Probabilistic Models for Contextual Agreement in Preferences

LOC DO and HADY W. LAUW, Singapore Management University

The long-tail theory for consumer demand implies the need for more accurate personalization technologies to target items to the users who most desire them. A key tenet of personalization is the capacity to model user preferences. Most of the previous work on recommendation and personalization has focused primarily on individual preferences. While some focus on shared preferences between pairs of users, they assume that the same similarity value applies to all items. Here we investigate the notion of "context," hypothesizing that while two users may agree on their preferences on some items, they may also disagree on other items. To model this, we design probabilistic models for the generation of rating differences between pairs of users across different items. Since this model also involves the estimation of rating differences on unseen items for the purpose of prediction, we further conduct a systematic analysis of matrix factorization and tensor factorization methods in this estimation, and propose a factorization model with a novel objective function of minimizing error in rating differences. Experiments on several real-life rating datasets show that our proposed model consistently yields context-specific similarity values that perform better on a prediction task than models relying on shared preferences.

CCS Concepts: • **Information systems** → **Data mining**; Recommender systems; • **Human-centered computing** → Collaborative filtering

Additional Key Words and Phrases: User preference, contextual agreement, generative model

ACM Reference Format:
Loc Do and Hady W. Lauw. 2016. Probabilistic models for contextual agreement in preferences. ACM Trans. Inf. Syst. 34, 4, Article 21 (June 2016), 33 pages.
DOI: http://dx.doi.org/10.1145/2854147

1. INTRODUCTION

Personalization is at the heart of many existing web platforms, providing a customized experience to each individual user. The key to personalization is how to model the preferences of each user. Modeling *individual preferences* aims to derive user-specific models from preference data (e.g., ratings). There are several well-known methods, such as the aspect model [Hofmann 2003, 2004], matrix factorization [Koren et al. 2009], and the content-based model [Adomavicius and Tuzhilin 2005; Pazzani and Billsus 2007].

Beyond individual preferences, a significant body of work focuses on the complementary issue of *shared preference* between pairs of users, for example, neighborhood-based collaborative filtering systems [Jin et al. 2004; Resnick et al. 1994; Breese et al. 1998]. 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

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).
Authors' addresses: L. Do and H. W. Lauw, School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902; emails: {hldo.2012, hadywlauw}@smu.edu.sg.

DOI: http://dx.doi.org/10.1145/2854147

Table I. MovieLens Users u_{38} and u_{197}

Movie	u_{38}'s Rating	u_{197}'s Rating	u_{38}'s Rating − u_{197}'s Rating
Conan the Barbarian	5	1	4
Volcano	5	2	3
First Knight	5	2	3
Scream	5	3	2
G. I. Jane	5	3	2
George of the Jungle	3	1	2
Titanic	5	4	1
Liar Liar	5	4	1
Top Gun	5	4	1
Braveheart	5	5	0
Jurassic Park	5	5	0
Conspiracy Theory	4	4	0
Die Hard (1995)	2	3	−1
Full Metal Jacket	2	3	−1
The Fugitive	3	5	−2
Batman (1989)	1	3	−2
The Godfather	2	5	−3
Die Hard 2 (1990)	1	4	−3
Ben Hur	1	5	−4
The Terminator	1	5	−4

model to be built. However, the limited record may already be sufficient to infer one's 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.

Shared preference implicitly assumes that the similarity between two users applies equally to all items under consideration. Realistically, users have diverse preferences. 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. To retain the most common framework in the literature, which is to rely on rating data alone, in this article, by "context," we refer to each specific item. In other words, we are interested in the *contextual agreement* of preferences between two users in the context of one item.

Example. To illustrate this, we use a real-life example from the MovieLens[1] dataset. In Table I, we show the ratings by two users (identified by their anonymized IDs u_{38} and u_{197}, respectively) on 20 movies that both of them had rated. The ratings are from 1 (low) to 5 (high). In addition to the ratings by the users on each movie, we indicate their rating differences. The top few movies in the list (e.g., Conan the Barbarian) are movies to which u_{38} assigns high ratings but u_{197} assigns low ratings, yielding large positive rating differences. The movies in the middle (shaded rows) are those that u_{38} and u_{197} tend to agree on, such as when both like the same movie (e.g., Jurassic Park). The movies at the bottom of the list are movies that u_{38} dislikes but u_{197} likes (e.g., Ben Hur, The Terminator), yielding large negative rating differences.

Evidently, u_{38} and u_{197} agree on some movies, and yet disagree on other movies. However, the traditional approach of shared preference is to measure the overall similarity between the two users. Using Pearson's correlation (in the range from -1 to 1),

[1]http://grouplens.org/datasets/movielens/.

their correlation is -0.25, indicating slight differences. Using cosine similarity (in the range from 0 to 1), their similarity is 0.81, indicating very high similarity. See Section 3 for the definitions of these measures. Hence, these two different similarity measures yield very different conclusions. This drives home the point that a single value cannot reveal the complex picture of the varying preferences of users. Therefore, agreement on preference should be seen in the context of individual items.

Problem. Given a set of users, a set of items, and some ratings by users on items, we seek to model the contextual agreement between a pair of users on a specific item. One key observation is that the observed rating values provide signals of the agreement or disagreement. As suggested by Table I, when there is a large difference in ratings, the two users are likely to disagree. Modeling the agreement in preferences gives rise to two subproblems. The first is how to express the agreement and disagreement in a more principled fashion. Rather than having yet another real-valued similarity, we propose to adopt a probabilistic modeling, expressing the probability that two users agree on a specific item. The second is that not all rating differences are observed, arising directly from not having observed all possible ratings. Therefore, there is a need to also predict the “unseen” rating differences.

Application. Modeling contextual agreement allows for a better estimation of the similarity between a pair of users on a specific item. This may be useful in several potential applications. For one, the agreement probability can calibrate the similarities between neighbors in an item-specific manner for a neighborhood-based recommender system. For another, it supports a scenario where a user recommends an item to his or her friends, based on whether the friends would have similar responses. The model could help to identify the subset of friends in agreement on the item. Furthermore, the model may also be useful in a study of prevalence of agreement in different communities.

Scope. While our work is related to recommender systems, our focus is on modeling agreement in 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 rating data, and not other information such as categories or ontologies [Missaoui et al. 2007]. We also assume that ratings are truthful and reflective of user preferences (and not artifacts of dishonesty or fraud [Fang et al. 2013]), which we believe is true for most users.

Contributions. In this article, we make the following contributions:

(1) To our best knowledge, this work is the first to propose modeling an item-specific context in estimating the agreement between a pair of users on an item.
(2) To realize this modeling, in Section 4, we develop a probabilistic generative model, called *Contextual Agreement Model* or *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.
(3) To extend this model to unseen triplets, in Section 5, we explore several matrix and tensor factorization methods. We also propose a new method, called *Differential Probabilistic Matrix Factorization* or *DPMF*, with a novel objective function that minimizes errors in rating differences.
(4) In Section 6, we validate these models comprehensively on real-life, publicly available rating datasets.

2. OVERVIEW

We now present an overview of our framework, which provides both the formal problem formulation and the high-level organization of the article.

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 consists of $\mathcal{U} \times \mathcal{U} \times \mathcal{I}$, excluding triplets involving the same users, for example, $\langle u, u, i\rangle$. 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 indicators 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, that is, $x_{uvi} = \mathcal{F}(r_{ui}, r_{vi})$. While there are many possible definitions of $\mathcal{F}$, in this article, we simply use the *rating difference* between two users on the same item, as shown in Equation (1):

$$x_{uvi} = r_{ui} - r_{vi}. \tag{1}$$

This choice of function also implies the relationship $x_{uvi} = -x_{vui}$.

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 are to be estimated from the observed x_{uvi}s.

Problem Formulation. Given rating data R and the previous 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 subproblems. The first is how to estimate $\mathrm{P}(y_{uvi}|x_{uvi})$ given the observed x_{uvi} values. The second is how to predict the unobserved $\hat{x}_{uvi}$ values.

For the first subproblem, we propose the probabilistic *CAM* model in Section 4. Since y_{uvi} is latent, we 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 afterward be estimated from the joint probabilities as follows:

$$\mathrm{P}(y_{uvi}|x_{uvi}) = \frac{\mathrm{P}(y_{uvi}, x_{uvi})}{\sum_{y'_{uvi}\in\mathcal{Y}} \mathrm{P}(y'_{uvi}, x_{uvi})}. \tag{2}$$

The second subproblem is how to predict the unseen $\hat{x}_{uvi}$. We will then use the predicted $\hat{x}_{uvi}$ with the parameters learned in the first subproblem to estimate $\mathrm{P}(y_{uvi}|\hat{x}_{uvi})$. One 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 to generate/predict unseen $\hat{x}_{uvi}$ through matrix or tensor factorization in Section 5.

Applications. As mentioned in Section 1, the agreement probabilities may be used in applications that benefit from knowing the agreement between two users. In this work, we will experiment with two applications.

One application of the agreement probabilities is as a similarity value in a neighborhood-based collaborative filtering (CF). User-based CF [Jannach et al. 2010] exploits the similarities between users to predict unseen ratings. Adopting the same rating prediction framework, we can use the agreement to weigh the contributions of neighbors. To predict an unseen rating $\hat{r}_{ui}$, we use Equation (3), which is the weighted average of ratings on i by u's neighbors. Neighbor v can be any user, weighted by w_{uvi}:

$$\hat{r}_{ui} = \frac{\sum_{v\neq u, r_{vi}\neq\phi} w_{uvi} \times r_{vi}}{\sum_{v\neq u, r_{vi}\neq\phi} w_{uvi}}. \tag{3}$$

In our case, we use $w_{uvi} = \mathrm{P}(y_{uvi}|x_{uvi})$, 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 [Resnick et al. 1994] and cosine similarity [Breese et al. 1998], which are reviewed in Section 3. The comparison is equitable 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.

Another application is to facilitate a special form of social recommendation. For instance, a user may wish to recommend an item only to specific neighbors [Bhatt et al. 2010], who would be expected to have a similar response. Given u and one of the user's adopted/rated items i, the task is to rank two of his or her neighbors v and z, whose preferences on i are not observed, based on their predicted rating differences with u on i (see Section 6).

3. RELATED WORK

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 [Adomavicius and Tuzhilin 2005]. 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 the *aspect model* [Hofmann 2003, 2004]. 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 has a distribution over items i to be adopted, that is, $\mathrm{P}(i|z_k)$, or ratings r, that is, $\mathrm{P}(r|z_k, i)$.

The second is *matrix factorization* [Koren et al. 2009]. User u's preference is modeled as a column vector S_u in a K-dimensional latent space. Each item i is associated with a rank-K column vector Q_i. The rating prediction $\hat{r}_{ui}$ is given by ${S_u}^T Q_i$. There are different methods [Lee and Seung 1999; Srebro et al. 2004; Lawrence and Urtasun 2009; Mackey et al. 2010], which vary in their objective functions, including probabilistic variants [Mnih and Salakhutdinov 2007; Salakhutdinov and Mnih 2008].

The third is the *content-based model* [Adomavicius and Tuzhilin 2005; Pazzani and Billsus 2007; Lops et al. 2011]. 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., metadata, text) of items that u likes.

Shared Preference. Modeling sharing of preferences is mostly found in neighborhood-based recommender systems [Jannach et al. 2010]. One approach is based on *similarity*. For user-based collaborative filtering (CF) [Jin et al. 2004], 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 [Resnick et al. 1994] and vector space or cosine similarity [Breese et al. 1998]. 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 (4) (where $\bar{r}_u$ and $\bar{r}_v$ are average ratings), and cosine as in Equation (5). For item-based CF [Sarwar et al. 2001; Linden et al. 2003], the similarity is between a pair of items.

$$w_{uv}^{pearson} = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2}\sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}} \tag{4}$$

$$w_{uv}^{cosine} = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u|| \times ||\mathbf{r}_v||}. \tag{5}$$

Another approach 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 [Ma et al. 2009, 2011]. Some exploit the taxonomy structure to induce sharing between items in the same category [Shan et al. 2012; Ahmed et al. 2013; Kanagal et al. 2012; Menon et al. 2011; Koenigstein et al. 2011].

Contextualized Preference. Most works base their approaches on the dyad of user-item pair. In some cases, additional information or "context" may be available. Rather than pairs $\langle u, i \rangle$, we observe triplets $\langle u, i, c \rangle$, where c refers to some context such as time [Xiong et al. 2010; Koren 2010], location [Levandoski et al. 2012], tags [Rendle and Schmidt-Thieme 2010], and so forth. Here, we briefly describe two common approaches to dealing with triplets, that is, incorporating contextual information into a preference model. A broader overview can be found in Adomavicius and Tuzhilin [2011].

The first direction is to separate the ratings of different contexts and model each context independently. For example, suppose each day of the week is a separate context; we could build seven matrix factorization models corresponding to every day of the week [Koren 2010]. Levandoski et al. [2012] consider a similar strategy but use locations as contexts. While applicable to those cases, this approach is not suitable for our problem, because we are interested in individual items as contexts.

The second direction is to model triadic relationships $\langle u, i, c \rangle$ directly through another form of factorization called tensor factorization. The basic form of tensor factorization is *Tucker Decomposition* [Tucker 1966]. To improve efficiency, several works propose special forms of tensor decomposition, such as *Canonical Decomposition* [Xiong et al. 2010] and *Pairwise Interaction Tensor Factorization* [Rendle and Schmidt-Thieme 2010]. While tensor factorization cannot solve our problem directly, in Section 5, we show how it may be adapted into a subcomponent of our model, with a specific modification to fit our scenario of modeling rating differences.

4. CONTEXTUAL AGREEMENT MODEL

In this section, we describe the generative model for *CAM*, outline the monotonicity property of its "decision function," and develop an algorithm to learn its parameters.

4.1. Generative Model

Given the observed x_{uvi}s, we 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, that is, 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 is generated according to $\mathrm{P}(x|y = 1)$. As x is real-valued, and we expect that its values will cluster together in the event of agreement, we model its generation as a Gaussian, with a mean μ_1 and variance σ_1^2. As mentioned in Section 2, the closer x_{uvi} is 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, $\sigma_1 = 0.9$).

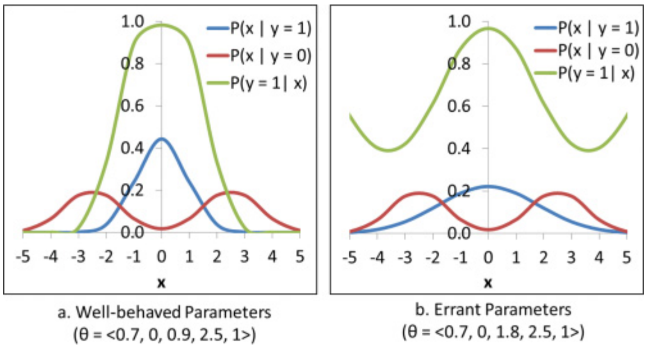


Fig. 1. Distributions of P($x|y$) and P($y|x$).

In the event of disagreement ($y = 0$), x is generated according to $\mathrm{P}(x|y = 0)$. 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 $x_{uvi} = -x_{vui}$, a reasonable model is 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 in 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 $y \in \{0, 1\}$:

$$y \sim \mathit{Bernoulli}(\alpha).$$

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

(a) In the event of agreement, that is, $y = 1$:

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

(b) Otherwise, in the event of disagreement, that is, $y = 0$:

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

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):

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

Parameters. For the previous 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

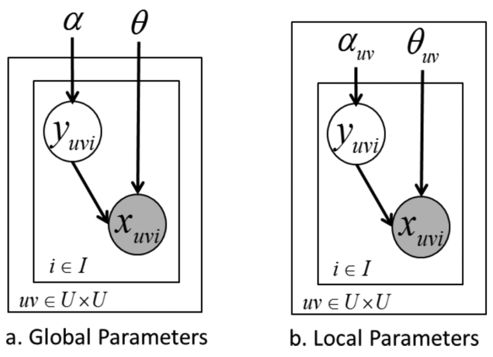


Fig. 2. Global versus Local parameters.

estimate θ from a single observation of x. Another approach is to tie together the parameters of a group of triplets. In this article, we will experiment with two approaches. First is the *Global* parameter, where θ is shared by all triplets. Second is the *Local* parameter, where there is a specific θ_{uv} for each pair of users u and v that applies to all items. The distinction between these two approaches can be seen clearly in the plate diagrams in Figure 2. For clarity, we draw α separately to show that y_{uvi} only depends on α, although $\alpha \in \theta$. In both cases, x_{uvi} is shaded, because they form the observations. For *Local*, θ_{uv} is within the plate of each pair of users. For *Global*, θ is outside.

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 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 counterintuitive, as it suggests that the probability of agreement is very high even as $x \to \infty$.

To enforce the monotonicity property, we propose introducing some constraints 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):

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

Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the p.d.f. of Normal distribution, that is, $\frac{1}{\sqrt{2\pi\sigma^2}}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$.

Because the p.d.f $\mathcal{G}(x)$ is continuous and differentiable, one way to ensure monotonicity is to constrain the gradient of $\mathcal{G}(x)$ to be negative for $x > 0$, 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 simultaneously:

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

After taking the derivative of $\mathcal{G}(x)$ w.r.t. x, we reduce Equation (9) into the inequality in Equation (10). The full step-by-step derivation is described in Appendix A.1:

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

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

PROPOSITION 4.1. *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 the left-hand side of Equation (10), that is, $\exp\{\frac{4x\mu_0}{2\sigma_0^2}\}(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2})$. 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):

$$\begin{aligned}
&\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) && (11)\\
&\geq \left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) && \text{(Step 1)}\\
&= 2x\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0^2}\right) && \text{(Step 2)}\\
&> 0. && \text{(Step 3)}
\end{aligned}$$

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. □

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

We seek to learn the parameters θ that best "describe" 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):

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

The strategy employed in this article 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 $\sigma_1 < \sigma_0$:

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

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 [Boyd and Vandenberghe 2004] 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. We introduce a slack variable s^2, whose positive value ensures that $\sigma_1 < \sigma_0$:

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

To learn the parameters that maximize the log-likelihood function $\mathcal{L}$, we turn to the Expectation Maximization (EM) algorithm [Bishop and Nasrabadi 2006]. The full derivation of $\mathcal{L}$ with respect to each parameter is given in Appendix A.2. Here, we show the computations in the E-step and M-step. The E-step and the M-step are computed iteratively till convergence.

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* explains the distributive properties of x_{uvi}s and provides the contextual agreement probability $\mathrm{P}(y_{uvi}|x_{uvi})$, it assumes that x_{uvi} is given. However, x_{uvi} is observed only for a relatively small subset of triplets. In order to extend the model's applicability to *unseen triplets*, we need to estimate these unseen triplets $\hat{x}_{uvi}$ from rating data. For this purpose, there are three lines of approaches that flow naturally from the problem setting. These approaches will be systematically investigated in the coming subsections. Their distinctions can also be identified from their graphical representations in Figure 3. In the following factorization framework, we do not incorporate bias terms [Koren et al. 2009] (an orthogonal issue), in order to isolate the effects of the structure of the matrix and tensor factorizations.

As $\hat{x}_{uvi}$ is essentially the difference between ratings $\hat{r}_{ui}$ and $\hat{r}_{vi}$, the **first** line of approaches rely on predicting these unseen ratings $\hat{r}_{ui}$ and $\hat{r}_{vi}$, and then taking their difference. Matrix factorization for rating prediction can be adapted for this purpose, as shown in Section 5.1. This approach assumes that fitting ratings will lead to fitting rating differences. This does not always hold, because fitting ratings optimize a different objective function, that is, minimizing residual error in predicted ratings. For instance, suppose that the true ratings are $r_{ui} = 4$ and $r_{vi} = 3$, which implies the rating difference $x_{uvi} = 1$. If the rating prediction model has an error of 0.1 for each rating, we may end up with an estimation of $\hat{r}_{ui} = 4.1$ and $\hat{r}_{vi} = 3.1$ that still preserves the rating difference, or with an alternative estimation $\hat{r}_{ui} = 4.1$ and $\hat{r}_{vi} = 2.9$ that enlarges the rating difference.

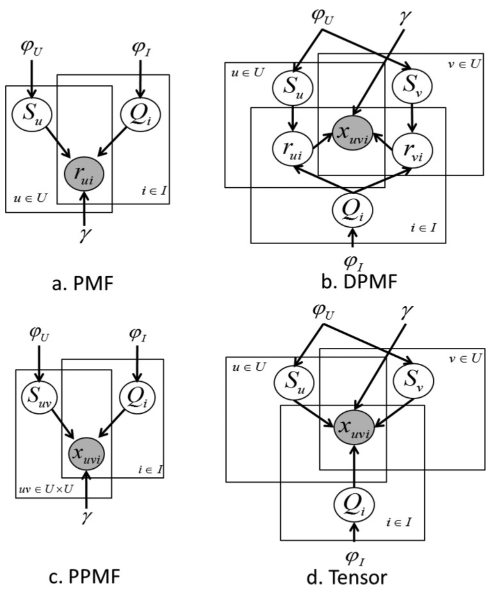


Fig. 3. Plate diagrams: Factorization models for rating difference prediction.

Hence, we propose the **second** line of approach that is still based on the same concept of completing the rating matrix but with a different objective function that directly minimizes the residual errors of rating differences, as described in Section 5.2.

Instead of decomposing $\hat{x}_{uvi}$ into its constituent ratings, the **third** line of approaches rely on predicting the triplets $\hat{x}_{uvi}$ directly. This can be accomplished by either matrix factorization or tensor factorization, both of which we will explore in Section 5.3.

5.1. Factorizing Ratings

One way to predict $\hat{x}_{uvi}$ is to first predict $\hat{r}_{ui}$ and $\hat{r}_{vi}$ and subsequently take their difference. As a representative of this approach, we employ the *Probabilistic Matrix Factorization* or *PMF* [Mnih and Salakhutdinov 2007]. 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 priors on S_u and Q_i (with standard deviations $\varphi_{\mathcal{U}}$ and $\varphi_{\mathcal{I}}$) to control the complexity of the parameters, that is, $S_u \sim \mathcal{N}(\mathbf{0}, \varphi_U^2\mathbf{I})$ 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)):

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

Parameter estimation is by maximizing the log-posterior distribution over item and user vectors with hyperparameters, equivalent to minimizing the sum of squared-error 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 latter constitutes the regularization. The fitting constraint keeps the model parameters fit to the training data. The regularizers avoid overfitting, making the model generalize better [Hastie et al. 2011]. λ_U and λ_I are the regularization parameters:

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

The estimation is done using gradient descent [Mnih and Salakhutdinov 2007], with the following gradients. Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_u}^T Q_i - {S_v}^T Q_i$:

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

$$\frac{\partial E}{\partial Q_i} = -\big(r_{ui} - {S_u}^T Q_i\big)S_u + \lambda_I Q_i. \tag{18}$$

5.2. Factorizing Ratings to Fit Rating Differences

As discussed earlier, fitting ratings is an indirect way of predicting rating differences. Here we propose a new factorization model that meets our learning objective more directly, which we call *Differential Probabilistic Matrix Factorization* or *DPMF*. The plate diagram is shown in Figure 3(b). 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 (19)):

$$\hat{x}_{uvi} \sim \mathcal{N}\big({S_u}^T Q_i - {S_v}^T Q_i, \gamma^2\big). \tag{19}$$

The objective function of *DPMF* in Equation (20) shows that we fit the prediction $\hat{x}_{uvi} = {S_u}^T Q_i - {S_v}^T Q_i$ to the observation $x_{uvi} = r_{ui} - r_{vi}$:

$$E = \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U}, v\neq u}\sum_{i\in\mathcal{I}}\mathbb{I}_R(u,v,i)((r_{ui}-r_{vi})-({S_u}^T Q_i-{S_v}^T Q_i))^2+\frac{\lambda_U}{2}\sum_{u\in\mathcal{U}}||S_u||^2+\frac{\lambda_I}{2}\sum_{i\in\mathcal{I}}||Q_i||^2. \tag{20}$$

Estimation by gradient descent uses the following gradients:

$$\frac{\partial E}{\partial S_u} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))Q_i + \lambda_U S_u \tag{21}$$

$$\frac{\partial E}{\partial S_v} = ((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))Q_i + \lambda_U S_v \tag{22}$$

$$\frac{\partial E}{\partial Q_i} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))(S_u - S_v) + \lambda_I Q_i. \tag{23}$$

Note that both the approaches, *PMF* and *DPMF*, utilize the same number of parameters, but trained on different types of data, that is, r_{ui} and x_{uvi}, respectively, with different objective functions.

5.3. Factorizing Rating Differences

Instead of using predicted ratings as a means to estimate unseen triplets $\hat{x}_{uvi}$, another way is to directly predict $\hat{x}_{uvi}$. Here we investigate two different views in this direction. The first view represents the triplets as a two-dimensional matrix, with user pairs on one dimension and items on the other dimension. The triplets can then be predicted using matrix factorization, as described in Section 5.3.1. The second view represents the triplets as a three-dimensional $|\mathcal{U}| \times |\mathcal{U}| \times |\mathcal{I}|$ tensor and employs tensor factorization, as described in Section 5.3.2.

5.3.1. Pairwise PMF (PPMF). This approach is to fit another matrix $\mathbb{X}$, of size $|\mathcal{U}|^2 \times |\mathcal{I}|$. Each row corresponds to a pair of users u and v. 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 (24):

$$\hat{x}_{uvi} \sim \mathcal{N}({S_{uv}}^T Q_i, \gamma^2). \tag{24}$$

We call this approach *Pairwise PMF* or *PPMF*. The plate diagram is shown in Figure 3(c), illustrating 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 (25):

$$E = \frac{1}{2} \sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v} \sum_{i \in \mathcal{I}} \mathbb{I}_R(u, v, i)(x_{uvi} - {S_{uv}}^T Q_i)^2 + \frac{\lambda_U}{2} \sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v} ||S_{uv}||^2 + \frac{\lambda_I}{2} \sum_{i \in \mathcal{I}} ||Q_i||^2. \tag{25}$$

The estimation is done using gradient descent, with the following gradients. Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_{uv}}^T Q_i$:

$$\frac{\partial E}{\partial S_{uv}} = -(x_{uvi} - {S_{uv}}^T Q_i) Q_i + \lambda_U S_{uv} \tag{26}$$

$$\frac{\partial E}{\partial Q_i} = -(x_{uvi} - {S_{uv}}^T Q_i) S_{uv} + \lambda_I Q_i. \tag{27}$$

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. These are somewhat rectified by the tensor factorization approach discussed next.

5.3.2. Tensor Factorization. Rating differences $\hat{x}_{uvi}$ can be represented as triadic interactions between two users and one item. These triadic interactions can be encapsulated by a three-dimensional tensor. Let $\mathcal{X}$ be the three-dimensional $|\mathcal{U}| \times |\mathcal{U}| \times |\mathcal{I}|$ tensor. Each element $x_{uvi} \in \mathcal{X}$ is an instance of rating difference as defined in Equation (1). We still associate each user u with a latent vector $S_u \in \mathbb{R}^K$, and each item i with $Q_i \in \mathbb{R}^K$. Applying the basic *Tucker Decomposition* [Tucker 1966] would require the following factorization, where $C \in \mathbb{R}^{K \times K \times K}$ is the core tensor that reflects the interaction among different components:

$$\hat{x}_{uvi} = \sum_{x=1}^{K} \sum_{y=1}^{K} \sum_{z=1}^{K} C_{xyz} S_{ux} Q_{iy} S_{vz}. \tag{28}$$

There are two issues with this factorization. The first issue arises from the requirement in our scenario that $\hat{x}_{uvi} = -\hat{x}_{vui}$. This conflicts with the commutativity of Equation (28), because both S_u and S_v appear in the factorization of both $\hat{x}_{uvi}$ and $\hat{x}_{vui}$. This issue did not arise in the conventional application of Tucker Decomposition [Karatzoglou et al. 2010], because there the three dimensions are separate, whereas two of our three tensor dimensions represent users. To resolve this, we would seek to factorize only the magnitudes, which in this case will be the same, that is, $|\hat{x}_{uvi}| = |\hat{x}_{vui}|$. This is reasonable because the symmetric property of contextual probability $\mathrm{P}(y|x_{uvi})$ (see Section 4) implies that only the magnitude (and not the sign) of x_{uvi} affects the probability $\mathrm{P}(y|x_{uvi})$.

The second issue is that of computational efficiency due to the nested sum of degree 3 in Equation (28). This is a known issue in tensor factorization [Rendle and Schmidt-Thieme 2010]. We resolve this by adapting two existing simplifications of Tucker Decomposition, namely, Canonical Decomposition Tensor Factorization (CDTF) [Xiong et al. 2010] and Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme 2010]. In the following, we review these works and identify the required modification due to the first issue mentioned earlier.

Canonical Decomposition Tensor Factorization. CDTF is a special case of the Tucker Decomposition when the core tensor C is diagonal. This simplification allows us to collapse the nested sum from degree 3 to degree 1, which improves the computational complexity from $\mathcal{O}(K^3)$ to $\mathcal{O}(K)$. $\hat{x}_{uvi}$ is thus modeled as an inner product of S_u, S_v, and Q_i as shown in Equation (29):

$$\hat{x}_{uvi} \approx \langle S_u, S_v, Q_i \rangle = \sum_{k=1}^{K} S_{uk} S_{vk} Q_{ik}. \tag{29}$$

Similarly to *PMF*, we account for noise in x_{uvi} by introducing a Gaussian prior:

$$\hat{x}_{uvi} \sim \mathcal{N}(\langle S_u, S_v, Q_i \rangle, \gamma^2). \tag{30}$$

To learn the parameters, we formulate the following objective function. As mentioned earlier, to deal with the commutativity issue, we model the absolute value $|x_{uvi}|$:

$$E = \frac{1}{2} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}, v \neq u} \sum_{i \in \mathcal{I}} \mathbb{I}_R(u, v, i)(|x_{uvi}| - \langle S_u, S_v, Q_i \rangle)^2 + \frac{\lambda_U}{2} \sum_{u \in \mathcal{U}} ||S_u||^2 + \frac{\lambda_I}{2} \sum_{i \in \mathcal{I}} ||Q_i||^2. \tag{31}$$

Due to the symmetric property, each pair of users u and v is considered only once for each item i in Equation (31). Estimation by gradient descent uses the gradients that follow, where $\odot$ is the element-wise product between two vectors:

$$\frac{\partial E}{\partial S_u} = -(|x_{uvi}| - \langle S_u, S_v, Q_i \rangle)(Q_i \odot S_v) + \lambda_U S_u \tag{32}$$

$$\frac{\partial E}{\partial S_v} = -(|x_{uvi}| - \langle S_u, S_v, Q_i \rangle)(Q_i \odot S_u) + \lambda_U S_v \tag{33}$$

$$\frac{\partial E}{\partial Q_i} = -(|x_{uvi}| - \langle S_u, S_v, Q_i \rangle)(S_u \odot S_v) + \lambda_I Q_i. \tag{34}$$

Pairwise Interaction Tensor Factorization. Another simplification of Tucker Decomposition is PITF, which assumes that $\hat{x}_{uvi}$ is the sum of three pairwise products as follows, which still achieves a computational complexity of $\mathcal{O}(K)$ because it involves three summations of degree 1:

$$\hat{x}_{uvi} \approx S_u \odot Q_i + S_v \odot Q_i + S_u \odot S_v = \sum_{k=1}^{K} S_{uk} Q_{ik} + \sum_{k=1}^{K} S_{vk} Q_{ik} + \sum_{k=1}^{K} S_{uk} S_{vk}. \tag{35}$$

Similarly to the *CDTF*, we also introduce a probabilistic version by modeling the Gaussian prior as follows:

$$\hat{x}_{uvi} \sim \mathcal{N}(S_u \odot Q_i + S_v \odot Q_i + S_u \odot S_v, \gamma^2). \tag{36}$$

This results in the following objective function. Note that *PITF* also requires the use of absolute value $|x_{uvi}|$ to deal with the commutativity issue:

$$E = \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U},v\neq u}\sum_{i\in\mathcal{I}}\mathbb{I}_R(u,v,i)(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))^2 + \frac{\lambda_U}{2}\sum_{u\in\mathcal{U}}||S_u||^2 + \frac{\lambda_I}{2}\sum_{i\in\mathcal{I}}||Q_i||^2. \quad (37)$$

Estimation by gradient descent uses the following gradients:

$$\frac{\partial E}{\partial S_u} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(Q_i + S_v) + \lambda_U S_u \quad (38)$$

$$\frac{\partial E}{\partial S_v} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(Q_i + S_u) + \lambda_U S_v \quad (39)$$

$$\frac{\partial E}{\partial Q_i} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(S_u + S_v) + \lambda_I Q_i. \quad (40)$$

6. EXPERIMENTS

Our experimental objectives are threefold. The first two deal with the two components of our model. First, we investigate the effectiveness of parameter learning for *CAM*. Second, we study the effectiveness of different methods for rating difference predictions. Finally, we study the integrated model against comparable baselines on an evaluative prediction task. In addition to these, we include a case study to better illustrate the workings of our approach.

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 four real-life, publicly available rating datasets, namely, Ciao,[2] Epinions,[2] Flixster,[3] and Movielens100K.[4] Flixster and MovieLens100K contain ratings on movies. Ciao and Epinions contain ratings on various categories such as books, electronics, and movies. 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 preprocess the raw data as follows. First, we retain only pairs of users who have corated at least 20 items. We call such user pairs *neighbors*. This is to ensure that there is sufficient data to learn the model parameters reasonably accurately. For each corated 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 to 2009, and retain only user pairs that 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 II. After preprocessing, all the datasets are still sizeable, with thousands of users/items and tens to hundreds of thousands rating differences.

[2]http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm.
[3]http://www.cs.ubc.ca/~jamalim/datasets/.
[4]http://grouplens.org/datasets/movielens/.

Table II. Datasets

	Original			Preprocessed				
Dataset	Users	Items	Ratings	User	Items	Rating	User Pairs	Rating Differences
Ciao	1.1×10^4	1.1×10^5	3×10^5	3.9×10^2	$7,4 \times 10^3$	3.4×10^4	3.3×10^3	9.1×10^4
Epinions	1.2×10^5	3.3×10^5	1.1×10^6	1.1×10^3	2.4×10^5	1.3×10^5	1.0×10^4	3.6×10^5
Flixster	1.4×10^5	4.8×10^4	8.1×10^6	–	–	–	–	–
- Flixster06				1.8×10^2	3.4×10^3	6.7×10^4	1.6×10^3	3.0×10^5
- Flixster07				1.8×10^2	3.6×10^4	3.9×10^3	1.6×10^3	1.0×10^5
- Flixster08				1.8×10^2	3.0×10^4	2.1×10^4	1.6×10^3	6.5×10^4
- Flixster09				1.8×10^2	$2,1 \times 10^4$	1.4×10^4	1.6×10^3	4.8×10^4
MovieLens100K	10^3	1.7×10^3	10^5	9.0×10^2	1.5×10^3	9.9×10^4	1.2×10^5	6.0×10^6

Training Versus Testing. For each dataset, we create two sets of training/testing data.

The first corresponds to the evaluation of the model components in Sections 6.2 and 6.3, where we work with rating difference triplets x_{uvi}s. Since the *Local* model (Section 4) works on the user pair level, we apply fivefold cross-validation on each pair of (u, v)'s observed data with a ratio of 80/20 for the training/testing set to ensure the pairs have all rating difference instances in both the training/testing set. For *Global*, we simply combine the training/testing set of all pairs in the same fold to create a large training/testing sample, X_{train} and X_{test}. Each experiment is run five times on each of the five folds. The final result is reported as the average over the 25 runs for each setting.

The second corresponds to the evaluation of the integrated model for rating prediction in Section 6.4, where we work with user-item ratings r_{ui}s. To form the corresponding training set for ratings R_{train} for each X_{train}, we "decompose" each x_{uvi} into the original r_{ui} and r_{vi}. Similarly, R_{test} is created from X_{test}. To prevent duplicates and to maintain the ratio of training versus testing, if any rating r_{ui} appears in both training and testing sets, it will be allocated randomly to the training set with probability 0.8 and to the testing set with probability 0.2. Note that each rating exists only in the training set or testing set, but not both. Since there are five folds for X_{train} and X_{test}, correspondingly, there are five folds for R_{train} and R_{test}.

Compared to the earlier experiments in Do and Lauw [2014], this work has two major enhancements in the experimental setup. First, the training and testing sets are now stratified into five independent folds that support the traditional cross-validation approach. Second, the ratings that exist in both training and testing sets are now distributed between them to guarantee that each user/item has instances in the training and testing set. Previously, such ratings were allocated completely to the training set, and as a result there were too few testing instances with enough neighboring ratings for prediction. The current mode is fairer to all the methods (which have access to exactly the same information) and is more reflective of the utility of contextual agreement in rating prediction.

6.2. Contextual Agreement Model

First, we study the parameter learning for *CAM*. As mentioned in Section 4.1, the parameters for *CAM* can either be *Global* (same θ for all user pairs) or *Local* (specific θ_{uv} to each user pair). 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}). Equation (41) shows that it is measured similarly as in Blei et al. [2003], where $p(x_{uvi})$ is the likelihood of observing x_{uvi} as shown in Equation (7). Lower perplexity is

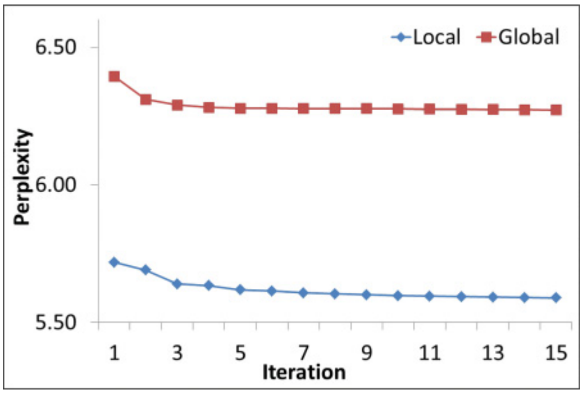


Fig. 4. Perplexity of *CAM* on Ciao testing set.

Table III. Perplexity of CAM on Testing Set (Statistically Significant Best-Performing Entries Have Asterisks)

	Ciao	Epinions	Flixster06	Flixster07	Flixster08	Flixster09	Movielens100K
Local	5.59*	5.12*	4.99*	4.86*	4.49*	4.45*	5.31*
Global	6.27	5.38	5.23	5.15	4.80	4.75	5.56

Table IV. Ratios of Number of Pairs in a Bin That *Local* Returns Higher Perplexity Than *Global*

No. of Observations	Ciao	Epinions	Flixster06	Flixster07	Flixster08	Flixster09	Movielens100K
≤ 50	78%	70%	71%	68%	70%	70%	71%
(50, 100]	82%	73%	72%	75%	77%	75%	76%
(100, 150]	88%	77%	79%	80%	80%	85%	78%
>150	93%	84%	86%	83%	84%	75%	88%

better, as it indicates a higher likelihood of observing the unseen data:

$$perplexity(X_{test}) = exp\left\{-\frac{\sum_{x_{uvi}\in X_{test}} \log p(x_{uvi})}{|X_{test}|}\right\}. \quad (41)$$

In Figure 4, we plot the log-likelihood achieved by *Global* versus *Local* over iterations on the Ciao dataset. It shows that convergence is attained relatively swiftly in just a few iterations. For this reason, the EM algorithms' stopping condition was set to 15 iterations. Similar trends are observed across all datasets. The perplexity after 15 iterations is shown in Table III. For all seven datasets, *Local* has lower (better) perplexity than *Global*. This result is expected as each user pair has a distinct behavior, and the global parameter will not fit all pairs equally well. Assigning each pair a unique set of parameters can capture their corating behavior better.

Although *Global* has worse perplexity in general, it may still have advantages over *Local* on some pairs with very few observations to learn from. In order to verify our hypothesis, we partition user pairs into bins according to the total number of observed rating differences. Each bin is represented as a range (a, b]. In each bin, we record the fraction of pairs for which *Local* has a better fit than *Global*. The same process is applied across the five folds of each dataset. Table IV reports the average percentages. The common observation across all datasets is that the fraction of pairs for which *Local* has a better fit is consistently decreasing with the number of observations. For Flixster09, the last bin has very few instances, which may explain why it is an exception to the general trend. *Global* indeed is more likely to perform better than *Local* on pairs with fewer observations than on pairs with many observations. However, since *Local*

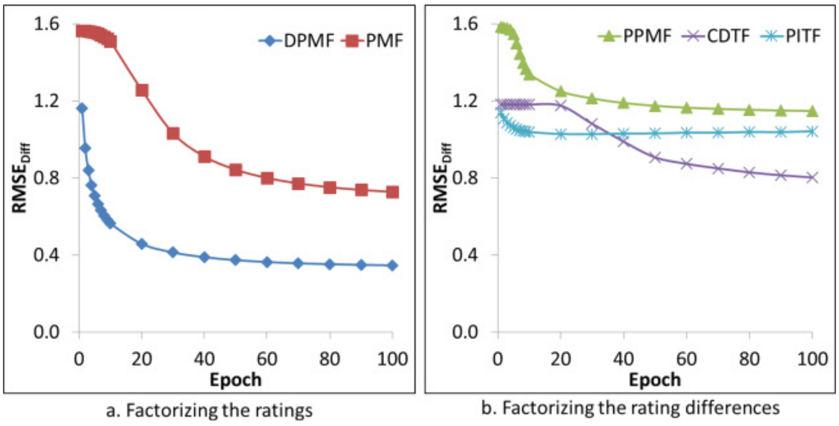


Fig. 5. Comparison of rating difference prediction methods ($RMSE_{diff}$) on Ciao dataset.

has better performance in general (i.e., all the fractions are greater than 50%), we will use *Local* for *CAM* in subsequent experiments.

The EM learning algorithms are relatively efficient. For *Global*, for each fold, convergence is achieved within 1 second for all datasets on an Intel Xeon Processor E5-2667 2.90GHz machine. For *Local*, for each fold, the parameters for all user pairs can be learned in 1 to 4 minutes.

6.3. Rating Difference Prediction

We study the efficacy of different factorization methods outlined in Section 5 in deriving good rating difference predictions. For each dataset, we use the same five folds as before. All except *PMF* are trained on X_{train}, while *PMF* is trained on the corresponding R_{train}. Parameter settings are adopted from the original paper for PMF [Mnih and Salakhutdinov 2007] (learning rate = 0.005, number of latent factors = 30, regularization coefficient = 0.002). All models 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 (42). We use absolute values because only magnitudinal error affects the contextual probability $\mathrm{P}(y|x_{uvi})$. A lower $RMSE_{diff}$ value indicates better performance:

$$RMSE_{diff} = \sum_{x_{uvi} \in X_{test}} \sqrt{\frac{(|\hat{x}_{uvi}| - |x_{uvi}|)^2}{|X_{test}|}}. \tag{42}$$

Vary Epochs. In Figure 5, we plot the $RMSE_{diff}$ across epochs on the Ciao dataset. One epoch corresponds to a full iteration over the whole training set. By 100 epochs, all the factorization methods have converged.

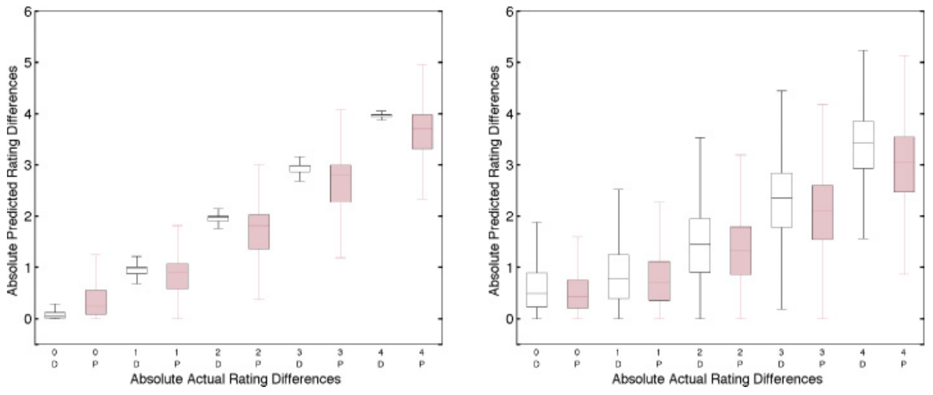
Comparing the two approaches that factorize ratings in Figure 5(a), we observe that *DPMF* converges faster than *PMF* and achieves a lower $RMSE_{diff}$ value. We attribute this to the approach of fitting the rating differences.

Comparing the three approaches that factorize rating differences in Figure 5(b), we observe that the tensor-based approaches *CDTF* and *PITF* perform better than the pairwise matrix factorization *PPMF*. We attribute this to the tensor factorization approach that ties together the latent vector for each user in different triplets.

Comparing all the approaches across all datasets in Table V, we observe that *DPMF* performs the best, followed by *PMF*. The tensor-based models *CDTF* and *PITF* have

Table V. Performance of Rating Difference Prediction Methods ($RMSE_{diff}$) for 100 Epochs and $k = 30$ (Statistically Significant Best-Performing Entries Have Asterisks)

	DPMF	PMF	PPMF	CDTF	PITF	Mean
Ciao	0.35*	0.73	1.15	0.8	1.04	1.18
Epinions	0.32*	0.62	0.91	0.82	0.82	0.89
Flixster06	0.47*	0.55	0.57	0.74	0.70	0.81
Flixster07	0.38*	0.45	0.68	0.73	0.71	0.79
Flixster08	0.33*	0.44	0.71	0.66	0.64	0.73
Flixster09	0.26*	0.40	0.72	0.65	0.61	0.72
MovieLens100K	0.70*	0.74	1.06	0.89	0.83	0.89



(a) Ciao (b) Movielens100K

Fig. 6. Comparison of predicted versus observed rating differences for DPMF (white) versus PMF (pink) on Ciao and Movielens100K.

middling performance, better than *PPMF* but worse than *DPMF* and *PMF*. They do not directly reflect the generation process of x_{uvi}. As defined in Equation (1), x_{uvi} is modeled as the difference between r_{ui} and r_{vi}. *CDTF* models x_{uvi} as inner products of three vectors (Equation (29)), which may find difficulties in capturing the correlations in rating differences. Although *PITF* explicitly addresses the underlying dyadic interactions (Equation (35)), its additive form does not reflect differences between two ratings.

We also introduce a baseline *Mean*, which simply averages the absolute values of all training instances as the prediction. Table V shows that all the proposed models outperform this simple baseline.

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

To better understand why *DPMF* has better performance than *PMF* in predicting rating differences, we conduct a deeper investigation of the two methods on two datasets: the *Ciao* and *Movielens*100*K* datasets. In Figure 6, the horizontal axes line up the observed rating differences, which range from 0 to 4. The maximum difference 4 is the difference between the lowest (1) and highest (5) ratings. The vertical axes show the range of predictions (median and interquartile) made by DPMF (white) and PMF (pink), respectively. First, we see that DPMF has much lower variances (narrow interquartile ranges) across all the bins, implying that its predictions are more precise. Second, the medians by DPMF are also closer to the actual observed rating differences than those by PMF. The degree of outperformance varies across datasets.

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

	Number of Latent Factors K				
Dataset	10	20	30	40	50
Ciao	0.85	0.43	0.35	0.34	0.34
Epinions	0.71	0.44	0.32	0.30	0.29
Flixster06	0.72	0.57	0.47	0.39	0.34
Flixster07	0.60	0.45	0.38	0.35	0.35
Flixster08	0.56	0.40	0.32	0.31	0.30
Flixster09	0.53	0.32	0.27	0.25	0.25
MovieLens100K	0.89	0.79	0.69	0.58	0.46

There is greater difference between DPMF's and PMF's performances on $Ciao$ than on $Movielens100K$.

One reason for DPMF's outperformance is given in the beginning of Section 5. DPMF has an error function that fits rating differences directly. In contrast, PMF seeks to fit ratings, which may not necessarily fit the rating differences as well, as the rating error may enlarge or narrow the rating differences. Another reason we posit here concerns the type of training instances. There are fewer rating instances than rating difference instances. Hence, it may cost PMF many more training epochs to learn effectively from the rating instances.

Vary Latent Factors. We conduct a separate experiment on *DPMF* for different numbers of latent factors K. The $RMSE_{diff}$s at 100 epochs are shown in Table VI. 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 methods, the parameters can be learned within 5 minutes for each fold on the same Intel Xeon Processor E5-2667 2.90GHz machine.

6.4. Application: Similarity-Based Neighborhood Collaborative Filtering

Here, we use the model parameters of *CAM* combined with the rating difference predictions by *DPMF* to generate contextual agreement probabilities $w_{uvi} = \mathrm{P}(y_{uvi}|\hat{x}_{uvi})$. These probabilities are used as similarity in neighborhood-based collaborative filtering, as outlined in Section 2. 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 (43):

$$RMSE_{rating} = \sum_{r_{ui} \in R_{test}} \sqrt{\frac{(\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}}. \tag{43}$$

Note that what is being evaluated here is the weights w_{uvi}s, and not the rating prediction method. Therefore, it is not our goal to compare to all rating prediction methods. Instead, the reasonable evaluation is to fix the prediction method to neighborhood-based collaborative filtering and vary the weights based on various baselines. What we consider baselines here are other approaches that also depend on similarity or agreement between a pair of users. We include the two most commonly used similarity measures, namely, cosine similarity and Pearson's correlation.

Contextual Versus Shared. We compare the efficacy of 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

Table VII. Versus Shared Preference ($RMSE_{rating}$, Statistically Significant Best-Performing Entries Have Asterisks)

		Shared Preference		
Dataset	CAM-DPMF	Uniform	Cosine	Pearson
Ciao	0.76*	1.15	1.14	1.14
Epinions	0.81*	1.06	1.06	1.06
Flixster06	0.95*	0.98	0.98	0.98
Flixster07	0.90*	0.98	0.95	0.95
Flixster08	0.90*	0.96	0.96	0.95
Flixster09	0.86*	0.93	0.92	0.91
MovieLens100K	0.83*	1.02	1.02	1.02

Table VIII. Versus Components ($RMSE_{rating}$, Statistically Significant Best-Performing Entries Have Asterisks)

		Components	
Dataset	CAM-DPMF	CAM-α	Linear-DPMF
Ciao	0.76*	1.12	0.82
Epinions	0.81*	1.04	0.86
Flixster06	0.95*	0.97	0.95
Flixster07	0.90*	0.94	0.90*
Flixster08	0.90*	0.93	0.91
Flixster09	0.86*	0.90	0.87*
MovieLens100K	0.83*	1.00	0.86

(see Section 3). We also include another baseline, called *Uniform*, which is the simple average of the ratings by neighbors, assuming all neighbors are considered to have the same similarity value. The prediction accuracies in terms of $RMSE_{rating}$ are listed in Table VII. For all of the datasets, *CAM-DPMF* has the lowest errors. 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.

***CAM-DPMF* Versus Components.** Since *CAM-DPMF* is a combination of *CAM* and *DPMF*, we now evaluate the efficacy of the individual components alone in the rating prediction task. *CAM-α* uses the α_{uv} of each pair as a shared similarity value. For the *DPMF* on its own, we linearly transform the predicted x_{uvi} into a similarity value as follows, where RD_{max} is the maximum possible value of rating differences in the training set. We call this approach as *Linear-DPMF*:

$$w_{uvi} = 1 - \frac{|\hat{x}_{uvi}|}{RD_{max}}. \tag{44}$$

Table VIII shows that the combined approach *CAM-DPMF* achieves the lowest error rates, which supports the necessity of integrating the two components to capture different aspects of the data. One interesting observation is that both *CAM-α* and *Linear-DPMF* also consistently perform better than conventional similarity measurements such as cosine or Pearson (statistically significant at 0.01). It shows the synergy when combining the ability of predicting unseen rating differences of *DPMF* with the ability of modeling user-pair agreement of *CAM*. Meanwhile, the relatively good performance of *DPMF*, though still worse than the combined *CAM-DPMF*, shows the important role of the former in contributing to the latter's performance.

Table IX. Ranking Application with Kendall's Tau (Statistically Significant Best-Performing Entries Have Asterisks)

		Shared Preference	
Dataset	*DPMF*	Cosine	Pearson
Ciao	0.122*	0.051	−0.028
Epinions	0.070*	0.059	0.028
Flixster06	0.223*	0.115	0.035
Flixster07	0.149*	0.039	0.021
Flixster08	0.222*	0.080	0.031
Flixster09	0.240*	0.117	0.018
MovieLens100K	0.119*	0.079	0.021

6.5. Application: Determining the More Similar Neighbor

As described in Section 2, in the second application, given two randomly selected neighbors of u, for example, v and z, we would like to determine who is more similar to u in his or her preference toward an item i. Because we observe the actual rating by v and z respectively on i (which are held out), we can determine the ground truth based on the observed $|x_{uvi}|$ and $|x_{uzi}|$ (smaller absolute difference is higher similarity).

This ranking-based application is different from the previous application in Section 6.4, which is based on weighted averages for rating prediction. For Section 6.4, the *CAM* component serves the purpose of "normalizing" the rating differences of different user pairs by transforming them into probabilities. For the ranking application in this section, that is unnecessary as we primarily care only about the direction of the comparison between two rating differences $|x_{uvi}|$ and $|x_{uzi}|$. This is the concern of the *DPMF* component, and therefore, here we rely on *DPMF* alone.

We create training and test sets for this application from the datasets earlier as follows. For every user u, we randomly select an item i among the set of items that u rated. We then select randomly two neighbors v and z with different real ratings to i. Let us denote their ratings r_{vi} and r_{zi}. Both ratings are held out from the training set. Therefore, the test set consists of several testing tuples in the form of $(u, i, v, z, r_{vi}, r_{zi})$. We sample at least 60 such folds using the procedure described.

For every testing tuple, v and z are ranked based on the similarity scores, w_{uvi} and w_{uzi}. In the case of cosine and Pearson's, they are the respective similarity values. In the case of *DPMF*, they are the predicted rating differences. The accuracy of ranking is measured by using the Kendall's Tau coefficient [Kendall 1938]. A pair (u, v) and (u, z) are said to be concordant if the ground truth $|x_{uvi}| < |x_{uzi}|$ and the similarity $w_{uv} > w_{uz}$ for cosine and Pearson, or $|\hat{x}_{uvi}| < |\hat{x}_{uzi}|$ for *DPMF*, are consistent (i.e., $|x_{uvi}| < |x_{uzi}|$ and $w_{uv} > w_{uz}$ or $|x_{uvi}| < |x_{uzi}|$ and $|\hat{x}_{uvi}| < |\hat{x}_{uzi}|$). Otherwise, that pair is discordant. Let us denote C and $\bar{C}$ as the number of concordant and discordant pairs. The overall Kendall's Tau coefficient is computed as $\tau = \frac{C-\bar{C}}{C+\bar{C}}$. The range of τ is therefore within $[-1, 1]$. A higher value of τ indicates better performance in ranking. A random ranking would result in $\tau = 0$.

Table IX shows the results of various methods across datasets. Each number in the table is an average over 10 different runs. Since *DPMF* is shown to be effective in predicting rating differences in Section 6.3, it is reasonable for *DPMF* to have the highest τ coefficient. The numbers also show that cosine and Pearson have lower τ coefficients, and their lower performance is due to using the same similarity across all items, rather than specific to each item.

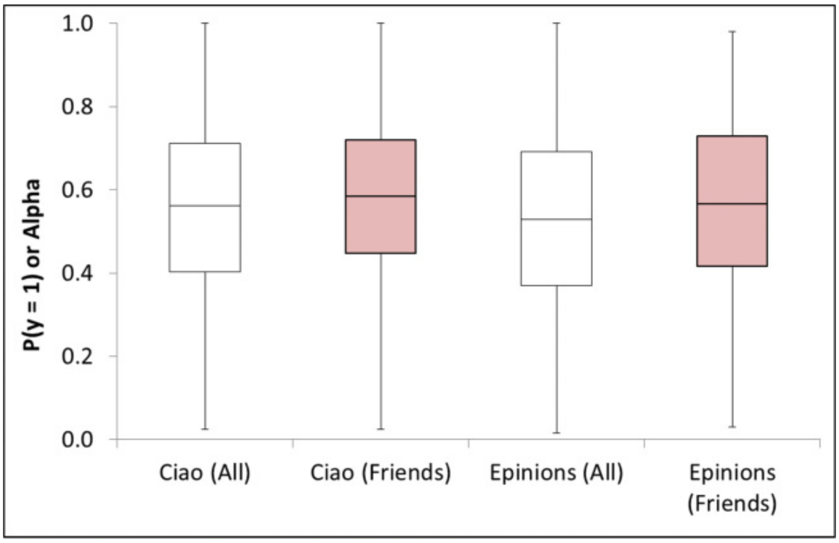


Fig. 7. Distributions of P($y_{uvi} = 1$) or α_{uv} with "friendship" factor.

6.6. Analysis of Prevalence of Agreement

In this section, we analyze the prior probability of agreement α_{uv}s (for different user pairs) for various datasets. This parameter is the prior probability of agreement P($y_{uvi} = 1$) for a pair of users u and v. In particular, we are interested in how these probabilities vary across user pairs, depending on varying attributes, such as friendship, demographics, and time.

Friendship. First, we test the frequently made hypothesis that a friendship or trust relationship can help in learning the preferences of users [Ma et al. 2009, 2011]. This analysis could only be performed on the *Ciao* and *Epinions* datasets. *MovieLens*100*K* does not have social network information. *Flixster* after filtering does not contain a sufficient number of social links for statistical tests.

In Figure 7, we draw the distributions of α_{uv}, for two populations. The first, drawn in white, concerns all pairs of users. The second, drawn in red, narrows down the population to only those user pairs sharing friendship or a trustor-trustee relationship. One observation is that friendship does contain some information. The comparison of every pair of white (all pairs) versus red (friends-only) box plots shows that friends have greater agreement (statistically significant) in general. 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 does not mean they always agree. Therefore, it is helpful to know the context of agreement.

Demographics. Since *MovieLens*100 contains demographic information, we study whether age and gender have an effect on the probability of contextual agreement. We split the users into three different age groups: Youth (up to 25), Adult (from 26 to 49), and Senior (above 50). Within each age group, we compare whether user pairs of the same gender (both males or both females) would have greater similarity than different genders. Figure 8 shows that within the age groups "Young" and "Adult," same-gender user pairs have higher agreement preferences (statistically significant) than pairs of different genders. The effects of gender on the "Senior" age group is much weaker. Overall, demographics do not seem to have as much of an effect as friendship does.

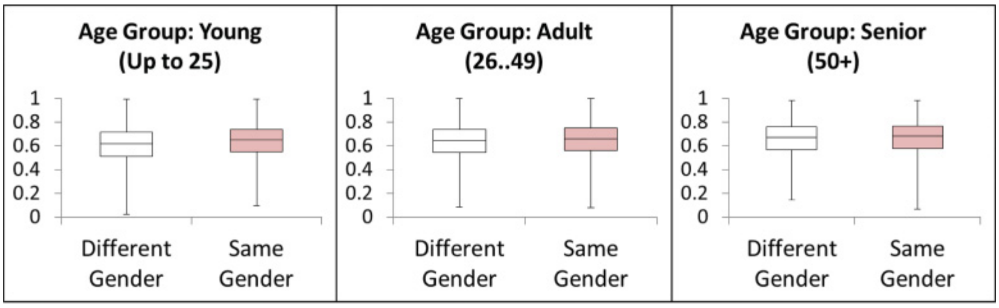


Fig. 8. Distribution of α_{uv} across different age groups and genders on MovieLens100K.

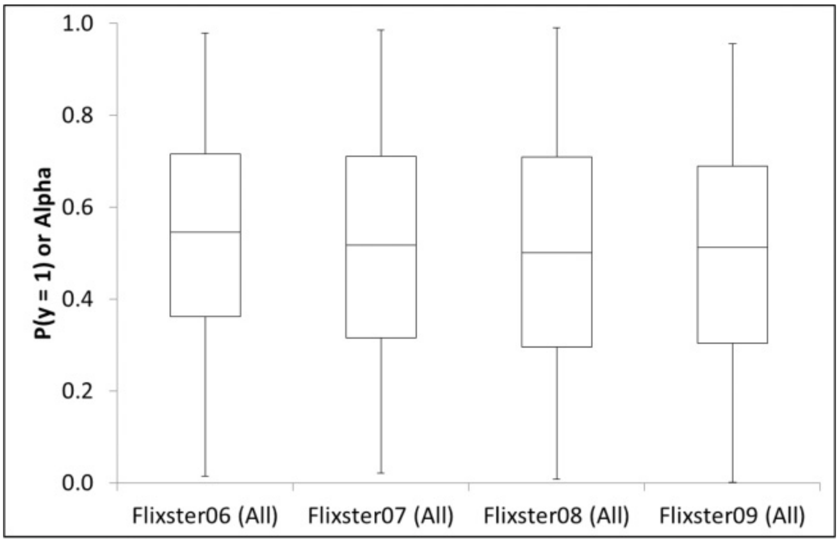


Fig. 9. Distributions of $P(y_{uvi} = 1)$ or α_{uv} with "time" factor.

Time. Since Flixster can be split into four datasets for different years, 2006, 2007, 2008, and 2009, we are also interested in the variation across time. In Figure 9, we plot the distribution of α_{uv} for each Flixster subset containing data belonging to a specific year. The plots show that user agreement generally remains stable across the years. We perform paired-sample t-tests on user pairs for two consecutive years each time. The variance across years is statistically insignificant during the three years of 2007, 2008, and 2009, except for Flixster06, which has somewhat higher agreement than Flixster07. Since Flixster is about movies, we hypothesize that the heterogeneity of movie themes across years may account partially for the variance in agreement.

6.7. Case Study

To illustrate the workings of *CAM*, we now show a case study drawn from the MovieLens100K dataset, involving the same pair of users as in Section 1. Table X shows the ratings of user u (u_{38}) and v (u_{197}) on 20 movies. Based on these ratings, the *CAM* parameters for this pair are as follows: $\alpha = 0.28$, $\mu_0 = 2.09$, $\sigma_0 = 1.36$, $\sigma_1 = 1.48$. The relatively low α suggests that this pair does not always agree. That $\mu_0 = 2.09$ suggests that when they disagree, their rating difference is around 2. This is evident from the fourth column labeled $|x_{uvi}|$, which tracks their rating differences. *CAM* uses

Table X. MovieLens Case Study

Movie	r_{ui}	r_{vi}	$\lvert x_{uvi}\rvert$	CAM	Pearson	Cosine
Conan the Barbarian	5	1	4	0.05		
Volcano	5	2	3	0.16		
First Knight	5	2	3	0.16		
Scream	5	3	2	0.41		
G. I. Jane	5	3	2	0.41		
George of the Jungle	3	1	2	0.41		
Titanic	5	4	1	0.80		
Liar Liar	5	4	1	0.80		
Top Gun	5	4	1	0.80		
Braveheart	5	5	0	1.00	−0.26	0.81
Jurassic Park	5	5	0	1.00		
Conspiracy Theory	4	4	0	1.00		
Die Hard (1995)	2	3	1	0.80		
Full Metal Jacket	2	3	1	0.80		
The Fugitive	3	5	2	0.41		
Batman (1989)	1	3	2	0.41		
The Godfather	2	5	3	0.16		
Die Hard 2 (1990)	1	4	3	0.16		
Ben Hur	1	5	4	0.05		
The Terminator	1	5	4	0.05		

these parameters to estimate the contextual probability of agreement shown in the fifth column. As expected, the probability of contextual agreement is high (close to 1) for the movies in the shaded middle of the table (where rating differences are low) and is low (close to 0) for the other movies. 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. However, we note that this case study shows a single case and is meant to be illustrative, not comparative. The comparative analyses across many user pairs in aggregate were conducted in Sections 6.4 and 6.5.

7. CONCLUSION

In this article, we address the novel problem of estimating the contextual agreement between two users in the context of one item. We formulate this problem into probabilistic modeling with two 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. The second component extends the use of *CAM* to unseen triplets by predicting rating differences between two users on the same item. We systematically outline five approaches based on matrix factorization and tensor decomposition, including a new proposed model called *DPMF*. Through experiments on real-life rating datasets, we describe how well the proposed algorithms can learn the model and show how the estimated contextual agreement probabilities can be useful in improving upon the rating predictions in neighborhood-based collaborative filtering as compared to approaches based on shared preferences.

APPENDIX

This appendix includes an expanded derivation of the parameter constraint to ensure the monotonicity property (Section 4.2) and the expanded derivation of the E-step and M-step of the EM algorithm to estimate the model parameters (Section 4.3).

A.1. Parameter Constraint for the Monotonicity Property

As discussed in Section 4.2, to enforce the monotonicity property, we enforce the following inequality (previously shown in Equation (9)):

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

We denote the numerator and denominator of the previous equation as $A(x)$ and $B(x)$, that is, $\mathcal{G}(x) = A(x) \div B(x)$. $A(x)$ is shown in Equation (45). $B(x)$ is shown in Equation (46):

$$\begin{aligned} A(x) &= \alpha \mathcal{N}(x; 0, \sigma_1) \\ &= \alpha \frac{1}{\sigma_1} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\} \end{aligned} \tag{45}$$

$$\begin{aligned} B(x) &= \alpha \mathcal{N}(x; 0, \sigma_1) + 0.5(1-\alpha)(\mathcal{N}(x; \mu_0, \sigma_0) + \mathcal{N}(x; -\mu_0, \sigma_0)) \\ &= \alpha \frac{1}{\sigma_1} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\} + 0.5(1-\alpha)\frac{1}{\sigma_0}\frac{1}{\sqrt{2\pi}}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\} + \exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\right). \end{aligned} \tag{46}$$

The derivative of $\mathcal{G}(x)$ thus can be expressed in terms of $A(x)$ and $B(x)$:

$$\frac{\partial \mathcal{G}(x)}{\partial x} = \frac{A'(x)B(x) - B'(x)A(x)}{(B(x))^2} < 0. \tag{47}$$

The first derivative of $A(x)$ w.r.t. x, or $A'(x)$, is as follows:

$$A'(x) = \alpha \frac{-x}{\sigma_1^3} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}. \tag{48}$$

The first derivative of $B(x)$ w.r.t. x, or $B'(x)$, is as follows:

$$\begin{aligned} B'(x) = {} & \alpha \frac{-x}{\sigma_1^3} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\} \\ & - 0.5(1-\alpha)\frac{1}{\sigma_0^3}\frac{1}{\sqrt{2\pi}}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}(x-\mu_0) + \exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}(x+\mu_0)\right). \end{aligned} \tag{49}$$

For Equation (47) to hold, as $(B(x))^2$ is positive, the condition in Equation (50) holds:

$$A'(x)B(x) - B'(x)A(x) < 0 \Leftrightarrow A'(x)B(x) < B'(x)A(x). \tag{50}$$

Substituting $A'(x)$ and $B'(x)$ into Equation (50), we have the following inequality:

$$\begin{aligned}
&-\alpha\frac{x}{\sigma_1^3}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}\Bigg(\alpha\frac{1}{\sigma_1}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}\\
&\qquad+0.5(1-\alpha)\frac{1}{\sigma_0}\frac{1}{\sqrt{2\pi}}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\right)\Bigg)\\
&<\alpha\frac{1}{\sigma_1}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}\Bigg(\alpha\frac{-x}{\sigma_1^3}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}\\
&\qquad-0.5(1-\alpha)\frac{1}{\sigma_0^3}\frac{1}{\sqrt{2\pi}}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}(x-\mu_0)+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}(x+\mu_0)\right)\Bigg).
\end{aligned}\tag{51}$$

Cancelling α, $\frac{1}{\sigma_1}$, $\frac{1}{2\pi}$, and $\exp\{-\frac{x^2}{2\sigma_1^2}\}$ from both sides of Equation (51), we have

$$\begin{aligned}
&\frac{x}{\sigma_1^2}\left(\alpha\frac{1}{\sigma_1}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}+0.5(1-\alpha)\frac{1}{\sigma_0}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\right)\right)\\
&>\alpha\frac{x}{\sigma_1^3}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}+0.5(1-\alpha)\frac{1}{\sigma_0^3}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}(x-\mu_0)+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}(x+\mu_0)\right).
\end{aligned}\tag{52}$$

Subtracting the term $\alpha\frac{x}{\sigma_1^3}\exp\{-\frac{x^2}{2\sigma_1^2}\}$ from both sides of Equation (52), we get

$$\begin{aligned}
&0.5(1-\alpha)\frac{1}{\sigma_0}\frac{x}{\sigma_1^2}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\right)\\
&\qquad>0.5(1-\alpha)\frac{1}{\sigma_0^3}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}(x-\mu_0)+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}(x+\mu_0)\right).
\end{aligned}\tag{53}$$

Cancelling 0.5, $1-\alpha$, and $\frac{1}{\sigma_0}$ from both sides of Equation (53), we get

$$\begin{aligned}
&\frac{x}{\sigma_1^2}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\right)\\
&\qquad>\frac{1}{\sigma_0^2}\left(\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}(x-\mu_0)+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}(x+\mu_0)\right).
\end{aligned}\tag{54}$$

Reorganizing Equation (54) by moving all terms to one side, we achieve the final form equivalent to Equation (10):

$$\begin{aligned}
&\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2}-\frac{x-\mu_0}{\sigma_0^2}\right)+\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2}-\frac{x+\mu_0}{\sigma_0^2}\right)>0\\
&\Leftrightarrow\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}+\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_0^2}-\frac{x-\mu_0}{\sigma_0^2}\right)+\left(\frac{x}{\sigma_1^2}-\frac{x+\mu_0}{\sigma_0^2}\right)>0\\
&\Leftrightarrow\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2}-\frac{x-\mu_0}{\sigma_0^2}\right)+\left(\frac{x}{\sigma_1^2}-\frac{x+\mu_0}{\sigma_0^2}\right)>0.
\end{aligned}\tag{55}$$

A.2. Parameter Estimation Using Expectation Maximization

Given the model described in Section 4.3, suppose we have $|X|$ number of observations, where $X = \{x\}$; the mixture of Gaussians can be defined for a data point x as in Equation (56), for some model parameter $\theta = \langle \alpha, \mu_1 = 0, \sigma_1, \mu_0, \sigma_0 \rangle$:

$$\begin{aligned}\mathrm{P}(x|\theta) &= \alpha \mathrm{P}(x|y=1) + (1-\alpha)\mathrm{P}(x|y=0)\\ &= \alpha \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\} + (1-\alpha)\frac{1}{2\sigma_0\sqrt{2\pi}}\left(\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\} + \exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}\right).\end{aligned} \tag{56}$$

We seek the parameters that maximize the following log-likelihood (previously shown in Equation (14)), where $\mathrm{P}(X|\theta) = \prod_{x\in X}\mathrm{P}(x|\theta)$. Here, $\alpha_1 = \alpha$, and $\alpha_0 = 1-\alpha$:

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

To outline the E-step and M-step, we first find the derivatives of $\mathcal{L}$, with respect to each individual parameter. These derivations are shown in the following sections.

A.2.1. Derivation of $\mathcal{L}$ w.r.t to μ_0. Equation (57) shows the differentiation of Equation (14) with respect to μ_0:

$$\begin{aligned}\frac{\partial}{\partial \mu_0}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial \mu_0} ln \mathrm{P}(x|\theta)\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial\mu_0}\mathrm{P}(x|\theta)\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}(1-\alpha)\frac{1}{2\sigma_0\sqrt{2\pi}}\left(\exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\frac{-(x+\mu_0)}{\sigma_0^2} + \exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}\frac{(x-\mu_0)}{\sigma_0^2}\right)\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2\sigma_0^2}(x(\mathcal{N}(x|-\mu_0,\sigma_0^2) - \mathcal{N}(x|\mu_0,\sigma_0^2)) - \mu_0(\mathcal{N}(x|-\mu_0,\sigma_0^2) + \mathcal{N}(x|\mu_0,\sigma_0^2))).\end{aligned} \tag{57}$$

In the **E-step**, we compute $C = \sum_{x\in X} c(x)$, where $c(x)$ is defined in Equation (58):

$$c(x) = \frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2}(\mathcal{N}(x|-\mu_0,\sigma_0^2) + \mathcal{N}(x|\mu_0,\sigma_0^2)). \tag{58}$$

Set Equation (57) to zero to solve for μ_0. In the **M-step**, we use C to update μ_0, as shown in Equation (59):

$$\begin{aligned}\sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2}(\mathcal{N}(x|-\mu_0,\sigma_0^2) + \mathcal{N}(x|\mu_0,\sigma_0^2)\mu_0 &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2}(\mathcal{N}(x|-\mu_0,\sigma_0^2) - \mathcal{N}(x|\mu_0,\sigma_0^2))x\\ C\mu_0 &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2}(\mathcal{N}(x|-\mu_0,\sigma_0^2) - \mathcal{N}(x|\mu_0,\sigma_0^2))x\\ \mu_0 &= \frac{1}{C}\sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{1-\alpha}{2}(\mathcal{N}(x|-\mu_0,\sigma_0^2) - \mathcal{N}(x|\mu_0,\sigma_0^2))x.\end{aligned} \tag{59}$$

Since it is hard to solve Equation (59) analytically in closed form due to the existence of μ_0 in both exponential form and the denominator, we refer to the EM-style iterative algorithm as described in Bishop and Nasrabadi [2006] to find the local optimal solution for μ_0. We start with a random initialization of the μ_0, then repeatedly update the parameter using its value in the previous iteration. To be exact, μ_0's values on the right-hand side of Equation (59) is the value computed from the previous iteration. Later, we shall see that this strategy also applies to learn other parameters.

A.2.2. Derivation of $\mathcal{L}$ w.r.t to α. Denote $\alpha_1 = \alpha$ and $\alpha_0 = 1 - \alpha$. The likelihood function for one data point in Equation (56) is transformed to Equation (60):

$$\mathrm{P}(x|\theta) = \alpha_1 \mathrm{P}(x|y=1) + \alpha_0 \mathrm{P}(x|y=0). \tag{60}$$

Equate the derivatives of Equation (14) with respect to α_1, α_0, and λ_α to zero, and we have following conditions:

—*Condition 1.1*: Set derivative of Equation (14) w.r.t. α_1 to zero:

$$\begin{aligned}\frac{\partial}{\partial \alpha_1}\mathcal{L} &= \sum_{x\in X} \frac{\partial}{\partial \alpha_1} \ln \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{\partial}{\partial \alpha_1} \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X} \frac{1}{\mathrm{P}(x|\theta)} \mathrm{P}(x|y=1) + \lambda_\alpha = 0.\end{aligned} \tag{61}$$

—*Condition 1.2*: Set derivative of Equation (14) w.r.t. α_0 to zero:

$$\begin{aligned}\frac{\partial}{\partial \alpha_0}\mathcal{L} &= \sum_{x\in X} \frac{\partial}{\partial \alpha_0} \ln \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{\partial}{\partial \alpha_0} \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X} \frac{1}{\mathrm{P}(x|\theta)} \mathrm{P}(x|y=0) + \lambda_\alpha = 0.\end{aligned} \tag{62}$$

—*Condition 1.3*: Set derivative of Equation (14) w.r.t. λ_α to zero:

$$\frac{\partial}{\partial \lambda_\alpha}\mathcal{L} = \alpha_1 + \alpha_0 - 1 = 0. \tag{63}$$

—*Condition 1.4*: Condition of Lagrange multiplier:

$$\lambda_\alpha \geq 0. \tag{64}$$

Multiplying α_1 into both sides of Equation (61) and α_0 into both sides of Equation (62) and summing them together, with the condition in Equation (63), we have $\lambda_\alpha = -|X|$.

In the **E-step**, we compute $d(x) = \frac{\alpha_1}{\mathrm{P}(x|\theta)} \mathrm{P}(x|y=1)$. In the **M-step**, we compute α_1 according to Equation (65):

$$\alpha_1 = \frac{1}{|X|} \sum_{x\in X} d(x). \tag{65}$$

Similarly to the learning of μ_0 earlier, we also use the iterative algorithm to learn α_1. Specifically, α_1's value in $d(x)$ is taken from a previous iteration. At the beginning, the value is randomly initialized.

A.2.3. Derivation of $\mathcal{L}$ w.r.t σ_1 and σ_0. Equate derivatives of Equation (14) with respect to σ_1, σ_0, s^2, and λ_σ to zero; we have following conditions:

—*Condition 2.1*: Set the derivative of Equation (14) with respect to the slack variable s to zero:

$$\frac{\partial}{\partial s}\mathcal{L} = -2\lambda_\sigma s = 0. \tag{66}$$

—*Condition 2.2:* Set derivative of Equation (14) with respect to the multiplier λ_σ to zero:

$$\begin{aligned}\frac{\partial}{\partial \lambda_\sigma}\mathcal{L} &= \sigma_0 - \sigma_1 - s^2 = 0\\ \sigma_0 &= \sigma_1 + s^2.\end{aligned} \tag{67}$$

—*Condition 2.3*: Set derivative of Equation (14) with respect to σ_1 to zero:

$$\begin{aligned}\frac{\partial}{\partial \sigma_1}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial \sigma_1}\ln \mathrm{P}(x|\theta) - \lambda_\sigma\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial \sigma_1}\mathrm{P}(x|\theta) - \lambda_\sigma\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\alpha\frac{\partial}{\partial \sigma_1}\left(\frac{1}{\sigma_1\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}\right) - \lambda_\sigma\\ &= \sum_{x\in X}\frac{\alpha \mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)}\left(1-\frac{x^2}{\sigma_1^2}\right) - \lambda_\sigma = 0.\end{aligned} \tag{68}$$

—*Condition 2.4*: Set derivative of Equation (14) with respect to σ_0 to zero:

$$\begin{aligned}\frac{\partial}{\partial \sigma_0}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial \sigma_0} ln\mathrm{P}(x|\theta) + \lambda_\sigma\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial \sigma_0}\mathrm{P}(x|\theta) + \lambda_\sigma\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}(1-\alpha)\frac{\partial}{\partial \sigma_0}\frac{1}{2\sigma_0\sqrt{2\pi}}\left(\exp\left\{-\frac{(x+\mu)^2}{2\sigma_0^2}\right\} + \exp\left\{-\frac{(x-\mu)^2}{2\sigma_0^2}\right\}\right) + \lambda_\sigma\\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}(1-\alpha)\left(-\frac{1}{\sigma_0}\mathrm{P}(x|y=0) + \frac{1}{2\sigma_0^3}\left(\mathcal{N}(x|-\mu,\sigma_0^2)(x+\mu)^2 + \mathcal{N}(x|\mu,\sigma_0^2)(x-\mu)^2\right)\right) + \lambda_\sigma = 0.\end{aligned} \tag{69}$$

Denote $D = \sum_{x\in X} d(x)$ as the quantity computed previously to obtain α_1. Set Equation (68) to zero to compute σ_1 in the **M-step**, and we have Equation (70):

$$\begin{aligned}\sum_{x\in X}\frac{\alpha \mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)} &= \sum_{x\in X}\frac{\alpha \mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)}\frac{x^2}{\sigma_1^2}\\ \sigma_1^2 &= \frac{1}{D}\sum_{x\in X} d(x)\cdot x^2.\end{aligned} \tag{70}$$

Denote E as the quantity computed in the E-step to compute σ_0:

$$E = \sum_{x\in X}(e_1(x) + e_2(x)) \tag{71}$$

$$e_1(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)}\mathcal{N}\big(x| - \mu_0, \sigma_0^2\big) \tag{72}$$

$$e_2(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)}\mathcal{N}\big(x|\mu_0, \sigma_0^2\big). \tag{73}$$

Set the Equation (69) to zero, and we have Equation (74):

$$\begin{aligned}\sum_{x\in X}\frac{(1-\alpha)\mathrm{P}(x|y=0)}{\mathrm{P}(x|\theta)} &= \frac{1}{2\sigma_0^2}\sum_{x\in X}\frac{(1-\alpha)}{\mathrm{P}(x|\theta)}(\mathcal{N}(x|-\mu_0,\sigma_0^2)(x+\mu_0)^2 + \mathcal{N}(x|\mu_0,\sigma_0^2)(x-\mu_0)^2)\\ \sigma_0^2 &= \frac{1}{2E}\sum_{x\in X}\frac{(1-\alpha)}{\mathrm{P}(x|\theta)}(\mathcal{N}(x|-\mu_0,\sigma_0^2)(x+\mu_0)^2 + \mathcal{N}(x|\mu_0,\sigma_0^2)(x-\mu_0)^2)\\ \sigma_0^2 &= \frac{1}{E}\sum_{x\in X}(e_1(x)\cdot(x+\mu_0)^2 + e_2(x)\cdot(x-\mu_0)^2)).\end{aligned} \tag{74}$$

Substitute Equation (67) into the right-hand side of Equation (74) to compute s^2 using the old value of σ_1, and then update the new value of σ_0 with the newly computed s using Equation (75) in the **M-step**:

$$\sigma_0 = s + \sigma_1. \tag{75}$$

The series of **E-step** and **M-step** derived in this appendix are summarized at the end of Section 4.3.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender Systems Handbook*. Springer, 217–253.

Amr Ahmed, Bhargav Kanagal, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeff Yuan. 2013. Latent factor models with additive and hierarchically-smoothed user preferences. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'13)*. 385–394.

Rushi Bhatt, Vineet Chaoji, and Rajesh Parekh. 2010. Predicting product adoption in large-scale social networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'10)*. ACM, 1039–1048.

Christopher M. Bishop and Nasser M. Nasrabadi. 2006. *Pattern Recognition and Machine Learning*. Springer.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

Stephen Poythress Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'98)*. 43–52.

Loc Do and Hady W. Lauw. 2014. Modeling contextual agreement in preferences. In *Proceedings of the ACM International Conference on World Wide Web (WWW'14)*. 315–326.

Hui Fang, Yang Baoy, and Jie Zhang. 2013. Misleading opinions provided by advisors: Dishonesty or subjectivity. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'13)*. 1983–1989.

Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. 2011. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Thomas Hofmann. 2003. Collaborative filtering via Gaussian probabilistic latent semantic analysis. In *Proceedings of the International ACM SIGIR Conference*. 259–266.

Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems* 22, 1 (2004), 89–115.

Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender Systems: An Introduction*. Cambridge University Press.

Rong Jin, Joyce Y. Chai, and Luo Si. 2004. An automatic weighting scheme for collaborative filtering. In *Proceedings of the International ACM SIGIR Conference*. 337–344.

Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Jeff Yuan, and Lluis Garcia-Pueyo. 2012. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proceedings of the VLDB Endowment (PVLDB)* 5, 10 (2012), 956–967.

Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'10)*. 79–86.

Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika* (1938), 81–93.

Noam Koenigstein, Gideon Dror, and Yehuda Koren. 2011. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'11)*. 165–172.

Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Communications of the ACM* 53, 4 (2010), 89–97.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 42–49.

Neil D. Lawrence and Raquel Urtasun. 2009. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML'09)*. 601–608.

Daniel D. Lee and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.

Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. 2012. Lars: A location-aware recommender system. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'12)*. 450–461.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. Springer, 73–105.

Hao Ma, Irwin King, and Michael R. Lyu. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the International ACM SIGIR Conference*. 203–210.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'11)*. 287–296.

Lester W. Mackey, David Weiss, and Michael I. Jordan. 2010. Mixed membership matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML'10)*. 711–718.

Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'11)*. 141–149.

Rokia Missaoui, Petko Valtchev, Chabane Djeraba, and Mehdi Adda. 2007. Toward recommendation based on ontology-powered web-usage mining. *IEEE Internet Computing* 11, 4 (2007), 45–52.

Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS'07)*. 1257–1264.

Michael J. Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The Adaptive Web*. Springer, 325–341.

Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'10)*. 81–90.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW'94)*. 175–186.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning (ICML'08)*. 880–887.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the ACM International Conference on World Wide Web (WWW'01)*. 285–295.

Hanhuai Shan, Jens Kattge, Peter B. Reich, Arindam Banerjee, Franziska Schrodt, and Markus Reichstein. 2012. Gap filling in the plant kingdom trait prediction using hierarchical probabilistic matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML'12)*. 1303–1310.

Nathan Srebro, Jason Rennie, and Tommi S. Jaakkola. 2004. Maximum-margin matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS'04)*. 1329–1336.

Ledyard R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM Conference on Data Mining (SDM'10)*, Vol. 10. SIAM, 211–222.

Received January 2015; revised September 2015; accepted December 2015