases of varying rating differences are common, we employ the concept of entropy from information theory. For each pair, we count the frequencies of rating differences, and measure the entropy, i.e., $\sum_{i=1} p(x_i) \ln p(x_i)$ where $p(x_i)$ is the normalized frequency of each rating difference value. If the entropy is high, the pair has rating differences that are varied, rather than uniform (if entropy is low). For instance, the user pair in the case study above has an entropy of 2.3. Figure 7 plots a histogram of user pairs binned by their entropies. There is a significant proportion of the population with high entropies. In fact, the low entropies are the exception, rather than the norm.

Movie	r_{ui}	r_{vi}	$\lvert x_{uvi}\rvert$	P($y_{uvi} = 1\lvert x_{uvi}$)	Pearson	Cosine
Paranormal Activity	5	5	0	1.00	0.53	0.88
Payback	3	3	0	1.00		
Coraline	5	5	0	1.00		
Pan's Labyrinth	5	5	0	1.00		
Memento	5	4	1	0.89		
Gran Torino	5	4	1	0.89		
The Hurt Locker	5	4	1	0.89		
Jurassic Park III	3	2	1	0.89		
Twilight	3	1	2	0.10		
Inception	5	3	2	0.10		
Daredevil	3	1	2	0.10		
I Am Legend	4	2	2	0.10		
Rosemary's Baby	5	2	3	0.00		
The Day After Tomorrow	4	1	3	0.00		
300	4	1	3	0.00		
Moulin Rouge	5	2	3	0.00		
Seven Pounds	4	1	3	0.00		
The Dark Knight	5	1	4	0.00		
The Last Samurai	5	1	4	0.00		
Star Wars Episode III: Revenge of the Sith	5	1	4	0.00		

Table 6: Epinions Case Study

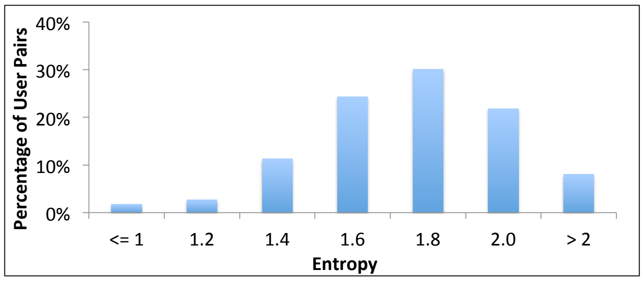


Figure 7: Entropy of Rating Differences in Epinions

7. CONCLUSION

We address the novel problem of estimating the contextual agreement between two users in the context of one item, by probabilistic modeling with two major components. The first, called *CAM*, models contextual agreement in generative form, as a mixture of Gaussians. To ensure monotonic behavior of the agreement probability, we propose a specific constraint, and describe how the constrained parameters can be learned through EM. To extend the use of *CAM* to unseen triplets, the second component predicts rating differences between two users on the same item. We outline three different matrix factorization approaches, including a proposed model called *DPMF* with a novel objective function. The models are shown to be effective through experiments on real-life rating datasets. As future work, we plan to investigate how the two components of our model can be joined more tightly together, such that the learning for one can help reinforce the other. In addition, just as we could apply *CAM-DPMF* in similarity-based collaborative filtering, it may be feasible to apply it in matrix factorization for rating prediction as well, which requires further investigation.

8. ACKNOWLEDGMENTS

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6), 2005.

[2] A. Ahmed, B. Kanagal, S. Pandey, V. Josifovski, L. G. Pueyo, and J. Yuan. Latent factor models with additive and hierarchically-smoothed user preferences. In *WSDM*, 2013.

[3] C. M. Bishop and N. M. Nasrabadi. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[6] H. Fang, Y. Baoy, and J. Zhang. Misleading opinions provided by advisors: Dishonesty or subjectivity. In *IJCAI*, 2013.

[7] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2011.

[8] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR*, 2003.

[9] T. Hofmann. Latent semantic models for collaborative filtering. *TOIS*, 22(1), 2004.

[10] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, 2013.

[11] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[12] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR*, 2004.

[13] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *PVLDB*, 5(10), 2012.

[14] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, 2011.

[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[16] Y. Koren and J. Sill. OrdRec: An ordinal model for predicting personalized item rating distributions. In *RecSys*, 2011.

[17] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *ICML*, 2009.

[18] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 1999.

[19] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 2003.

[20] X. Liu and K. Aberer. SoCo: a social network aided context-aware recommender system. In *WWW*, 2013.

[21] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[22] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *SIGIR*, 2009.

[23] H. Ma, H. Yang, M. R. Lyu, and I. King. SoRec: Social recommendation using probabilistic matrix factorization. In *CIKM*, 2008.

[24] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, 2011.

[25] L. W. Mackey, D. Weiss, and M. I. Jordan. Mixed membership matrix factorization. In *ICML*, 2010.

[26] L. B. Marinho, A. Nanopoulos, L. Schmidt-Thieme, R. Jäschke, A. Hotho, G. Stumme, and P. Symeonidis. Social tagging recommender systems. In *Recommender Systems Handbook*, pages 615–644. Springer, 2011.

[27] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *KDD*, 2011.

[28] R. Missaoui, P. Valtchev, C. Djeraba, and M. Adda. Toward recommendation based on ontology-powered web-usage mining. *IEEE Internet Computing*, 11(4), 2007.

[29] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 2007.

[30] W. Pan and L. Chen. GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, 2013.

[31] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The Adaptive Web*, pages 325–341. Springer, 2007.

[32] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*, 1994.

[34] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, 2008.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[36] H. Shan, J. Kattge, P. B. Reich, A. Banerjee, F. Schrodt, and M. Reichstein. Gap filling in the plant kingdom trait prediction using hierarchical probabilistic matrix factorization. In *ICML*, 2012.

[37] Y. Shen and R. Jin. Learning personal + social latent factor model for social recommendation. In *KDD*, 2012.

[38] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, 2008.

[39] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[40] J. Wang, Y. Zhang, C. Posse, and A. Bhasin. Is it time for a career switch? In *WWW*, 2013.

[41] E. Zhong, W. Fan, and Q. Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM*, 2012.