Context-Adaptive Matrix Factorization for Multi-Context Recommendation

Tong Man[1], Huawei Shen[1], Junming Huang[2*], Xueqi Cheng[1]
[1]CAS Key Lab of Network Data Science and Technology
Institute of Computing Technology, Chinese Academy of Sciences
mantong@software.ict.ac.cn {shenhuawei,cxq}@ict.ac.cn
[2]CompleX Lab, Web Sciences Center
University of Electronic Science and Technology of China, Chengdu 611731, China
mail@junminghuang.com

ABSTRACT

Data sparsity is a long-standing challenge for recommender systems based on collaborative filtering. A promising solution for this problem is multi-context recommendation, i.e., leveraging users' explicit or implicit feedback from multiple contexts. In multi-context recommendation, various types of interactions between entities (users and items) are combined to alleviate data sparsity of a single context in a collective manner. Two issues are crucial for multi-context recommendation: (1) How to differentiate context-specific factors from entity-intrinsic factors shared across contexts? (2) How to capture the salient phenomenon that some entities are insensitive to contexts while others are remarkably context-dependent? Previous methods either do not consider context-specific factors, or assume that a context imposes equal influence on different entities, limiting their capability of combating data sparsity problem by taking full advantage of multiple contexts.

In this paper, we propose a context-adaptive matrix factorization method for multi-context recommendation by simultaneously modeling context-specific factors and entity-intrinsic factors in a unified model. We learn an entity-intrinsic latent factor for every entity, and a context-specific latent factor for every entity in each context. Meanwhile, using a context-entity mixture parameter matrix we explicitly model the extent to which each context imposes influence on each entity. Experiments on two real scenarios demonstrate that our method consistently outperforms previous multi-context recommendation methods on all different sparsity levels. Such a consistent performance promotion forms the unique superiority of our method, enabling it to be a reliable model for multi-context recommendation.

Categories and Subject Descriptors

H.2.8.d [**Information Technology and Systems**]: Database Applications - Data Mining

*This work is done when Junming Huang as a PhD candidate in Institute of Computing Technology, Chinese Academy of Sciences.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CIKM 2015, October 19-23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806503.

General Terms

Algorithms, Measurement, Experimentation

Keywords

Recommender System, Collaborative Filtering, Multi-context Recommendation, Data Sparsity

1. INTRODUCTION

With the increasingly growing amount of information available online, recommender system becomes a necessary tool to help users efficiently find their desired items, e.g., movies [1], music [2], news [3], books [4], online friends [5] and travel packages [6]. Personalized recommender systems gain great success in industry by collecting and analyzing users' explicit or implicit feedback, generally represented as interactions between *users* and *items*. For example, users offer explicit scores to movies they watched or books they read; users listen to their favorite songs multiple times; users frequently browse the products they potentially want to buy. Based on these observed interactions, collaborative filtering is widely used to infer users' preference and to predict items they are likely to interact with. Existing collaborative filtering methods could be classified into neighborhood-based methods [7] and model-based methods [8, 9]. Among those methods, matrix factorization [1] is the mainstream technique in personalized recommender system.

Matrix factorization (MF) formalizes collaborative filtering into a matrix completion problem, recovering an incomplete user-item preference matrix according to observed interactions or ratings between users and items [1]. MF factorizes the preference matrix into two low-rank matrices that represent latent factors for users and items respectively. With the two latent factor matrices, recommendation problem is addressed simply by their inner product. Recommendation accuracy of matrix factorization heavily depends on observed preference matrix. Unfortunately, preference matrix is often highly *sparse* in practical scenarios, forming a hard barrier for most real-world recommender systems [10]. Moreover, preference matrix is *unevenly distributed*, i.e., a majority of inactive users express preference on a small number of items and a majority of unpopular items get few feedbacks. Such a data sparsity problem is particularly severe for newly-joining users and newly-arriving items, forming the so-called cold-start problem. As the number of users and items rapidly grows, the sparsity problem and the cold-start problem become the bottleneck of modern recommender systems in business.

Many efforts have been made to combat the long-standing data sparsity problem. Existing methods could be roughly classified into

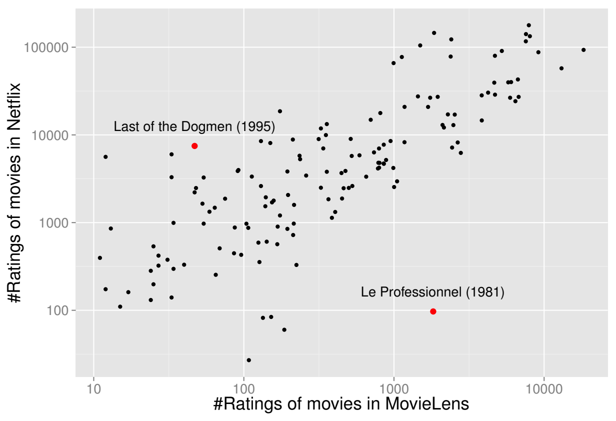
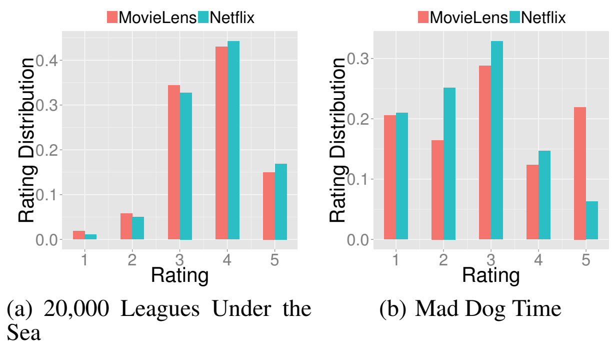


Figure 1: Rating counts in two movie datasets.



(a) 20,000 Leagues Under the Sea

(b) Mad Dog Time

Figure 2: Rating distribution of two movies under two different contexts, i.e., MovieLens and Netflix.

two paradigms. The first focuses on improving recommendation methods or models, dealing with data sparsity problem by introducing appropriate priors or regularization [11]. Although some improvements have been achieved, these methods quickly hit the limit caused by data sparsity of single context. Therefore researchers turn to multi-context recommendation, leveraging feedbacks or ratings from multiple contexts to improve the recommendation accuracy in a particular context [12, 13, 14]. This is motivated by the fact that entities (users or items) are always involved in multiple contexts. For example, one user may express his/her preferences on music at Last.fm and preferences on videos at YouTube; one movie could receive ratings in different movie-rating websites, such as Netflix and MovieLens. Moreover, the degree of data sparsity for a user/item could be quite different across contexts. Figure 1 illustrates the distribution of rating counts for 140 randomly-sampled movies occurring in two different datasets — Netflix and MovieLens. Although the distribution of rating counts in the two datasets as a whole is linearly correlated, several movies have remarkably different rating counts in the two datasets. For example, the French action thriller film '*Le Professionnel*' has only a few ratings in Netflix but gets a lot of ratings in MovieLens, and the western adventure film '*Last of the Dogmen*' is popular in Netflix but cold in MovieLens. For multi-context recommendation, it is expected that users/items that have no or little data in the context of interest may have data in other contexts that could be leveraged to improve the recommendation in the target context.

Collective matrix factorization (CMF) was first proposed to address the multi-context recommendation problem [12]. CMF assumes that *one entity (user or item) shares the same latent factor across different contexts*. The global factor for each entity is learned based on the observed interaction data from all the contexts in which this entity is involved. CMF simply takes the observed interactions in all contexts as being from a single one. Consequently, in CMF, there is a danger that the context with high number of observed interactions could dominate other contexts with few observations. This is problematic for multi-context recommendation because the volume of interactions in different contexts could be remarkably different in real scenario. To address this problem, localized matrix factorization (LMF) [13] and HeteroMF [14] were proposed. In the two models, each entity has two latent factors, i.e., a global latent factor shared across different contexts and a local latent factor specific to each context. Context-specific factors are viewed as being generated from the global one by multiplying it with a context-specific transfer matrix. By modeling context-specific factors and context-independent factors separately, these two models outperform CMF for entities that have spare observations in one context but lots of observations in other contexts. However, their performance is limited by their underlying assumption: the same context exerts the same influence to all entities indiscriminately, reflected by the transfer matrix specific to each context. Indeed, the influence of the same context to different entities could be quite different. Figure 2 depicts the rating distribution of two movies under two contexts. For movie '*20,000 Leagues Under the Sea*', the rating distribution is almost the same in these two contexts, indicating that it is insusceptible to contexts. For movie '*Mad Dog Time*', on the other hand, its rating distribution is highly affected by contexts. Neglecting the difference at the degree to which contexts influence different entities, existing models fail to fully, even improperly, capture the knowledge offered by multi-context observations.

In this paper, we address two critical issues in multi-context recommendation: (1) How to differentiate context-specific factors from entity-intrinsic factors shared across contexts? (2) How to capture such a salient phenomenon: some entities are insensitive to contexts while others are remarkably context-dependent? We propose a context-adaptive matrix factorization (AdaMF) method, addressing the multi-context recommendation problem via simultaneously modeling context-specific factors and entity-intrinsic factors in a unified model. We learn for every entity an entity-intrinsic latent factor shared cross different contexts and a latent factor specific to each context. Meanwhile, using a context-entity mixture parameter matrix we explicitly model the degree to which each context imposes influence on each entity. We develop a Gaussian mixture model to describe the interaction between users' intrinsic factors and users' context-specific factors. Experiments on two real scenarios demonstrate that our method consistently outperforms previous methods for multi-context recommendation on all different sparsity levels.

Our main contributions are summarized as follows:

- We propose a novel AdaMF method for multi-context recommendation, adaptively learning global entity-intrinsic factors shared across different contexts and context-specific factors. Compared with existing models, our model could flexibly distinguish the context-specific interactions and context-independent ones, improving the recommendation performance for multi-context recommendation.

- We apply the AdaMF method in two real scenarios – an item-aligned scenario where movies receive ratings in MovieLens and Netflix, and a user-aligned scenario where users of Douban give ratings to three types of items, i.e., music, books, and movies. Experimental results demonstrate that our method consistently improves the prediction accuracy in both scenarios on all the sparsity levels.

This paper is organized as follows: we first give some preliminaries about multi-context recommendation. Next, related works are described in Section 3. In Section 4, we propose our model and design an EM framework to train the model. In Section 5, we present our experiments on two multi-context datasets, namely an item-aligned context and a user-aligned context. Section 6 concludes this paper.

2. PRELIMINARY

Let us consider L different contexts, each of which could represent a recommendation application. We use special indexing letters to distinguish users from items: a user set $\mathcal{U}$ with M users and an item set $\mathcal{I}$ with N items. These users and items are embedded in the L contexts. We use $R_{ij}^{(l)}$ to denote the rating that user i gives to item j in context l. All the rating information in context l is stored in a context-specific rating matrix $R^{(l)}$. Typically, $R^{(l)}$ is highly sparse. We denote all rating matrices as $R = \{R^{(1)}, R^{(2)}, ..., R^{(L)}\}$. Then the multi-context recommendation problem could be formalized as:

Multi-Context Recommendation: Observing the rating information R from L different contexts, the user set $\mathcal{U}$ and item set $\mathcal{I}$, for a user-item pair (i, j) and a context l such that $R_{ij}^{(l)}$ is unknown, we predict $R_{ij}^{(l)}$.

In real application, there are two typical scenarios of multi-context recommendation, i.e., user-aligned scenario and item-aligned scenario. For user-aligned scenario, users are involved in multiple contexts. For example, a user may register in multiple online social networks, such as Facebook and Twitter. Even in one network, the user could be involved in multiple contexts, reflecting their different actions or feedbacks. For example, in Twitter, a user involves himself in multiple contexts by following other users or forwarding tweets. The other scenario is item-aligned. Taking movie recommendation as an example, there are many movie rating resources, such as Netflix, MovieLens, and IMDB. With movies' names, release date, actors, directors, and other information, we can identify movies that appear in multiple contexts.

What is a context?

Context is a multi-faceted concept across different research disciplines. For context-aware recommender systems (CARS) [15, 16], the term "context" refers to the contextual information when a user takes an action on an item, e.g., rating a movie. Contextual information can be explicit, like time, location, and mood. CARS aims to incorporate these explicit information into recommendation models.

The context considered in this paper refers to the data source of users' (items') information. In the real world, users' and items' information is decentralized into multiple data sources, which could be considered as a context for the users and items involved. Some related works considered cross-domain recommendation [17, 18], a special scenario for multi-context problem where users are aligned across different contexts and items are heterogenous in different contexts. The definition of multi-context recommendation is more general, since items could be homogenous across different contexts. To summarize, the key point of multi-context recommendation is the method of combining the information from different contexts, where items from different contexts could be heterogenous or homogeneous.

Matrix Factorization

Matrix factorization (MF) is one of the state-of-the-art techniques in collaborative filtering. MF was proposed to make prediction for a single user-item rating matrix under single-context scenario. In MF, each user/item is associated with a low-dimensional latent factor. Users' latent factor vectors are stored in matrix U, where latent factor of user i is the ith column of matrix U, denoted by U_i. The latent factor V_j of item j is the jth column of the item factor matrix V. The rating of user-item pair (i, j) is modeled by a probabilistic model with Gaussian observation noise. The conditional distribution over observed rating matrix R as

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)\right]^{I_{ij}}, \qquad (1)$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is an indicator function that reflects whether user i rated item j. Generally, spherical Gaussian priors are placed on user and item factor vectors:

$$p(U|\sigma_u^2) = \prod_{i=1}^{M} (U_i|\vec{\mathbf{0}}, \sigma_u^2 \mathbf{I}), \qquad (2)$$

$$p(V|\sigma_v^2) = \prod_{j=1}^{N} (V_j|\vec{\mathbf{0}}, \sigma_v^2 \mathbf{I}). \qquad (3)$$

Figure 3(a) shows the graphical model of matrix factorization. The latent factors of users and items are inferred in the learning phase using the observed rating matrix. Typically, stochastic gradient descent learning method is used. To predict the rating of user i on item j, the inner product of latent factor vectors U_i and V_j is computed. Our AdaMF model is built on matrix factorization.

3. RELATED WORKS

3.1 Recommendation with Side Information

Former researchers tried to solve data sparsity problem and cold-start problem by involving new information sources such as social relationship and taxonomy information. For example, Ma [19] and Liu [20] included social network information to improve the accuracy of recommender systems. In these works, social information is incorporated into recommendation models as regularization terms, constraining taste difference between a user and his/her friends. Noam [2] and Weng [21] used taxonomy information on the item side. Noam extended the matrix factorization model by incorporating a rich bias model with terms that capture information from the taxonomy of items. Weng modified the similarity measurement with the item taxonomy information, improving the traditional neighborhood-based recommender system.

Recently, some general frameworks are proposed to incorporate multiple kinds of information across multiple contexts. Berkovsky extended the neighborhood-based models into the multi-context scenario [22]. Similarity between users and items are calculated by assembling all the information across all the contexts. Since this work is only an simple extension of neighborhood-based models, the improvement is limited. Chen proposed SVDFeature, a feature-based matrix factorization framework [23]. The feature-based setting allows to build factorization models, incorporating side information like temporal dynamics, neighborhood relationship, and hierarchical information. Xiao solved the recommendation problem in a heterogeneous information network [24]. Using the concept of meta-path to construct many different user-item preference matrices, they built recommendation algorithm based on these matrices separately. The final results are formulated by combining all these preference matrices using a linear model. However, side-information, both from user-side and item-side, is hard to get in many real world applications.

3.2 Transfer Learning

Multi-context recommendation problem could also be done by *transfer learning*, which transfers the knowledge from some auxiliary data source to a target data source [25]. Pan considered a collective factorization model to transfer rating knowledge from some auxiliary data source in binary form to a target numerical rating matrix [26]. Li considered a Codebook transfer (CBT) algorithm under the scenario to transfer rating information across different domains [17]. Auxiliary rating matrix is first compressed into an informative and yet compact cluster-level rating pattern representation referred to as a codebook, and then the target rating matrix is reconstructed by expanding the codebook. However, the hard clustering constraint greatly reduces the expressive power of CBT. A generative model is introduced in [27], relaxing the constraints in CBT from hard clustering to soft clustering using a probabilistic graphical model. In our model, all rating matrices in different contexts are factorized simultaneously, without the need to explicitly distinguish these matrices into auxiliary and target ones. The knowledge is transferred on two directions for all these matrices.

3.3 Collective Matrix Factorization

As discussed before, matrix factorization is proposed for single context, where only one rating matrix is considered. In multi-context recommender system, users (items) will be involved in interactions across multiple contexts. Factorizing the rating matrix in each context separately would not take advantage of any correlation between contexts. We use MF as one of our baseline model in our experiments.

There are some related works of multi-context recommendation based on direct extension of matrix factorization model. Collective matrix factorization (CMF) is proposed by Singh and Gordon [12] to deal with multi-context data. CMF decomposes the rating matrix in each context into a product of two latent factor matrices, representing users' and items' latent interest space respectively. Whenever one entity participates into more than one context, the latent factor for the entity is shared across all contexts. Fig 3(b) shows the probabilistic graphical model of CMF. In this model, latent factors of users and items are shared in all different contexts.

In CMF, the latent factor for an entity is learned based on the observed ratings from the contexts in which the entity participates. However, there is one issue with CMF that objects share the same latent factor across different contexts. This is claimed to be problematic in [13, 14] in two aspects. Firstly, latent factors for objects that are cold in a context will be learned mainly based on the data from other contexts where it is not cold. Consequently, latent factors for the cold objects are not properly learned. Secondly, the latent factors for objects participating in multi-context are learned mainly based on the dominating context and the dominated context has little effect on the learned latent factors. LMF and HeteroMF models are proposed to tackle these problems. In both of these two models, each entity has one global latent factor. For each context, there is a transfer matrix to transfer the global latent factor into the context-specific latent factor. The probabilistic graphical model of HeteroMF is shown in Fig 3(c). Just as the probabilistic graphical model of CMF, there are L contexts, rating for user-item pair (i, j) in the lth context is labeled by $R_{ij}^{(l)}$. For the ith user, the global factor is U_i, and for each item there is a global factor V_j. For each context and the entity (namely, user and item) type, there is a transfer matrix to model the influence of context. The local factor is the result of the interaction between the transfer matrix and the global factor. Taking user-side as an example, the local factor for the ith user in the lth context is generated by the product of the global factor and a transfer matrix M_u^l. Following the notation in the preliminary, the probability distribution of the global and local factors of the ith user are:

$$\begin{aligned} &U_i \sim \mathcal{N}(\vec{\mathbf{0}}, \sigma_u^2 \mathbf{I}), \\ &U_i^{(l)} \sim \mathcal{N}(M_u^l U_i, \sigma_{u,l}^2 \mathbf{I}). \end{aligned} \tag{4}$$

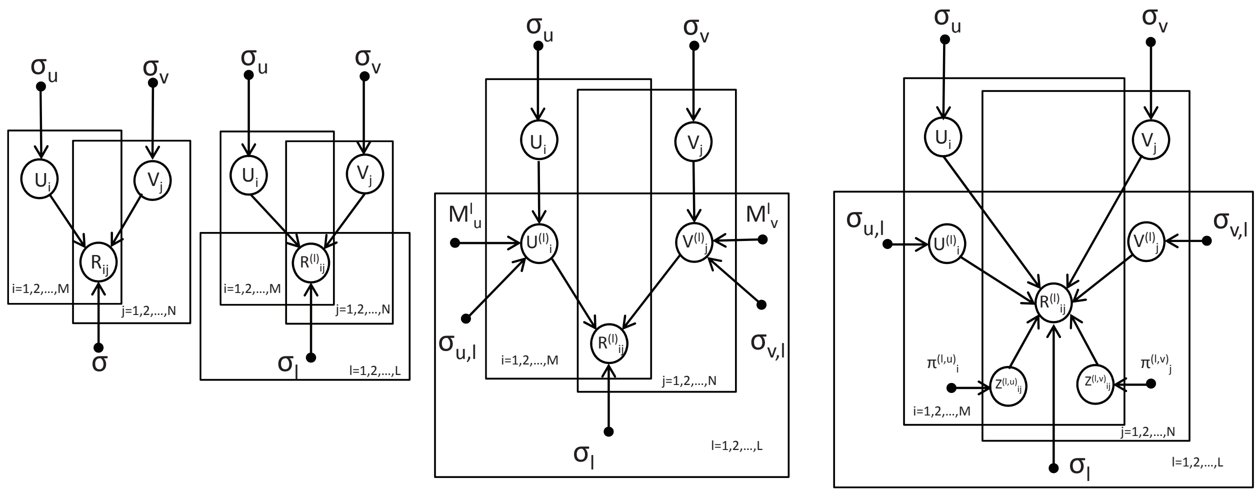
The definition of the global factor and local factor of the items are just the same as the users. The underlying assumption of modeling the context as a transfer matrix is that the effect of one context to all objects in the context is the same. We relax this assumption by directly modeling the context-specific factor of one entity in a given context. The final local factor for one entity is the result of a mixture interaction between the context-specific factor and the global factor. The parameters of the mixture model are adaptively learned from the data.

4. AdaMF

In this section we propose AdaMF, a *context-adaptive matrix factorization* model, to address multi-context recommendation. Consider M users and N items involved in L different contexts. Each context l is associated with a preference matrix $R^{(l)}$, whose element $R_{ij}^{(l)}$ describes the preference (usually a rating) of user i on item j. Our task is to predict the missing value $R_{ij}^{(l)}$.

As shown in Figure 3(d), AdaMF predicts a missing preference $R_{ij}^{(l)}$ based on latent factors representing user i and item j. Different from a standard matrix factorization model that uses one latent factor for each user/item, a sharing-user version of AdaMF uses multiple factors to describe each user, representing her intrinsic preference and context-specific interests respectively, in a scenario where different contexts share an identical set of users. A sharing-item version of AdaMF uses multiple factors to describe each item, representing its intrinsic quality and context-specific quality respectively. Here, as an example, we discuss the sharing-item version of AdaMF without loss of generalization. Specifically, each user i is assigned with a latent factor U_i while each item j is assigned with multiple latent factors, including an *entity-intrinsic factor* V_j related to its inherit property, and a *context-specific factor* $V_j^{(l)}$ associated with each context l. A rating $R_{ij}^{(l)}$ is then drawn from a mixture of two Gaussian distributions, with mean values of $U_i^T V_j$ and $U_i^T V_j^{(l)}$, reflecting the preference of user i on the entity-intrinsic and context-specific part of item j respectively. Those two distributions are mixed with an item-context mixture parameter $\pi_j^{(l)}$, written as

$$\begin{aligned} P(R_{ij}^{(l)} | U_i, V_j, V_j^{(l)}, \pi_j^{(l)}) = \pi_j^{(l)} &\mathcal{N}(R_{ij}^{(l)} | U_i^T V_j^{(l)}, {\sigma_l}^2) \\ &+ (1 - \pi_j^{(l)}) \mathcal{N}(R_{ij}^{(l)} | U_i^T V_j, {\sigma_l}^2). \end{aligned} \tag{5}$$



(a) MF **(b) CMF** **(c) HeteroMF** **(d) AdaMF**

Figure 3: Probabilistic graphical models.

We assume zero-mean spherical Gaussian priors for all latent factors

$$p(U_i|\sigma_u) = \mathcal{N}(U_i|\vec{\mathbf{0}}, \sigma_u^2\mathbf{I}),$$
$$p(V_j^{(l)}|\sigma_{v,l}) = \mathcal{N}(V_j^{(l)}|\vec{\mathbf{0}}, \sigma_{v,l}^2\mathbf{I}),$$
$$p(V_j|\sigma_v) = \mathcal{N}(V_j|\vec{\mathbf{0}}, \sigma_v^2\mathbf{I}),$$

where **I** denotes an identity matrix.

We introduce a latent Bernoulli variable $Z_{ij}^{(l)} \sim Bern(\pi_j^{(l)})$. $Z_{ij}^{(l)} = 0$ indicates that the rating $R_{ij}^{(l)}$ is drawn from the distribution based on the entity-intrinsic factor, while $Z_{ij}^{(l)} = 1$ indicates that the context-specific factor is used.

The joint distribution of $R_{ij}^{(l)}$ and $Z_{ij}^{(l)}$ is as follows,

$$\begin{aligned} &P(R_{ij}^{(l)}, Z_{ij}^{(l)}|U_i, V_j^{(l)}, V_j, \pi_j^{(l)}) \\ &= \left(\pi_j^{(l)}\mathcal{N}(R_{ij}^{(l)}|U_i^T V_j^{(l)}, {\sigma_l}^2)\right)^{Z_{ij}^{(l)}} \\ &\cdot \left((1-\pi_j^{(l)})\mathcal{N}(R_{ij}^{(l)}|U_i^T V_j, \sigma_l^2)\right)^{\left(1-Z_{ij}^{(l)}\right)}. \end{aligned}$$

Accordingly, the joint likelihood of observations and latent variables of the whole dataset could be written as

$$\begin{aligned} P(R, Z, U, V, \bigcup_l V^{(l)}|\pi, \Omega) = \prod_l P(R^{(l)}, Z^{(l)}|U, V, V^{(l)}, \pi^{(l)}, \sigma_l) \\ \prod_i P(U_i|\sigma_u) \prod_j P(V_j|\sigma_v) \prod_l \prod_j P(V_j^{(l)}|\sigma_{v,l}), \end{aligned}$$

where $R = \{R_{ij}^{(l)}\}$, $Z = \{Z_{ij}^{(l)}\}$, $U = \{U_i\}$, $V = \{V_i\}$, $V^{(l)} = \{V_j^{(l)}\}$, and $\Omega = \{\sigma_u, \sigma_v, \bigcup_l \sigma_{v,l}\}$ is the variance parameter set of the model, $\pi = \{\pi^{(l)}\}$ is a mixture parameter matrix. Here, $\pi^{(l)}$ is the mixture parameter vector of items in context l.

4.1 Inference

We use an expectation maximization (EM) algorithm to infer model parameters and latent factors by maximizing the logarithmic likelihood $\mathcal{L}$ of the whole dataset,

$$\begin{aligned} &\mathcal{L} = \\ &\sum_l \sum_i \sum_j \left(Z_{ij}^{(l)} \left(\log \pi_j^{(l)} - \frac{1}{2} \left(\frac{\left(R_{ij}^{(l)} - U_i^T V_j^{(l)}\right)^2}{\sigma_l^2} + \log \sigma_l^2 \right) \right) \right. \\ &\left. + \left(1 - Z_{ij}^{(l)}\right) \left(\log\left(1 - \pi_j^{(l)}\right) - \frac{1}{2}\left(\frac{\left(R_{ij}^{(l)} - U_i^T V_j\right)^2}{\sigma_l^2} + \log \sigma_l^2\right) \right) \right)^{I_{ij}^l} \\ &- \frac{1}{2}\sum_i \frac{1}{\sigma_u}||U_i||_F^2 - \frac{1}{2}\sum_j \frac{1}{\sigma_v}||V_j||_F^2 - \frac{1}{2}\sum_l\sum_j \frac{1}{\sigma_{v,l}}||V_j^{(l)}||_F^2, \end{aligned} \tag{6}$$

where $I_{i,j}^l = 1$ if user i has ratings of item j in context l; $I_{i,j}^l = 0$ otherwise. $||\cdot||_F^2$ denotes the Frobenius norm. In the E-step, we compute the expected logarithmic likelihood with respect to the mixture parameters of items and latent variables. In the M-step, we maximize that expected likelihood of the whole dataset. The two steps are repeated alternately until convergence.

4.1.1 E-step

In the E-step, we use the current mixture parameter values π and latent factors to find the posterior distribution of the latent variables Z. We then use this posterior distribution to find the expectation of the complete-data logarithmic likelihood evaluated according to the mixture parameters and latent factors.

The expected values (responsibility) of $Z_{ij}^{(l)}$ is evaluated as follow,

$$\begin{aligned} &\gamma(Z_{ij}^{(l)}) = E[Z_{ij}^{(l)}] \\ &= \frac{\pi_j^{(l)} \cdot \mathcal{N}(R_{ij}^{(l)}|U_i^T V_j^{(l)}, \sigma_l^2)}{\pi_j^{(l)} \cdot \mathcal{N}(R_{ij}|U_i^T V_j^{(l)}, \sigma_l^2) + (1-\pi_j^{(l)}) \cdot \mathcal{N}(R_{ij}^{(l)}|U_i^T V_j, \sigma_l^2)} \\ &= \frac{1}{1 + \frac{1-\pi_j^{(l)}}{\pi_j^{(l)}} \cdot \frac{e^{-(R_{ij}^{(l)} - U_i^T V_j)^2/\sigma_l^2}}{e^{-(R_{ij}^{(l)} - U_i^T V_j^{(l)})^2/\sigma_l^2}}}. \end{aligned} \tag{7}$$

The expected log-likelihood is computed as follows,

$$
\begin{aligned}
&E_Z[\mathcal{L}] = \\
&\sum_l \sum_i \sum_j \Big(\gamma\left(Z_{ij}^{(l)}\right) \log \pi_j^{(l)} + \left(1 - \gamma\left(Z_{ij}^{(l)}\right)\right) \log\left(1 - \pi_j^{(l)}\right)\Big)^{I_{ij}^l} \\
&\quad - \frac{1}{2}\Big(\gamma\left(Z_{ij}^{(l)}\right)\left(\frac{1}{\sigma_l^2}\left(R_{ij}^{(l)} - U_i^T V_j^{(l)}\right)^2 + \log \sigma_l^2\right) \\
&\quad + \left(1 - \gamma\left(Z_{ij}^{(l)}\right)\right)\left(\frac{1}{\sigma_l^2}\left(R_{ij}^{(l)} - U_i^T V_j\right)^2 + \log \sigma_l^2\right)\Big)^{I_{ij}^l} \\
&\quad - \frac{1}{2}\sum_i \frac{1}{\sigma_u}||U_i||_F^2 - \frac{1}{2}\sum_j \frac{1}{\sigma_v}||V_j||_F^2 \\
&\quad - \frac{1}{2}\sum_l \sum_j \frac{1}{\sigma_{v,l}}||V_j^{(l)}||_F^2.
\end{aligned}
\tag{8}
$$

4.1.2 M-step

In the M-step, we maximize the expected log-likelihood of the complete dataset. Note that the expected log-likelihood could be divided into two parts. The first part only involves the mixture parameters $\pi_j^{(l)}$, which could be estimated by

$$
\pi_j^{(l)} = \frac{\sum_i (\gamma(Z_{ij}^{(l)}))^{I_{ij}^l}}{\sum_i I_{ij}^l}. \tag{9}
$$

The second part of the objective function with respect to the latent variables is maximized by the stochastic gradient descent strategy (SGD) with a tiny update step η. Just as what was done in [9], maximizing the second part of the objective function is equivalent to minimizing the sum-of-squared-errors objective function with quadratic regularization terms, written as

$$
\begin{aligned}
E = &\sum_l \sum_i \sum_j I_{ij}^l \{\gamma(Z_{ij}^{(l)})(R_{ij}^{(l)} - U_i^T V_j^{(l)})^2 \\
&+ (1 - \gamma(Z_{ij}^{(l)}))(R_{ij}^{(l)} - U_i^T V_j)^2\} \\
&+ \lambda_u \sum_i ||U_i||_F^2 + \lambda_v \sum_j ||V_j||_F^2 \\
&+ \sum_l \lambda_{v,l} \sum_j ||V_j^{(l)}||_F^2 + \mathcal{C}.
\end{aligned}
\tag{10}
$$

For simplicity, we set all the σ_l and $\sigma_{v,l}$ as the same for all contexts: so $\lambda_u = \sigma_l^2/\sigma_u^2$, $\lambda_v = \sigma_l^2/\sigma_v^2$, $\lambda_{v,l} = \sigma_l^2/\sigma_{v,l}^2$. We set $\lambda = \{\lambda_u, \lambda_v, \bigcup_l \lambda_{v,l}\}$ as the regularization parameters set, and $\mathcal{C}$ is the constant term.

For each rating in one context $R_{ij}^{(l)}$, we update for U_i, V_j, $V_j^{(l)}$ leveraging gradients,

$$
\begin{aligned}
U_i &\leftarrow U_i - \eta \cdot \Big(\gamma(Z_{ij}^{(l)})(U_i^T V_j^{(l)} - R_{ij}^{(l)})V_j^{(l)} \\
&\qquad + (1 - \gamma(Z_{ij}^{(l)}))(U_i^T V_j - R_{ij}^{(l)})V_j + \lambda_u U_i\Big), \\
V_j &\leftarrow V_j - \eta \cdot \Big((1 - \gamma(Z_{ij}^{(l)}))(U_i^T V_j - R_{ij}^{(l)})U_i + \lambda_v V_j\Big), \\
V_j^{(l)} &\leftarrow V_j^{(l)} - \eta \cdot \Big(\gamma(Z_{ij}^{(l)})(U_i^T V_j^{(l)} - R_{ij}^{(l)})U_i + \lambda_{v,l} V_j^{(l)}\Big).
\end{aligned}
\tag{11}
$$

The framework of the whole algorithm is formalized in Algorithm 1.

After inference, we predict an unobserved rating of user i on item j in context l with its expectation,

$$
\hat{R}_{ij}^{(l)} = \pi_j^{(l)} \cdot U_i^T V_j^{(l)} + (1 - \pi_j^{(l)}) \cdot U_i^T V_j.
$$

Algorithm 1 AdaMF

```
$[U, V, \bigcup_l V^{(l)}, \pi] = AdaMF(R, \eta, \lambda)$
  E-Step:
    For all observed user-item pairs $(i, j)$ in all contexts:
    Evaluate $P(Z_{ij}^{(l)}|R_{ij}^{(l)}, U_i, V_j, V_j^{(l)}) = \gamma(Z_{ij}^{(l)})$
  M-Step:
    Maximize $E_Z[\mathcal{L}]$ with respect of $U, V, \bigcup_l V^{(l)}, \pi$
      Estimate $\pi$ by Eq. (9)
      SGD used in this step:
        For one rating pair in one context $(i, j, l)$, update the latent
  factors and parameters $U_i$,$V_j$,$V_j^{(l)}$ by Eq. (11)
  Repeated until convergence.
```

5. EXPERIMENTS

AdaMF is evaluated with two real-world datasets, compared with mainstream baselines. The MovieLens-Netflix dataset contains users and aligned movies in two contexts corresponding to two public benchmark datasets from Netflix Prize[1] and MovieLens project[2]. $5,871$ movies are aligned by IMDB meta information after data cleaning. Since there are much fewer MovieLens users than Netflix users, we sampled $70,132$ users from Netflix for balancing two contexts. The Douban dataset contains items and aligned users in three contexts, namely books, movies, and music. It is crawled from an online social network Douban[3] where users rate books, movies, and music [28]. We remove inactive users with less than 5 items, and obtain $10,000$ users and their ratings. Statistics of the two datasets are shown in Table 1 and Table 2.

Table 1: Statistics of MovieLens-Netflix dataset

Statistics	MovieLens	Netflix
Num of movies	5,871	5,871
Num of users	69,258	70,132
Num of ratings	7,891,832	11,658,783

Each dataset is split into a 80% training set and a 20% testing set. The dimension of the latent factors are chosen as 20, 50 and 100. In the training phase, we set all regularization parameters as 0.01, although later we empirically find that our model is insensitive to those parameters. The training data is split into 5 parts to do 5-fold cross validation, determining when we stop the learning phase. In the testing phase, we evaluate prediction performance of AdaMF using root mean squared error (RMSE) as,

$$
RMSE = \sqrt{\frac{\sum_{R_{ij} \in R_{testing}} (R_{ij} - \hat{R}_{ij})^2}{|R_{testing}|}},
$$

where R_{ij} denotes the actual rating that user i posts to item j, $\hat{R}_{ij}$ is the predicted rating, and $R_{testing}$ denotes the ratings in the testing set.

Besides the rating-based metric RMSE, we also considered a ranking-based metric, mean average precision (MAP). Given all the candidate items, we generate a recommendation list L for each user by ranking all the items in a descending order according to the predicted rating score. Then, the top-K average precision (AP@K)

[1]http://www.netflixprize.com/
[2]http://www.movielens.org/
[3]http://www.douban.com/

Table 2: Statistics of Douban dataset

Statistics	Movie	Music	Book
Num of users	10,000	10,000	10,000
Num of items	25,342	78,133	51,204
Num of ratings	2,287,712	1,060,704	832,103

is defined as

$$AP@K(u) = \frac{\sum_{j=1}^{K} P(j)I(L_j)}{\sum_{j=1}^{K} I(L_j)},$$

$P(j)$ is the precision of the recommendation list at position j, $I(L_j)$ is the function to indicate if the jth item in the list is preferred by the user. In our experiments, we set the highest rated item as the preferred items for one user. K is set as 5 in our experiments. We evaluate recommendation method using the mean AP over all users, i.e.,

$$MAP = \frac{\sum_{i=1}^{M} AP@K(i)}{M},$$

where M is the number of users.

Three baseline methods are:

- **Matrix Factorization (MF)**: for each context, latent factors of users and items are learned using standard matrix factorization method with regularization.
- **Collective matrix factorization (CMF)**: each user or item has a context-sharing latent factor.
- **HeteroMF**: each entity has a global latent factor shared across contexts. Context-dependent factor for each entity is generated from the global one by multiplying it with a context-specific transfer matrix. This model assumes that the same context exerts the same influence to all entities indiscriminately.

5.1 Prediction Performance

We first compare our algorithm with baseline methods on the whole dataset. Table 3 and table 4 show the performance of all these methods on the movie-aligned dataset. On the two different movie contexts, our method consistently outperforms all the baselines on all the tested number of dimensions. As the dimension of latent factors increases, the improvement becomes more significant.

Table 5 and table 6 show the performance of all the methods on the user-aligned dataset. As there are three domains of items in Douban dataset, we conduct experiments on three scenarios: movie-book, movie-music, and book-music respectively. The results show that our AdaMF model outperforms other baselines on all the three scenarios. However, the improvement of AdaMF is not always significant. For example, the improvement of music recommendation in book-music scenario is not as significant as that in movie-music scenario. This phenomenon may be related to correlation between these domains.

Table 3: RMSE on MovieLens-Netflix dataset

D	Models	MovieLens	Netflix
20	MF	0.8051	0.8334
	CMF	0.7986 (-0.81%)	0.8330 (-0.05%)
	HeteroMF	0.7937 (-1.42%)	0.8315 (-0.23%)
	AdaMF	**0.7907* (-1.79%)**	**0.8296* (-0.46%)**
50	MF	0.8023	0.8321
	CMF	0.7957 (-0.81%)	0.8309 (-0.14%)
	HeteroMF	0.7917 (-1.32%)	0.8300 (-0.25%)
	AdaMF	**0.7859** (-2.04%)**	**0.8245** (-0.91%)**
100	MF	0.7990	0.8303
	CMF	0.7928 (-0.78%)	0.8283 (-0.24%)
	HeteroMF	0.7895 (-1.18%)	0.8280 (-0.27%)
	AdaMF	**0.7821** (-2.12%)**	**0.8203** (-1.20%)**

Significantly outperforms HeteroMF at the:
** 0.01 and * 0.05 level, paired t-test

Table 4: MAP on MovieLens-Netflix dataset

D	Models	MovieLens	Netflix
20	MF	0.6247	0.6885
	CMF	0.6290 (+0.68%)	0.6890 (+0.07%)
	HeteroMF	0.6320 (+1.16%)	0.6898 (+0.18%)
	AdaMF	**0.6404** (+2.50%)**	**0.6930** (+0.65%)**
50	MF	0.6279	0.6902
	CMF	0.6331 (+0.82%)	0.6910 (+0.11%)
	HeteroMF	0.6355 (+1.21%)	0.6912 (+0.14%)
	AdaMF	**0.6450** (+2.72%)**	**0.6962** (+0.87%)**
100	MF	0.6299	0.6925
	CMF	0.6352 (+0.84%)	0.6934 (+0.13%)
	HeteroMF	0.6380 (+1.28%)	0.6937 (+0.17%)
	AdaMF	**0.6490** (+3.03%)**	**0.7001** (+1.09%)**

Significantly outperforms HeteroMF at the:
** 0.01 and * 0.05 level, paired t-test

5.2 Against Sparsity

Due to the imbalance problem in observations, users/items with extremely few ratings suffer most severely from data sparsity. The AdaMF model is particularly effective on those sparsity entities. To show this, we categorize movies in the MovieLens-Netflix dataset into three sparsity levels: "cold" movies with 100 or less ratings in a context, "normal" movies with $100 - 1,000$ ratings, and "warm" movies with more than $1,000$ ratings. In this way, a movie can be warm in MovieLens and normal in Netflix. In total, 7 label configurations are observed, leaving two empty zones, since no warm movies in MovieLens are observed cold in Netflix and vice versa. Figure 4(a) reports the RMSE of prediction on MovieLens (target context), leveraging ratings in Netflix (auxiliary context). In most configurations, standard matrix factorization performs the worst. CMF slightly reduces RMSE on movies that are cold (bottom row) and normal (middle row), but performs even worse on movies labeled warm (top row). That might be due to that it does no good to incorporate external ratings for items with sufficient ratings in the target context. HeteroMF outperforms MF and CMF on movies labeled "normal" and "warm" in the target context (middle and right column), but shows modest advantage on movies labeled "cold" in the target context (left column), implying that a fixed form of transfer matrix sharing mechanism as done in HeteroMF does harm to items sparse in the target context. In contrast, AdaMF consistently shows significant advantage on every configuration, demonstrating that AdaMF works well in any sparse level. Figure 4(b) reports prediction RMSE on Netflix, exhibiting similar results.

5.3 Case Analysis

To offer some intuition about why adaMF works well, we present three typical cases in Figure 5. The first movie is '*Love is all there is*' that receives sparse ratings in both contexts forming a cold-cold case. '*Hellfighter*' is sparse in MovieLens but warm in Netflix, forming a cold-warm case. '*Titanic*' receives plenty of ratings in both contexts forming a warm-warm case. From the perspective of model complexity, a simple model works well when few ratings are available, and a complex model captures more detailed information with a large dataset. Therefore, as the simplest model, CMF uses

Table 5: RMSE on Douban dataset

		Scenario					
D	Models	Movie-Music		Movie-Book		Book-Music	
20	MF	0.7219	0.6933	0.7219	0.7630	0.7630	0.6933
	CMF	0.7277 (+0.80%)	0.6868 (-0.94%)	0.7324 (+1.45%)	0.7460 (-2.22%)	0.7531 (-1.30%)	0.6953 (+0.29%)
	HeteroMF	0.7210 (-0.12%)	0.6869 (-0.92%)	0.7215 (-0.05%)	0.7593 (-0.48%)	0.7552 (-1.02%)	0.6943 (+0.14%)
	AdaMF	**0.7199* (-0.28%)**	**0.6860* (-1.05%)**	**0.7210* (-0.12%)**	**0.7455** (-2.29%)**	**0.7466** (-2.15%)**	**0.6871** (-0.90%)**
50	MF	0.7187	0.6902	0.7187	0.7600	0.7600	0.6902
	CMF	0.7252 (+0.90%)	0.6840 (-0.90%)	0.7284 (+1.35%)	**0.7411 (-2.49%)**	0.7495 (-1.38%)	0.6922 (+0.29%)
	HeteroMF	0.7155 (-0.45%)	0.6835 (-0.97%)	0.7182 (-0.07%)	0.7563 (-0.49%)	0.7482 (-1.55%)	0.6910 (+0.12%)
	AdaMF	**0.7130* (-0.79%)**	**0.6807** (-1.38%)**	**0.7167* (-0.28%)**	0.7421** (-2.36%)	**0.7451* (-1.96%)**	**0.6841** (-0.88%)**
100	MF	0.7152	0.6888	0.7152	0.7555	0.7555	0.6888
	CMF	0.7202 (+0.70%)	0.6811 (-1.12%)	0.7254 (+1.42%)	0.7389 (-2.20%)	0.7480 (-1.00%)	0.6917 (+0.42%)
	HeteroMF	0.7135(-0.24%)	0.6806 (-1.19%)	0.7140 (-0.17%)	0.7550 (-0.07%)	0.7471 (-1.11%)	0.6882 (-0.09%)
	AdaMF	**0.7101** (-0.71%)**	**0.6775** (-1.64%)**	**0.7101** (-0.71%)**	**0.7370** (-2.45%)**	**0.7440* (-1.52%)**	**0.6813** (-1.09%)**

Significantly outperforms HeteroMF at the: ** 0.01 and * 0.05 level, paired t-test

Table 6: MAP on Douban dataset

		Scenario					
D	Models	Movie-Music		Movie-Book		Book-Music	
20	MF	0.8026	0.8890	0.8026	0.8840	0.8840	0.8890
	CMF	0.8020 (-0.07%)	0.8910 (+0.22%)	0.8021 (-0.06%)	0.8950 (+1.24%)	0.8889 (+0.55%)	0.8887 (-0.03%)
	HeteroMF	0.8040 (+0.17%)	0.8908 (+0.20%)	0.8011 (-0.19%)	0.8876 (+0.41%)	0.8872 (+0.36%)	0.8889 (-0.01%)
	AdaMF	**0.8075** (+0.61%)**	**0.9054** (+1.84%)**	**0.8058** (+0.40%)**	**0.8990** (+1.70%)**	**0.8942** (+1.15%)**	**0.9011** (+1.36%)**
50	MF	0.8060	0.8908	0.8060	0.8873	0.8873	0.8908
	CMF	0.8050 (-0.12%)	0.8930 (+0.25%)	0.8055 (-0.06%)	0.8989 (+1.31%)	0.8919 (+0.52%)	0.8912 (+0.04%)
	HeteroMF	0.8078 (+0.22%)	0.8947 (+0.44%)	0.8065 (+0.06%)	0.8904 (+0.35%)	0.8907 (+0.38%)	0.8915 (+0.08%)
	AdaMF	**0.8119** (+0.73%)**	**0.9096** (+2.11%)**	**0.8105** (+0.56%)**	**0.8993** (+1.35%)**	**0.8990** (+1.31%)**	**0.9050** (+1.59%)**
100	MF	0.8093	0.8928	0.8093	0.8901	0.8901	0.8928
	CMF	0.8081 (-0.15%)	0.8955 (+0.30%)	0.8075 (-0.22%)	0.9022 (+1.36%)	0.8943 (+0.47%)	0.8930 (+0.02%)
	HeteroMF	0.8092 (-0.01%)	0.8969 (+0.46%)	0.8090 (-0.04%)	0.8915 (+0.16%)	0.8931 (+0.34%)	0.8932 (+0.04%)
	AdaMF	**0.8130** (+0.46%)**	**0.9115** (+2.09%)**	**0.8175** (+1.01%)**	**0.9043** (+1.60%)**	**0.9020** (+1.34%)**	**0.9081** (+1.71%)**

Significantly outperforms HeteroMF at the: ** 0.01 and * 0.05 level, paired t-test

an identical latent factor to represent all occurrences of a movie on different contexts, and not surprisingly outperforms MF and HeteroMF when predicting the first movie and the sparse side of the second movie. On the other hand, HeteroMF builds a complex model to describe specific behaviors of a movie in multiple contexts, and significantly outperforms CMF for the third movie and the rich side of the second movie. Due to the different applicability of simple and complex models, no baseline consistently perform well in all levels of sparsity. Noticeably, with a flexible parameter to adjust the weight of a global latent factor and a context-specific one, our model actually gains the ability to adaptively fit different levels of sparsity, and as a result outperforms baselines on all three typical cases.

5.4 Mixture Parameters

In AdaMF model, context-entity mixture parameter reflects the balance between entity-intrinsic factor and context-specific factor. One interesting question is: what is the distribution of the values of this mixture parameter over all users/items? As shown in Figure 6, for most movies the mixture parameters locate within $[0.4, 0.6]$, indicating that an entity's intrinsic attributes play almost the same important role in producing preference with the context-specific attributes.

Table 7 shows the mixture parameters for the three movies, studied in the aforementioned case analysis. For movie '*Love is all there is*', mixture parameters are small in both contexts. In this case, entity-intrinsic factors are more critical than context-specific factors. For movie '*Hellfighter*', the mixture parameter is small in the "cold" context but large in the "warm" context, indicating that the AdaMF indeed adaptively capture the degree to which context influences rating. For movie '*Titanic*', a "warm-warm" case, the mixture parameter in both contexts is larger than 0.5, indicating that context matters much more. As above, the diversity of mixture parameters reveal the necessity to develop an adaptive method rather than a fixed form as done in HeteroMF.

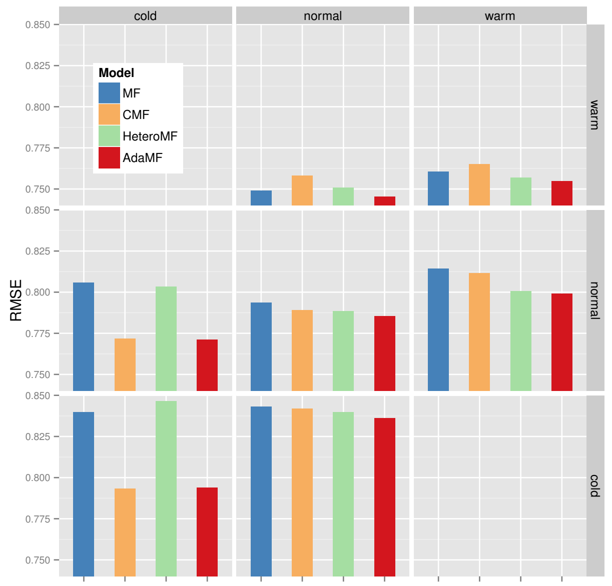
5.5 Model Complexity

We now discuss the issue of model complexity. To this end, we list the number of effective parameters for all the multi-context recommendation models discussed in this paper. We assume that there are M users, N items, and l contexts. We denote with d the dimension of latent factors. For CMF, the effective number is $(M+N)*d$. For HeteroMF, the effective number is $(M+N)*d+d*d*l$. For our AdaMF, the effective number is $(M+N)*d+(M+N)*d*l$. We can see that HeteroMF and AdaMF are more complex than CMF since the two models attempt to model context-specific factors in different schemas. HeteroMF simply assumes that each context indiscriminately affects all entities, reflected by a transfer matrix. AdaMF aims to adaptively capture the influence of context through a context-entity mixture matrix.

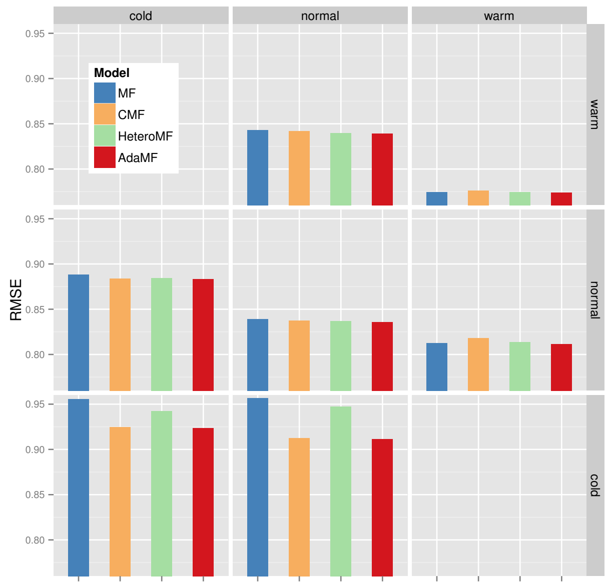
Note that the main superiority of AdaMF over other competing models is its flexibility at adaptively capturing the degree that each context exerts on each entity. As such, AdaMF achieves consistent

Table 7: Mixture parameter learned in the two contexts.

Movie Name	Context	
	MovieLens	Netflix
Love is all there is	0.260	0.330
Hellfighters	0.045	0.821
Titanic	0.601	0.560

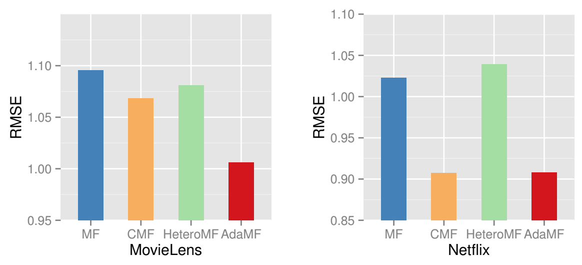


(a) RMSE in MovieLens. X-axis represents the sparsity level in MovieLens, Y-axis represents the sparsity level in Netflix

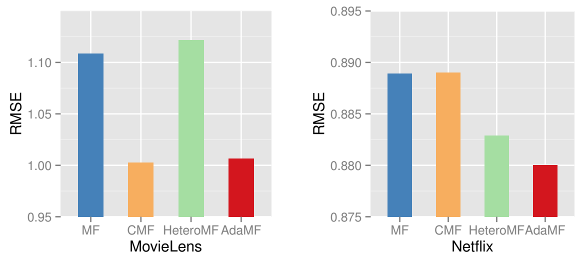


(b) RMSE in Netflix. X-axis represents the sparsity level in Netflix, Y-axis represents the sparsity level in MovieLens

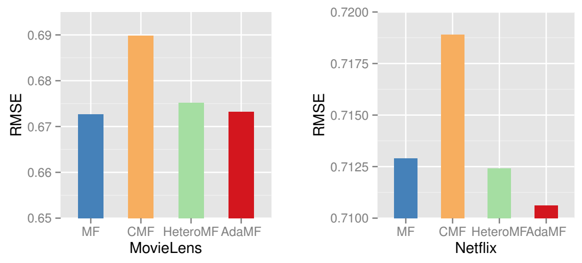
Figure 4: Rating prediction accuracy of movies with different sparse levels.



(a) Cold-cold case: '*Love is all there is*'



(b) Cold-warm case: '*Hellfighters*'



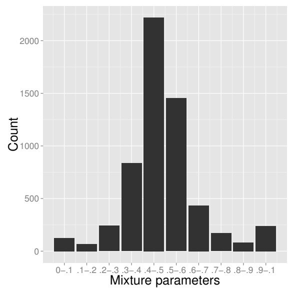
(c) Warm-warm case: '*Titanic*'

Figure 5: Performance in different sparse levels (a) Cold-cold case: movie '*Love is all there is*' has 23 ratings in MovieLens dataset and 32 ratings in Netflix dataset; (b) Cold-warm case: movie '*Hellfighter*' has 16 ratings in MovieLens dataset and 455 ratings in Netflix dataset; (c) Warm-warm case: movie '*Titanic*' has 27550 ratings in MovieLens dataset and 21234 ratings in Netflix dataset.

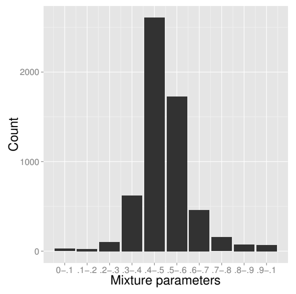
improvements for entities at all sparsity levels, i.e., cold, normal, and warm (Figure 4). This consistent improvement is particularly important for a real recommender system, since such a system exhibits reliable performance than systems that work better on some cases but worse on others. Perhaps, some people may argue that the improvement of the AdaMF is marginal, compared with HeteroMF. Indeed, the main point of this paper is that AdaMF could consistently improve multi-context recommendation by adaptively balancing the entity-intrinsic factor and context-specific factor. Its gain deserves its cost.

6. CONCLUSIONS AND DISCUSSIONS

Data sparsity is one of the challenges for recommender system. Lack of relevant information for users and items is the key issue for data sparsity problem. In the real world, users (items) are involved in multiple contexts but not isolated. In this paper, we propose a context-adaptive matrix factorization method for multi-context recommendation problem via simultaneously modeling context-specific factors and entity-intrinsic factors in a unified model. We learn for every entity an entity-intrinsic latent factor and a context-specific latent factor in each context. Meanwhile, using a context-entity mixture parameter we explicitly model the degree to which each context imposes influence on each entity. We performed experiments on two real scenarios, an item-aligned movie dataset and a user-aligned online social network, demonstrating that our model



(a) Movies in MovieLens



(b) Movies in Netflix

Figure 6: Distribution of mixture parameters.

is better than the baseline models on all sparsity level. As future work, we will extend the AdaMF model to heterogenous networks containing contexts with various types.

Acknowledgments

This work was funded by the National Basic Research Program of China (the 973 program) under grant numbers 2014CB340401 and 2012CB316303, and the National Natural Science Foundation of China with Nos. 61232010, 61472400, 61202215, 61433014. This work is also funded by the National Information Security Program of China (the 242 program) with grant number 2014A137.

7. REFERENCES

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[2] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 165–172. ACM, 2011.

[3] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[4] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

[5] Michael Moricz, Yerbolat Dosbayev, and Mikhail Berlyant. Pymk: friend recommendation at myspace. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 999–1002. ACM, 2010.

[6] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 407–416. IEEE, 2011.

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[8] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[9] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[10] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[11] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[12] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.

[13] Deepak Agarwal, Bee-Chung Chen, and Bo Long. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 609–617. ACM, 2011.

[14] Mohsen Jamali and Laks Lakshmanan. Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the 22nd international conference on World Wide Web*, pages 643–654. International World Wide Web Conferences Steering Committee, 2013.

[15] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[16] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[17] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, volume 9, pages 2052–2057, 2009.

[18] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. Personalized recommendation via cross-domain triadic factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 595–606. International World Wide Web Conferences Steering Committee, 2013.

[19] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[20] Xin Liu and Karl Aberer. Soco: a social network aided context-aware recommender system. In *Proceedings of the 22nd international conference on World Wide Web*, pages 781–802. International World Wide Web Conferences Steering Committee, 2013.

[21] Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 2, pages 113–120. IEEE, 2008.

[22] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling 2007*, pages 355–359. Springer, 2007.

[23] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research*, 13(1):3619–3622, 2012.

[24] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: a heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292. ACM, 2014.

[25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[26] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, volume 10, pages 230–235, 2010.

[27] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 617–624. ACM, 2009.

[28] Junming Huang, Xue-Qi Cheng, Hua-Wei Shen, Tao Zhou, and Xiaolong Jin. Exploring social influence via posterior effect of word-of-mouth recommendations. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 573–582. ACM, 2012.