Improving Implicit Recommender Systems with Auxiliary Data

JINGTAO DING, Tsinghua University
GUANGHUI YU, Washington University in St. Louis
YONG LI, Tsinghua University
XIANGNAN HE, University of Science and Technology of China
DEPENG JIN, Tsinghua University

Most existing recommender systems leverage the primary feedback only, despite the fact that users also generate a large amount of auxiliary feedback. These feedback usually indicate different user preferences when comparing to the primary feedback directly used to optimize the system performance. For example, in E-commerce sites, view data is easily accessible, which provides a valuable yet weaker signal than the primary feedback of purchase. In this work, we improve implicit feedback-based recommender systems (dubbed *Implicit Recommender Systems*) by integrating auxiliary view data into matrix factorization (MF). To exploit different preference levels, we propose both pointwise and pairwise models in terms of how to leverage users' viewing behaviors. The latter model learns the pairwise ranking relations among purchased, viewed, and non-viewed interactions, being more effective and flexible than the former pointwise MF method. However, such a pairwise formulation poses a computational efficiency problem in learning the model. To address this problem, we design a new learning algorithm based on the *element-wise Alternating Least Squares* (eALS) learner. Notably, our designed algorithm can efficiently learn model parameters from the whole user-item matrix (including all missing data), with a rather low time complexity that is dependent on the observed data only. Extensive experiments on two real-world datasets demonstrate that our method outperforms several state-of-the-art MF methods by 6.43%~6.75%. Our implementation is available at https://github.com/dingjingtao/Auxiliary_enhanced_ALS.

CCS Concepts: • **Information systems** → **Recommender systems**; *Learning to rank*; *Personalization*; **Collaborative filtering**;

Additional Key Words and Phrases: Auxiliary feedback, eALS, implicit feedback, item recommendation, matrix factorization

This work was supported in part by The National Key Research and Development Program of China under Grant No. SQ2018YFB180012, the National Nature Science Foundation of China under Grants No. 61971267, No. 61972223, No. 61861136003, and No. 61621091, Beijing Natural Science Foundation under Grant No. L182038, Beijing National Research Center for Information Science and Technology under Grant No. 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.

Authors' addresses: J. Ding, Tsinghua University; email: dingjt15@mails.tsinghua.edu.cn; G. Yu, Washington University in St. Louis; email: ygh690181479@gmail.com; Y. Li, Tsinghua University; email: liyong07@tsinghua.edu.cn; X. He, University of Science and Technology of China; email: xiangnanhe@gmail.com; D. Jin, Tsinghua University; email: jindp@tsinghua.edu.cn.

1046-8188/2020/02-ART11 $15.00
https://doi.org/10.1145/3372338

ACM Reference format:
Jingtao Ding, Guanghui Yu, Yong Li, Xiangnan He, and Depeng Jin. 2020. Improving Implicit Recommender Systems with Auxiliary Data. *ACM Trans. Inf. Syst.* 38, 1, Article 11 (February 2020), 27 pages.
https://doi.org/10.1145/3372338

1 INTRODUCTION

Recent research on recommendation has shifted from explicit ratings [19] to implicit feedback, such as purchases, clicks, and watches [3]. Distinct from explicit feedback-like ratings, in implicit feedback data, the negative signal about user preference over items is naturally scarce, resulting in one-class learning problem [31, 54]. Therefore, to learn from implicit feedback, it is crucial to account for both observed and missing data. A state-of-the-art MF method for implicit feedback is the eALS [15], which treats all missing data as the negative feedback but with a lower weight. This whole-data-based formulation has been shown to be superior to the prevalent sampling-based method [35] that models partial missing data only.

In an online information system, in addition to the primary feedback that is directly related with the business KPI, there are also other kinds of user feedback available [17, 33, 41]. For example, in E-commerce sites, a user must view a product, i.e., click the product page, before purchasing it. This kind of view data provides valuable signal on user preference, which can complement the purchase data in two folds. First, if a user views an item, regardless of whether purchasing or not, it at least reflects that the user is interested in the item, i.e., positive signal, compared to the non-viewed items. Second, if a user views a product but does not purchase it afterward, it means that the item is of less interest to the user, i.e., negative signal, compared to the purchased items. As such, view data can be seen as an intermediate feedback between purchase and missing data, which enriches the two-level implicit feedback with multiple levels that better distinguish user preference. Previous works on utilizing this auxiliary information in sampling-based implicit MF methods have indeed shown the superior performance in terms of learning user preference [27, 28]. However, for the state-of-the-art implicit MF method, i.e., eALS, there still remains no progress in extending it into a multiple-feedback scenario. Therefore, in this work, we aim to integrate the valuable auxiliary data into the eALS, to enhance the performance of implicit recommender systems by taking advantage of both data augmentation and whole-data-based learning strategy.

Nevertheless, it is non-trivial to integrate such intermediate feedback into the eALS, which is designed for learning from binary 0/1 data only. Specifically, it performs regression by treating purchased interactions as having a label of 1 and other missing interactions as having a label of 0. Unlike a plain MF designed for explicit rating data with different rating scores, modeling auxiliary signal in eALS requires a specific design to learn more accurate user preferences and, at the same time, not to impact the modeling of missing data.

In this work, we make a novel technical contribution in integrating intermediate feedback into eALS. Our first solution is to assign the auxiliary interactions with an "intermediate label," i.e., a value between 0 and 1, and do pointwise learning. Since setting a uniform value for all intermediate interactions oversimplifies the problem, we further propose a pairwise learning solution that models the relative preference orders among different interactions. Specifically, taking the twofold semantics of view data as an example, we consider the pairwise ranking relations between (1) purchased and viewed interactions, and (2) viewed and non-viewed interactions. The idea is to regularize eALS by enforcing that the predictions of a user over purchased items should be larger than that of viewed items, and the same regularization applies to viewed and non-viewed items.

Despite soundness, this solution poses strong challenges to the learning efficiency. In particular, the large number of non-viewed interactions (i.e., missing entries) makes even pointwise

regression over them become unaffordable [15], not to mention the pairwise comparisons between viewed and non-viewed interactions, which introduce near $|\mathcal{V}| \cdot N$ terms with $|\mathcal{V}|$ and N as the number of interactions (additional feedback) and items, respectively. To make the learning tractable, we develop a fast algorithm that leverages the bilinear structure of MF to achieve speedups. Through rigorous mathematical analysis, we identify computational bottlenecks in optimization and resolve the bottlenecks via clever design of memoization strategies. Moreover, we demonstrate how to extend the proposed pointwise and pairwise models into the more general cases where multiple types of auxiliary feedback data are available.

We summarize the contributions of the article as follows.

- We improve implicit recommender systems by incorporating auxiliary view data, proposing both Pointwise View-enhanced eALS (PointVALS) method and Pairwise View-enhanced eALS (PairVALS) method. Different from the former pointwise solution, the PairVALS models the pairwise relations among purchased, viewed, and non-viewed interactions. We also extend both PointVALS and PairVALS into a more general scenario where multiple types of auxiliary feedback are available.
- We propose a fast algorithm that solves the challenging PairVALS problem with a controllable time complexity that is determined by the number of observed interactions only. Compared to a direct implementation, our algorithm is almost N times faster, which empirically costs the same magnitude of time as PointVALS.
- We conduct extensive experiments on two real datasets, with one and two kinds of auxiliary feedback, respectively, demonstrating both scalability and accuracy of our PointVALS and PairVALS methods. Integrating these additional information can achieve a relative performance of 4.2%~17.5%. When comparing with state-of-the-art auxiliary-enhanced recommender systems, our proposed methods still outperform with a large margin, about 6.43%~6.75%. Moreover, with much less training iterations, their overall training time is significantly shorter than other sampling-based methods.

The remainder of the article is organized as follows. We first introduce related works in Section 2. We then provide the preliminaries and detailed model design in Sections 3 and 4, respectively. The empirical experiments are performed in Section 5, followed by a prospect for future work.

2 RELATED WORK

We first review some related works on modeling user preference from implicit data. Then, we discuss two types of methods for improving implicit recommender systems with multiple feedback. As the two main contributions of our proposed model include a margin-based design of loss function and a fast-learning algorithm, we also discuss the related works on these two aspects.

Implicit Recommender Systems. Handling missing data is notoriously difficult for recommendation with implicit feedback, which is a common problem existing in both shallow and deep recommendation models [46, 47]. To solve this problem, two strategies are proposed. Sampling-based learning strategy overcomes this problem by sampling negative instances from missing data [9, 31, 35], while whole-data-based strategy treats all missing data as negative [15, 16]. Though suffering the efficiency issue, whole-data-based methods always have higher prediction accuracy. Therefore, when leveraging the additional view data in implicit recommender systems, we choose to develop a view-enhanced MF method based on eALS [15], which is the state-of-art implicit MF method that adopts the whole-data-based learning strategy. Moreover, we provide a fast-learning algorithm to resolve the inherent computational inefficiency issue.

Learning from Multi-feedback Data. In terms of methods for recommending items based on multiple types of feedback, the first type is the model-based methods that design

independent models for each feedback type. A typical method of this type is collective matrix factorization (CMF), which performs multiple relational learning by sharing information between models of different feedback [7, 38, 51]. While CMF was originated for explicit rating prediction, it has been extended for implicit recommender systems as well [5, 20, 48, 55]. However, as CMF-based model generates different user-item relations, i.e., latent factors, for each type of feedback, it is hard to differentiate their preference levels. Besides CMF, other methods also consider to represent a complementary user vector learnt from auxiliary feedback data, like SVD++ [18] and a recently proposed method [27], namely, Multiple Feedback Personalized Ranking (MFPR). Similar to CMF-based methods, it still cannot represent preference order among different types of feedback. In real-world scenarios, the viewed interactions are a superset of add-cart, which are also a superset of purchase. Recently, this relationship has also been analyzed and utilized to boost the recommendation performance by tensor factorization [43] and neural multi-task learning [12]. To leveraging users' preference information in both browses and pruchases, a transfer-learning-based solution is proposed based on the assumption that users have two roles, i.e., browser and purchaser, in e-commerce websites [32]. Based on the observation that the underlying spectrum of user preferences is reflected in various types of interactions with items, a unified neural learning framework is proposed in terms of latent relational learning in metric space [56]. In contrast, our VALS method learns the same user-item relation to explicitly indicate relative preference order among purchase and view data, which is more effective.

The second type is the learning-based methods that integrate multiple types of feedback in a negative sampler of BPR [23]. Specifically, Multi-channel BPR (MC-BPR) assigns different preference levels to different types of user feedback. Then, in the training process, it samples the item pairs based on these preference orders [28, 29]. A recent proposal also jointly learns from a primary feedback and secondary (auxiliary) feedback and observes significant performance improvements [17]. However, these BPR-based solutions still suffer from the shortcomings of sampling-based methods [9], i.e., degradation on both performance and fidelity, as well as an expensive tuning on learning rate. Our VALS method differs from them by integrating different preference levels based on whole-data-based learning strategy. To our knowledge, VALS is the first attempt to exploit different preference levels of implicit feedback in whole-data-based MF methods.

Element-wise Alternating Least Squares. With advantage of quick convergence, Alternating Least Squares (ALS) is a popular approach to optimize implicit MF model [16, 26] and graph regularization [14]. By optimizing parameters at the element level, i.e., optimizing each coordinate of the latent vector, and combining with the specificlly designed memoization strategy, element-wise ALS technique is able to achieve state-of-the-art efficiency for implicit MF model [15], faster than other scalable models [8, 42]. Recently, it has been extended into a content-aware MF model integrating both user and item features [25] and a cross-city POI recommendation framework modeling both user interest drift and transfer [11]. Motivated by them, we also leverage eALS technique to speed-up the learning process and, with the specific design of handling pairwise ranking relations among different types of feedback, achieve a scalable time complexity that is linear to the observed data size.

Margin-based Loss Function. The margin-based loss functions are generally used when learning to discriminate the positive and negative instances with a large margin. In the field of collaborative prediction, maximum-margin matrix factorization firstly adopted this idea by maximazing the margin [39]. Later on, the well-known BPR algorithm for implicit recommender systems also achieved the maximum-margin between a positive instance and an unobserved instance, for each training pair [35]. Different from them, the margin can be designed as a hyper-parameter [4] and thus finely tuned to achieve better prediction performance [22, 52]. In our case with multiple user feedback, it is natural to design a margin-based loss function to characterize those auxiliary

Table 1. List of Commonly Used Notations

Notation	Description
M, N, K	The numbers of users, items, and factors.
$\mathbf{P}, \{\mathbf{p}_u\}$	The latent factor matrix and vector for users.
$\mathbf{Q}, \{\mathbf{q}_i\}$	The latent factor matrix and vector for items.
$\mathcal{R}, \mathcal{R}_u, \mathcal{R}_i$	The sets of all purchased (u, i) pairs, items purchased by u, users that have purchased i.
$\mathcal{V}, \mathcal{V}_u, \mathcal{V}_i$	Notations for viewed interactions, similar to those for purchased interactions.
$\mathcal{RV}, \mathcal{RV}_u, \mathcal{RV}_i$	Notations for the union of $\mathcal{R}$ and $\mathcal{V}$, i.e., $\mathcal{R} \cup \mathcal{V}$.
$\hat{r}_{ui}, \hat{r}_{uv}, \hat{r}_{uj}$	Predictions of user u over purchased items i, viewed items v and non-viewed items j.
ω_{ui}	Weight of the purchased interaction (u, i).
s_j	Item-oriented weight of item j in missing data.
c_v	Item-oriented weight of item v in view data.
γ_1, γ_2	Margin value between $\hat{r}_{ui}$ and $\hat{r}_{uv}$, $\hat{r}_{uv}$ and $\hat{r}_{uj}$.
λ	Regularization parameter.

feedback, which represent the intermediate user preference, neither positive nor negative. Moreover, our designed loss function models the pairwise ranking relations with the form of least squared loss, which is a general form of similar loss in previous works that only considered primary feedback [40, 53].

Based on the original version of this work [10], the following fields are substantially enhanced. To leverage auxiliary user feedback data, we design both pointwise and pairwise methods for learning user preference. As for the proposed alternating optimization algorithm that aims to learn the pairwise model fast, we provide the design of the updating process for item latent factors, which is far more complicated than that of user latent factors. More importantly, we discuss the generality of these models in terms of incorporating various auxiliary feedback. Our further experiment demonstrates their capability of improving implicit recommender system with more than one auxiliary feedback. Last but not least, compared with experiments in the previous version, we additionally investigate the efficiency of proposed methods, the advantage of whole-data-based learning strategy and compare with more competitive baselines.

3 PRELIMINARIES

We start by introducing some basic notations. For a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$, M and N denote the number of users and items, respectively, and $\mathcal{R}$ denotes the set of user-item pairs that have interactions. For a specific user u, vector $\mathbf{p}_u \in \mathbb{R}^K$ denotes the K-dimensional latent feature vector, and set $\mathcal{R}_u$ denotes the set of items that are interacted by u. Similarly, for an item i, notations $\mathbf{q}_i$ and $\mathcal{R}_i$ are used. Matrices $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ denote the latent factor matrix for users and items. The standard MF is used as the predictive model. Mathematically, each entry r_{ui} of $\mathbf{R}$ is estimated as $\hat{r}_{ui} = <\mathbf{p}_u, \mathbf{q}_i> = \mathbf{p}_u^T \mathbf{q}_i$. For readability, we summarize the major notations throughout the article in Table 1.

To learn user/item latent factors, Ref. [15] developed an eALS method that introduces a weighted regression function, which assigns a zero r_{ui} value to missing entries with a confidence variable:

$$J = \sum_{(u,i) \in \mathcal{R}} \omega_{ui} (\hat{r}_{ui} - r_{ui})^2 + \sum_{u=1}^{M} \sum_{i \notin \mathcal{R}_u} s_i \hat{r}_{ui}^2 + \lambda \left(||\mathbf{P}||_F^2 + ||\mathbf{Q}||_F^2 \right), \tag{1}$$

where ω_{ui} denotes the weight of entries $(u, i) \in \mathcal{R}$. Determined by an item popularity-aware weighting strategy, s_i denotes the confidence that item i missed by users is a true negative assessment.

The above problems can be solved using ALS-based technique by iteratively optimizing each coordinate of the latent vector, while leaving others fixed [36]. In each iteration, the fth latent factor of u's vector, namely, p_{uf}, is updated in sequence by setting the derivative of objective function to 0. After computing all the user factors, item factors can be similarly updated. Due to space limit, we directly list the update rule for user/item factors,

$$
\begin{aligned}
p_{uf} &= \frac{\sum_{i \in \mathcal{R}_u} [\omega_{ui} r_{ui} - (\omega_{ui} - s_i) \hat{r}_{ui}^f] q_{if} - \sum_{k \neq f} p_{uk} e_{kf}^q}{\sum_{i \in \mathcal{R}_u} (\omega_{ui} - s_i) q_{if}^2 + e_{ff}^q + \lambda}, \\
q_{if} &= \frac{\sum_{u \in \mathcal{R}_i} [\omega_{ui} r_{ui} - (\omega_{ui} - s_i) \hat{r}_{ui}^f] p_{uf} - s_i \sum_{k \neq f} q_{ik} e_{kf}^p}{\sum_{u \in \mathcal{R}_i} (\omega_{ui} - s_i) p_{uf}^2 + s_i e_{ff}^p + \lambda},
\end{aligned}
\tag{2}
$$

where two caches are defined as $\mathbf{E}^q = \sum_{i=1}^N s_i \mathbf{q}_i \mathbf{q}_i^T \in \mathbb{R}^{K \times K}$ and $\mathbf{E}^p = \sum_{u=1}^M \mathbf{p}_u \mathbf{p}_u^T \in \mathbb{R}^{K \times K}$ [15]. By pre-computing these two caches between each iteration, the total time complexity is $O((M + N)K^2 + |\mathcal{R}|K)$ for one iteration, which approaches SGD method $\sim O(|\mathcal{R}|K)$ when $(M + N)K$ and $|\mathcal{R}|$ are close.

Since eALS learns latent factors from the whole missing data, it not only can retain model's fidelity but also achieves higher accuracy than the sampling-based methods that sample negative instances from missing data. Considering these advantages, we choose to develop a view-enhanced whole-data-based method based on this framework (details in Section 4).

4 PROPOSED VALS MODEL

Based on eALS, we consider how to effectively learn user preference from both purchase and view data. On the one hand, to differentiate their preference levels using the same user-item relation, we consider the view signal as an intermediate feedback and do pointwise learning. On the other hand, following the idea of learning to rank, we consider the pairwise ranking order between a viewed item and purchased (or non-viewed) item in the objective function. More specifically, we use two margins to describe the intermediate preference level of user's viewed interactions, which is lower than purchased ones but higher than non-viewed ones. However, introducing the above pairwise relationship in view-enhanced objective function makes the original eALS learning algorithm become N times slower, which is unsuitable for large-scale data. Therefore, we further develop a fast VALS learning algorithm to efficiently optimize the view-enhanced objective function. Finally, we discuss the generality of our proposed method when faced with multiple types of auxiliary feedback data, i.e., not only the view data.

4.1 Pointwise Regression Model

In E-commerce recommender system, besides the purchases as the primary feedback that is directly related to optimizing the conversion rate, the view logs of users can be intuitively treated as the intermediate feedback between the purchased and missing data. Therefore, for user u's viewed item v, it should have an intermediate value of prediction $\hat{r}_{uv}$ between those of non-viewed item j (i.e., missing entry) and purchased item i, i.e., $\hat{r}_{uj}$ and $\hat{r}_{ui}$. An intuitive solution is to assign a specific label value r_{uv} for $\hat{r}_{uv}$ and optimize the squared error between them, following the idea of pointwise learning [15, 16].

Therefore, we propose the following view-enhanced objective function:

$$\begin{aligned} L &= L_{\text{eALS}} + L_{\text{view}} + L_{\text{Reg}} \\ &= \sum_{(u,i)\in\mathcal{R}} \omega_{ui}(\hat{r}_{ui} - r_{ui})^2 + \sum_{u=1}^{M} \sum_{j\notin\mathcal{R}\mathcal{V}_u} s_j \hat{r}_{uj}^2 \\ &\quad + \sum_{(u,v)\in\mathcal{V}} c_v(\hat{r}_{uv} - r_{uv})^2 + \lambda\Big(||\mathbf{P}||_F^2 + ||\mathbf{Q}||_F^2\Big). \end{aligned} \tag{3}$$

Note that this objective function can be divided into three terms, where the L_{eALS} and L_{Reg} represent the prediction error and regularizer in the former eALS solution, while the L_{view} term describes the intermediate preference level of the view signal. For u's viewed item v, the prediction $\hat{r}_{uv}$ is optimized to be a fixed value r_{uv}, where $r_{uv} \in [0, 1]$. By tuning r_{uv}, we are able to explore the preference level that best characterizes the view signal. For example, the viewed interactions can be seemed as the pure positive feedback with $r_{uv} = 1$ and pure negative feedback with $r_{uv} = 0$, respectively.

To learn model parameters, i.e., latent factors for each user and item, the similar speed-up technique can be leveraged. Since our proposed L_{view} term also has the similar form as that purchase-related term in L_{eALS}, we can directly rewrite the corresponding update rule based on Equation (2). For user factors p_{uf}, it can be updated by

$$\begin{aligned} p_{uf} &= \frac{\sum_{i\in\mathcal{R}\mathcal{V}_u}[\omega'_{ui} r_{ui} - (\omega'_{ui} - s_i)\hat{r}_{ui}^f]q_{if} - \sum_{k\neq f} p_{uk} e_{kf}^q}{\sum_{i\in\mathcal{R}\mathcal{V}_u}(\omega'_{ui} - s_i)q_{if}^2 + e_{ff}^q + \lambda}, \\ \text{where } \omega'_{ui} &= \begin{cases} \omega_{ui} & i \in \mathcal{R}_u, \\ c_i & i \in \mathcal{V}_u. \end{cases} \end{aligned} \tag{4}$$

Compared to Equation (2), the viewed interactions are integrated and share the same notation i with the purchased interactions. Similarly, item factors q_{if} can be updated by

$$\begin{aligned} q_{if} &= \frac{\sum_{u\in\mathcal{R}\mathcal{V}_i}[\omega'_{ui} r_{ui} - (\omega'_{ui} - s_i)\hat{r}_{ui}^f]p_{uf} - s_i \sum_{k\neq f} q_{ik} e_{kf}^p}{\sum_{u\in\mathcal{R}\mathcal{V}_i}(\omega'_{ui} - s_i)p_{uf}^2 + s_i e_{ff}^p + \lambda}, \\ \text{where } \omega'_{ui} &= \begin{cases} \omega_{ui} & u \in \mathcal{R}_i, \\ c_i & u \in \mathcal{V}_i. \end{cases} \end{aligned} \tag{5}$$

In a word, the total complexity for updating all the user and item vectors in one iteration is $O((M + N)K^2 + (|\mathcal{R}| + |\mathcal{V}|)K)$, which only depends on the number of observed interactions. We denote this method as pointwise view-enhanced eALS (PointVALS) method.

4.2 Pairwise Ranking Model

Though PointVALS is able to model users' viewed interactions as the intermediate feedback, it is difficult to choose an appropriate $\hat{r}_{uv}$ for different users. To solve this problem, instead of assigning a uniform value r_{uv} for $\hat{r}_{uv}$ to optimize, we consider the pairwise ranking between the $\hat{r}_{uv}$ and the predictions over other items, including $\hat{r}_{uj}$ and $\hat{r}_{ui}$. By this means, we are able to differentiate the preference levels between view feedback and others, in a more accurate and flexible way.

The pairwise view-enhanced objective function is then designed as follows:

$$\begin{aligned} L &= L_{\text{eALS}} + L_{\text{view}} + L_{\text{Reg}} \\ &= \sum_{(u,i)\in\mathcal{R}} \omega_{ui}(\hat{r}_{ui} - r_{ui})^2 + \sum_{u=1}^{M} \sum_{j\notin\mathcal{R}\mathcal{V}_u} s_j \hat{r}_{uj}^2 \\ &\quad + \sum_{(u,v)\in\mathcal{V}} c_v \left[\sum_{i\in\mathcal{R}_u} \Big(\gamma_1 - (\hat{r}_{ui} - \hat{r}_{uv})\Big)^2 + \sum_{j\notin\mathcal{R}\mathcal{V}_u} \Big(\gamma_2 - (\hat{r}_{uv} - \hat{r}_{uj})\Big)^2 \right] \\ &\quad + \lambda\Big(||\mathrm{P}||_F^2 + ||\mathrm{Q}||_F^2\Big). \end{aligned} \tag{6}$$

Again, this objective function can be divided into three terms, where L_{view} term describes the intermediate preference level of the view signal. For u's viewed item v, the pairwise ranking between v and another purchased item i or non-viewed item j is achieved through a margin-based loss. More specifically, prediction $\hat{r}_{uv}$ is optimized to be lower than $\hat{r}_{ui}$ with a margin, namely, γ_1, as indicated by $(\gamma_1 - (\hat{r}_{ui} - \hat{r}_{uv}))^2$ term. Similarly, $\hat{r}_{uv}$ is optimized to be higher than $\hat{r}_{uj}$ with another margin, namely, γ_2. For each (u, v), $N - |\mathcal{V}_u|$ terms are considered in pairwise ranking loss, which has a total time cost of $O(|\mathcal{V}|(N - |\mathcal{V}_u|))$. The sum of each (u, v)'s loss is aggregated with an v-dependent weight, namely, c_v, which indicates the weight of view data in L. By varying margin values (γ_1, γ_2), we are able to control possible scoring range of predictions over viewed items, and thus search the best preference level for view signal. Without loss of generality, we only consider those viewed but not purchased items, indicating $\mathcal{R} \cap \mathcal{V} = \phi$.

4.3 Fast-learning Algorithm

As mentioned above, the pairwise ranking loss between a viewed item and another one introduces nearly $|\mathcal{V}| \cdot N$ terms in L_{view} of Equation (6), making the time cost become N times more if we directly use the original eALS method [15]. To overcome this efficiency challenge, we develop a fast Pairwise VALS (PairVALS) learning algorithm that can speed-up this learning process by avoiding massive repeated computations in L_{view}. We first detail this for user latent factors, followed by the counterpart for item factors.

4.3.1 Updating User Factors. First, following the idea of ALS technique, the user u's f^{th} latent factor is updated by setting $\partial L/\partial p_{uf}$ to 0, while the others are fixed. Since $L = L_{\text{eALS}} + L_{\text{Reg}} + L_{\text{view}}$ and the speed-up strategies of L_{eALS} and L_{Reg} have been discussed in [15], we only present the speed-up of L_{view} term. According to Equation (6), we obtain the derivative of L_{view} *w.r.t.* p_{uf} as follows:

$$\frac{\partial L_{\text{view}}}{\partial p_{uf}} = 2 \sum_{v\in\mathcal{V}_u} \sum_{i\in\mathcal{R}_u} c_v (q_{if} - q_{vf})^2 \cdot p_{uf} \tag{7}$$

$$+ 2 \sum_{v\in\mathcal{V}_u} \sum_{j\notin\mathcal{R}\mathcal{V}_u} c_v (q_{jf} - q_{vf})^2 \cdot p_{uf} \tag{8}$$

$$+ 2 \sum_{v\in\mathcal{V}_u} \sum_{i\in\mathcal{R}_u} c_v (\hat{r}_{ui}^f - \hat{r}_{uv}^f)(q_{if} - q_{vf}) \tag{9}$$

$$+ 2 \sum_{v\in\mathcal{V}_u} \sum_{j\notin\mathcal{R}\mathcal{V}_u} c_v (\hat{r}_{uj}^f - \hat{r}_{uv}^f)(q_{jf} - q_{vf}) \tag{10}$$

$$- 2 \sum_{v\in\mathcal{V}_u} \Big\{ \sum_{i\in\mathcal{R}_u} \gamma_1 c_v (q_{if} - q_{vf}) - \sum_{j\notin\mathcal{R}\mathcal{V}_u} \gamma_2 c_v (q_{jf} - q_{vf}) \Big\}, \tag{11-12}$$

where $\hat{r}^f_{u\bullet} = \hat{r}_{u\bullet} - p_{uf}q_{\bullet f}$, i.e., the prediction without the component of latent factor f. Clearly, the bottleneck lies in Equations (8), (10), and (12) terms that contain summations over item pairs (v, j), introduced by the pairwise ranking between viewed items and non-viewed items. It takes $O(|\mathcal{V}_u|(N - |\mathcal{RV}_u|))$ time for a raw implementation. To solve this inefficiency issue, we first break down the summations over item pairs into two independent summations over one item index only, which reduces the time complexity into $O(N - |\mathcal{RV}_u|)$. Then, we further apply memoization strategy to avoid the massive repeated computations on non-viewed item j, achieving an efficient learning in $O(|\mathcal{RV}_u| + K)$ time.

We detail the above process by focusing on Equation (10) term in $\partial L_{\text{view}}/\partial p_{uf}$, as the rest can be done likewise. More specifically, we can obtain

$$\begin{aligned}\sum_{v\in\mathcal{V}_u}\sum_{j\notin\mathcal{RV}_u} c_v(\hat{r}^f_{uj}-\hat{r}^f_{uv})(q_{jf}-q_{vf}) &= \sum_{v\in\mathcal{V}_u} c_v \cdot \sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf} \\ - \sum_{v\in\mathcal{V}_u} c_v q_{vf}\cdot\sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj} - \sum_{v\in\mathcal{V}_u} c_v\hat{r}^f_{uv}\cdot\sum_{j\notin\mathcal{RV}_u} q_{jf} &+ \sum_{v\in\mathcal{V}_u} c_v\hat{r}^f_{uv}q_{vf}\cdot\sum_{j\notin\mathcal{RV}_u} 1.\end{aligned} \tag{13}$$

By this reformulation, we observe that each term above can be factorized as two parts that are only dependent on one item index, i.e., viewed item v or non-viewed item j. Then, the original summation over item pairs (v, j) can be broken down into two independent summations. As the summation over v takes $O(|\mathcal{V}_u|)$ time, the current bottleneck falls onto those j-dependent summations, which require a traversal of the whole negative space and take near $O(N)$ time. Therefore, we move forward to speed up the calculation of these terms,

$$\begin{aligned}\sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj} &= \sum_{j=1}^{N}\hat{r}^f_{uj} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} = \sum_{j=1}^{N}\sum_{k\neq f} p_{uk}q_{jk} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} \\ &= \sum_{k\neq f} p_{uk}\sum_{j=1}^{N} q_{jk} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}; \\ \sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf} &= \sum_{k\neq f} p_{uk}\sum_{j=1}^{N} q_{jk}q_{jf} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf}.\end{aligned} \tag{14}$$

As shown above, the major computation lies in $\sum_{j=1}^{N} q_{jk}$ and $\sum_{j=1}^{N} q_{jk}q_{jf}$ terms, which are independent of u. Therefore, when updating the latent factors for different users, it is unnecessary to repeatedly compute these terms.

We define $\mathbf{D}^q$ cache as $\mathbf{D}^q = \sum_{i=1}^{N}\mathbf{q}_i \in \mathbb{R}^K$, and $\mathbf{E}^q$ cache as $\mathbf{E}^q = \sum_{i=1}^{N}\mathbf{q}_i\mathbf{q}_i^T \in \mathbb{R}^{K\times K}$, which can be pre-computed and used in updating the latent factors for all users. Based on these two caches, Equation (14) can be further evaluated as

$$\sum_{k\neq f} p_{uk}d^q_k - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} \text{ and } \sum_{k\neq f} p_{uk}e^q_{kf} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf}, \tag{15}$$

which can be done in $O(K + |\mathcal{RV}_u|)$ time.

Overall, Equation (13) can be calculated efficiently with the help of $\mathbf{D}^q$ and $\mathbf{E}^q$ caches. As for the remaining terms of $\partial L_{\text{view}}/\partial p_{uf}$, we can apply the same strategy of breaking down and memoizing summations to calculate them in $O(K + |\mathcal{RV}_u|)$ time. Combining with previous solution of $\partial(L_{\text{eALS}} + L_{\text{Reg}})/\partial p_{uf}$, the time complexity for updating p_{uf} is still $O(K + |\mathcal{RV}_u|)$.

4.3.2 Updating Item Factors. Unlike the original L_{eALS} objective function, our proposed L_{view} assumes the pairwise ranking relations among the purchased items, viewed items, and non-viewed

items of the same user. Thus, it does not have the similar counterpart for item factors q_{if}. Focusing on L_{view}, we obtain the derivative *w.r.t.* q_{if} as follows:

$$\frac{\partial L_{\text{view}}}{\partial q_{if}} = 2 \sum_{u \in \mathcal{V}_i} c_i \sum_{j \notin \mathcal{V}_u} p_{uf}^2 \cdot q_{if} \tag{16}$$

$$+ 2 \sum_{u \notin \mathcal{V}_i} \sum_{j \in \mathcal{V}_u} c_j p_{uf}^2 \cdot q_{if} \tag{17}$$

$$+ 2 \sum_{u \in \mathcal{V}_i} \sum_{j \notin \mathcal{V}_u} c_i (\hat{r}_{ui}^f - \hat{r}_{uj}) p_{uf} \tag{18}$$

$$+ 2 \sum_{u \notin \mathcal{V}_i} \sum_{j \in \mathcal{V}_u} c_j (\hat{r}_{ui}^f - \hat{r}_{uj}) p_{uf} \tag{19}$$

$$+ 2\gamma_1 \left[\sum_{u \in \mathcal{V}_i} \sum_{j \in \mathcal{R}_u} c_i p_{uf} - \sum_{u \in \mathcal{R}_i} \sum_{j \in \mathcal{V}_u} c_j p_{uf} \right] \tag{20-21}$$

$$+ 2\gamma_2 \left[\sum_{u \notin \mathcal{R}\mathcal{V}_i} \sum_{j \in \mathcal{V}_u} c_j p_{uf} - \sum_{u \in \mathcal{V}_i} \sum_{j \notin \mathcal{R}\mathcal{V}_u} c_i p_{uf} \right]. \tag{22-23}$$

Again, the bottleneck lies in Equations (18) and (19) terms that contain summations over user-item pairs (u, j). A raw implementation takes $O(|\mathcal{V}_i|N - \sum_{u \in \mathcal{V}_i} |\mathcal{V}_u|)$ and $O(|\mathcal{V}| - \sum_{u \in \mathcal{V}_i} |\mathcal{V}_u|)$ for Equations (18) and (19), respectively. However, unlike updating p_{uf}, item index j are dependent on the user index u. Consequently, the summations cannot be directly broken down into two independent summations over u and j, respectively.

To tackle this challenge, we need to design the new memoization strategy for j-dependent terms. We detail above process by focusing on Equation (19) term in $\partial L_{\text{view}}/\partial q_{if}$, as the rest can be done likewise. We first reformulate it as

$$\begin{aligned} \sum_{u \notin \mathcal{V}_i} \sum_{j \in \mathcal{V}_u} c_j (\hat{r}_{ui}^f - \hat{r}_{uj}) p_{uf} &= \sum_{u=1}^{N} \hat{r}_{ui}^f p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j \\ - \sum_{u \in \mathcal{V}_i} \hat{r}_{ui}^f p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j + \sum_{u \in \mathcal{V}_i} p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj} &- \sum_{u=1}^{N} p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj}. \end{aligned} \tag{24}$$

The first two terms contain a constant, $\sum_{j \in \mathcal{V}_u} c_j$, which we denote as $\tilde{c}_u$ and pre-compute after initialization. Then these two terms can be evaluated as

$$\sum_{u=1}^{N} \hat{r}_{ui}^f p_{uf} \tilde{c}_u \text{ and } \sum_{u \in \mathcal{V}_i} \hat{r}_{ui}^f p_{uf} \tilde{c}_u. \tag{25}$$

A direct computation takes $O(N)$ and $O(|\mathcal{V}_i|)$, respectively. Similar to Equation (14), we define item-independent $\mathbf{E}^p$ cache as $\mathbf{E}^p = \sum_{u=1}^{M} \tilde{c}_u \mathbf{p}_u \mathbf{p}_u^T \in \mathbb{R}^{K \times K}$. By pre-computing and using it in updating the latent factors for all items, $\sum_{u=1}^{N} \hat{r}_{ui}^f p_{uf} \tilde{c}_u$ can be fast calculated in $O(K)$ time.

As for the remaining two terms in Equation (24), the major computation lies in $\sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj}$, i.e., the summation over the predictions of u's viewed items. Therefore, we define the user-dependent

cache $\mathbf{Lv}^r$ as $lv_u^r = \sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj}$ for each user u. Then these two can be further evaluated as

$$\begin{aligned} &\sum_{u \in \mathcal{V}_i} p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj} - \sum_{u=1}^{N} p_{uf} \cdot \sum_{j \in \mathcal{V}_u} c_j \hat{r}_{uj} \\ &= \sum_{u \in \mathcal{V}_i} p_{uf} lv_u^r - \sum_{u=1}^{N} p_{uf} lv_u^r, \end{aligned} \tag{26}$$

where the first costs $O(|\mathcal{V}_i|)$ and the second requires a speed-up. Accordingly, we define the $\mathbf{T}$ cache as $t_f = \sum_{u=1}^{N} p_{uf} lv_u^r$, which has the size of latent factors, K, independent of both u and j. With above pre-defined caches, i.e., $\mathbf{E}^p$, $\mathbf{Lv}^r$, and $\mathbf{T}$, Equation (24) can be calculated in $O(K + |\mathcal{V}_i|)$ time.

However, due to the existence of $\hat{r}_{uj}$ terms, calculating $\mathbf{Lv}^r$ and $\mathbf{T}$ requires both user factors and item factors, which is different from the pre-defined caches such as $\mathbf{E}^p$. Thus, after updating item factor q_{if}, it is necessary to update these two caches before the follow-up of $q_{i,f+1}$, which should maintain the same computational complexity as calculating Equation (24). According to the definition of $\mathbf{Lv}^r$, item factor q_{if} is related to a set of $\{lv_u^r | u \in \mathcal{V}_i\}$, and thus updating $\mathbf{Lv}^r$ takes $O(|\mathcal{V}_i|)$ time. As for $\mathbf{T}$ cache, updating one element t_f also takes $O(|\mathcal{V}_i|)$ time. For calculating the next item factor $q_{i,f+1}$, updating t_{f+1} is enough, taking $O(|\mathcal{V}_i|)$ time.

To summarize, calculating and updating Equation (24) only takes $O(K + |\mathcal{V}_i|)$ time, which also works for the rest terms of $\partial L_{\text{view}} / \partial q_{if}$. Therefore, combining with previous solution of $\partial(L_{\text{eALS}} + L_{\text{Reg}})/\partial q_{if}$, the overall time complexity for updating q_{if} is still $O(K + |\mathcal{R}\mathcal{V}_i|)$.

Algorithm 1 summarizes the accelerated algorithm for our VALS learner. In each iteration, the fth latent factor of u's vector, namely, p_{uf}, is updated in sequence (Line 4-10). More specifically, terms in $\partial(L_{\text{eALS}} + L_{\text{Reg}})/\partial p_{uf}$ are first calculated (Line 6), followed by $\partial L_{\text{view}}/\partial p_{uf}$ (Line 7). Then, p_{uf} can be updated by solving $\partial(L_{\text{eALS}} + L_{\text{Reg}} + L_{\text{view}})/\partial p_{uf} = 0$ (Line 8). After computing all the user factors, item factors can be similarly updated (Lines 12–18). Overall, one PairVALS iteration takes $O((M + N)K^2 + (|\mathcal{R}| + |\mathcal{V}|)K)$ time, which only depends on the number of observed interactions. For convergence, one can either monitor the value of objective function on training set or check the prediction performance on a hold-out validation data.

Note that some previous works [40, 53] have also used the least squared loss $l(t) = (1 - t)^2$ and learning method based on ALS or coordinate descent technique. However, compared to PairVALS, they only solve a sub-problem with only one feedback type and no tunable margin to control pairwise ranking relations. Also, the update of user vectors is embarrassingly parallel, while that of item vectors can influence with each other and thus introduce approximate loss in parallel.

4.4 Generality

In real-world online information systems, the user auxiliary feedbacks are more than just one type. For example, in E-commerce websites, besides users' viewed interactions, other auxiliary interactions like add-to-cart and collect are also available for learning user preference among different items. We will demonstrate how the above designed PointVALS and PairVALS methods can be extended into the general case where multiple types of auxiliary feedback data are available. For clearer representation, we denote them as Pointwise Auxiliary-enhanced ALS (PointAALS) and Pairwise Auxiliary-enhanced ALS (PairAALS), respectively. More specifically, we consider the following two cases: (1) the preference order among different feedback is pre-defined; (2) the preference order is not defined.

In both cases, the PointAALS method can easily integrate multiple feedback by assigning them intermediate label values. First, with the pre-defined preference order, the label value $\hat{r}_{u\bullet}$ for each

ALGORITHM 1: Fast PairVALS Learning algorithm.

Input: $\mathbf{R}$, $\mathbf{V}$, K, λ, $\mathbf{W}$, item confidence vector $\{\mathbf{s}, \mathbf{c}\}$ and margin values $\{\gamma_1, \gamma_2\}$;
Output: Latent feature matrix $\mathbf{P}$ and $\mathbf{Q}$;

```
Randomly initialize P and Q;
while Stopping criteria is not met do
    // Update user factors
    for u ← 1 to M do
        for f ← 1 to K do
            Calculate each term in ∂(L_eALS + L_Reg)/∂p_uf;
            Calculate each term in ∂L_view/∂p_uf;
            Update p_uf by setting ∂(L_eALS+L_Reg+L_view)/∂p_uf = 0;
        end
    end
    // Update item factors
    for i ← 1 to N do
        for f ← 1 to K do
            Calculate each term in ∂(L_eALS + L_Reg)/∂q_if;
            Calculate each term in ∂L_view/∂q_if;
            Update q_if by setting ∂(L_eALS+L_Reg+L_view)/∂q_if = 0;
        end
    end
end
```

type of feedback should be set accordingly. Oppositely, in the second case, a uniform value is set for all auxiliary feedbacks, as learning preference order among them is not the objective. Then, with the speed-up techniques in Section 4.1, the total time complexity for one iteration is still linear to the size of observed data. For example, suppose there exists a relation of "purchase > collect > view" among users' interactions in E-commerce websites, then we can set the labels of the above three interactions as 1, r_1, and r_2, respectively, where $1 > r_1 > r_2$ and both r_1 and r_2 are tunable hyper-parameters. Compared with the plain MF, the biggest difference is that PointAALS is under the framework of whole-data-based MF, which is more effective on implicit feedback data but requires more efficient learning strategy.

As for the PairAALS, we can extend it into the multiple feedback scenario similarly. Assume that there are L types of auxiliary feedback data, denoted as $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_L\}$. For better representation, we further denote the primary feedback (i.e., purchased interactions in above case) as $\mathcal{A}_{L+1}$ and other unobserved data as $\mathcal{A}_0$. Then, for the first case with ascending preference order among $\{\mathcal{A}_l\}$ ($l \in [0, L+1]$), the pairwise ranking objective function can be designed as

$$
\begin{aligned}
&L_{\text{auxiliary}} = \sum_{l=0}^{L} L(\mathcal{A}_l, \mathcal{A}_{l+1}), \\
&\text{where } L(\mathcal{A}_l, \mathcal{A}_{l+1}) = \sum_{\substack{(u,v)\in\mathcal{A}_l \\ (u,i)\in\mathcal{A}_{l+1}}} \left(\gamma_{l,l+1} - (\hat{r}_{ui} - \hat{r}_{uv})\right)^2.
\end{aligned} \tag{27}
$$

Therefore, the view-enhanced objective function in Equation (6), i.e., L_{view} term, is the special case when $L = 1$. As each $L(\mathcal{A}_l, \mathcal{A}_{l+1})$ term models a pairwise ranking relation with the same form of

margin-based loss, we can apply the same strategy of breaking down and memoizing summations to accelerate the learning process, as done in Section 4.3. Thus, the total time complexity for one iteration can be reduced to be linear to the size of observed data. Finally, for the second case where the preference order among different auxiliary feedback, i.e., $\{\mathcal{A}_l\}$ ($l \in [1, L]$), is not defined, we can directly model the pairwise ranking relations of each auxiliary feedback $\mathcal{A}_l$, with respect to both $\mathcal{A}_0$ and $\mathcal{A}_{L+1}$. The corresponding objective function can be designed as

$$L_{\text{auxiliary}} = \sum_{l=1}^{L} L(\mathcal{A}_0, \mathcal{A}_l) + L(\mathcal{A}_l, \mathcal{A}_{L+1}), \tag{28}$$

with the similar time complexity to that in Equation (27). Still taking the relation "purchase > collect > view" as an example, we expect that there exist three margins ($\{\gamma_1, \gamma_2, \gamma_3\}$) between the model predictions of each two adjacent interactions, i.e., purchase-collect, collect-view, and view-else, respectively.

To summarize, our proposed PointVALS and PairVALS methods have a good generality to multiple types of auxiliary feedback data, with both model effectiveness and learning efficiency. However, one limitation is that the number of hyper-parameters increases with more types of feedbacks, which is hard to tune in real applications. One solution is to tune feedback-related hyper-parameters successively. For example, for "purchase > collect > view," we first tune common ones like regularization coefficient and factor size, then those for a specific feedback according to the descending order, with previous hyper-parameters fixed. We find this strategy to obtain good results in experiments. Also, automatic hyper-parameter optimization tools like Optuna [1] can also be considered, which is much more efficient and guaranteed to find fairly good hyper-parameters.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets and Preprocessing. We perform experiments on two real-world datasets of Beibei and Tmall:

Beibei[1]: Beibei is the largest E-commerce platform for maternal and infant products in China. We sample a subset of user interactions that contain views and purchases within the time period from 2017/05/25 to 2017/06/28.

Tmall[2]: Tmall is the largest E-commerce platform in China. To make our results reproducible, we use a public benchmark released in IJCAI-2015 challenge.[3] The time period is from 2014/06/01 to 2014/09/30. Besides users' purchases and views, this dataset also contains a certain number of collected interactions. We include all these three types of user feedback in the experiments to demonstrate the capability of incorporating multiple auxiliary data.

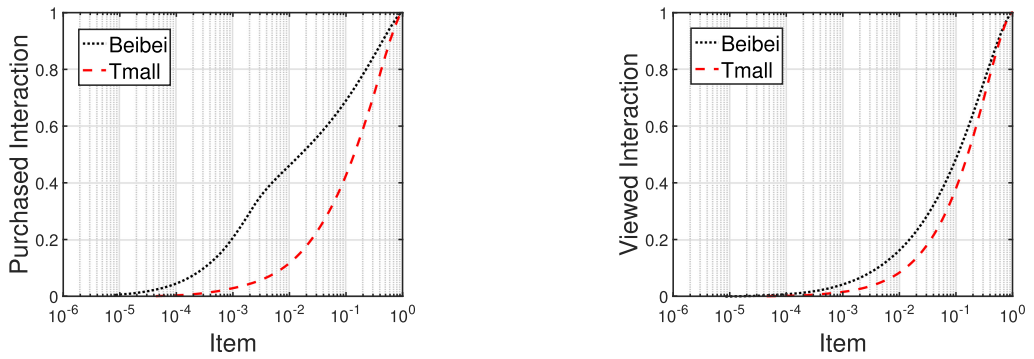
We take three steps for data preprocessing. We first merge the repetitive purchases of the same user and item into one purchase with the earliest timestamp, as we aim to recommend novel items. Next, we filter out users' views (and collects) on those purchased items to avoid information leaking. Finally, we filter out users and items with less than 12 and 16 purchases, respectively, to overcome the high sparsity of the raw datasets. Table 2 summarizes the data statistics. With both primary (purchase) and auxiliary (view and collect) feedback collected, these datasets are sufficient for our research on improving implicit recommender systems.

Observations. The popularity skewness exists in many recommender systems and impacts the performance. Therefore, we investigate the popularity skewness in our data, in terms of item

[1] http://www.beibei.com/.
[2] https://www.tmall.com/.
[3] The dataset is downloaded from https://tianchi.aliyun.com/datalab/dataSet.htm?id=5.

Table 2. Statistics of the Evaluation Datasets

Dataset	Purchase#	View#	Collect#	User#	Item#	Sparsity
Beibei	2,654,467	23,668,454	#	158,907	119,012	99.99%/99.87%
Tmall	160,840	531,640	24,681	12,921	22,570	99.94%/99.82%



(a) purchase (b) view

Fig. 1. Popularity skewness of the Beibei and Tmall datasets.

purchases and views, and show the result in Figures 1(a) and 1(b), respectively. The y-axis represents the ratio of interactions for a given ratio of items on the x-axis, sorted by decreasing popularity. For item purchases, Beibei is the most popularity skewed dataset, where the top-1% of the items accounts for 50% of the purchased interactions, much larger than 10% in Tmall dataset. Such difference in skewness no longer exists in item views, where the top-1% of the items accounts for 16% and 9% of the viewed interactions in Beibei and Tmall, respectively. In summary, users in Beibei are more likely to purchase those popular items, which may affect the performance of personalized recommendation algorithms. On the contrary, users in Tmall are less likely to view those popular items, meaning that there may exist a stronger personal preference in users' views.

Evaluation Methodology. In the evaluation, we adopt the *leave-one-out* protocol [15, 35], where the latest purchase interaction of each user is held out for testing and the models are trained on the remaining data. During each iteration of training process, we randomly select one record for each user from the training data as the validation set and tune hyper-parameters on it. For the metrics, we employ *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) on the ranking of all non-purchased items for a user. We truncate the ranked list at the position of 100 and report the average score of all users.

Baselines. We compare with two types of methods. For the methods that only use primary purchase feedback, we choose:

—**eALS** [15]. This is a state-of-the-art MF method for implicit recommender systems. We tuned the weight of missing data s_i.

—**BPR** [35]. BPR optimizes the MF model with a pairwise ranking loss and learns model parameters with Stochastic Gradient Descent (SGD) method. We tuned the learning rate ε_{BPR}.

For the second type of methods that integrate both purchase and auxiliary feedback, we choose the following methods,

—**RankALS** [40]. This method directly minimizes a ranking objective function without sampling. We adapt it into multiple auxiliary feedback scenario by assigning each type of feedback with a different label value (r_1, r_2).

Table 3. Parameter Exploration for Baselines and Optimal Settings

	Method	Tuning Range	Beibei	Tmall
s_0	eALS	$[1, 2, 4, 8, 16, 32, 64] \times 10^2$	1600	800
ε_{BPR}	(MR/MC-)BPR	$[0.05, 0.1, 0.5, 1, 5] \times 10^{-2}$	0.001	0.01
α	MR-BPR	$[0.125, 0.25, 0.5, 1, 2]$	1	1 (view) 1 (collect)
β_1	MC-BPR	$[0.01, 0.05, 0.07, 0.1, 0.5, 0.7, 1, 5, 7, 10]$	0.05	0.10 (view) 0.70 (collect)
β_2		$[0.01, 0.05, 0.1, 0.5, 1]$	0.5	0.05
$\varepsilon_{\text{MFPR}}$	MFPR	$[0.05, 0.1, 0.5, 1, 5] \times 10^{-2}$	0.001	0.001
r	RankALS	$[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$	#	0.4 (view), 0.6 (collect)
p	NMTR	$[1/10, 1/5, 1/4, 1/3, 1/2, 1]$	$[1/2, 1/2]$	$[1/3, 1/3, 1/3]$

—**MR-BPR** [20]. Applying CMF method to BPR, this method exerts the impact of auxiliary behavior (like views) on predicting purchases with a weight α.

—**MC-BPR** [28]. This method samples both positive and negative instances from viewed items. Two parameters control this process: 1) β_1 denotes relative weight of viewed items when sampling positive instances; 2) β_2 denotes possibility of sampling a viewed item as a negative instance.

—**MFPR** [27]. This is a most recent method that integrates multiple types of implicit feedback. We adapt it to our case by generating training item pairs in the same way as BPR, and use a fixed learning rate $\varepsilon_{\text{MFPR}}$ in SGD.

—**NMTR** [12]. This method incorporates multiple types of user behaviors by predicting each of them in a multi-task learning manner. The overall framework accounts for the cascading relationship among these sub-tasks (e.g., a user must click on a product before purchasing it). The relative preference ranking of different behaviors are decided by the cascading relationship and the learning weight $\{p_i\}$ for each sub-task.

Parameter settings. For the above baselines, we have carefully explored the corresponding parameters and listed the result in Table 3. Note that we uniformly set the weight of missing data as $s_i = s_0/N$, as the effectiveness of popularity-biased weighting strategy is beyond the scope of this article. The score of purchased interactions r_{ui} and its weight ω_{ui} are both uniformly set to 1, which are the suggested values in the eALS implementation. For regularization, we have tuned the best value in $[0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5]$. According to experiment results, we set λ as 0.001 for ALS-based methods and NMTR. As for BPR-based methods, λ is set as 0.01 and 0.1 on Beibei and Tmall, respectively. Since the findings are consistent across the number of latent factors K, we report the results of $K = 32$ only. Note that we only run eALS-based methods (i.e., eALS, PointVALS, and PairVALS) for 200 iterations on two datasets and run other methods except MC-BPR for 1,500 iterations on Beibei dataset, which are enough for them to converge. We find that MC-BPR is hard to converge, requiring about 2,500 iterations on Beibei dataset. RankALS method is quite slow that we are only able to evaluate it on Tmall dataset, with 20 iterations to converge. As for NMTR implemented in tensorflow, we find that runing 10-20 iterations is enough for it to converge.

5.2 Efficiency Evaluation

We first investigate the scalability of our proposed view-enhanced solutions, i.e., PointVALS and PairVALS, whose analytical time complexity should be $O((M + N)K^2 + (|\mathcal{R}| + |\mathcal{V}|)K)$. As the eALS is currently the most efficient method designed for purchase data only, with a time complexity of $O((M + N)K^2 + |\mathcal{R}|K)$, we compare their efficiency empirically based on the experiments on the same machine (Intel Xeon 2.10 GHz CPU, single-thread). The result of actually training time per

Table 4. Training Time per Iteration of Different Methods with Varying Data Size (Ratio) or Number of Factors (K)

(a) Varying data size, fixing $K = 32$

Dataset	Beibei			Tmall			
Ratio	eALS	PointVALS	PairVALS	eALS	PointVALS	PairVALS	RankALS
25%	2 s	11 s	15 s	0.2 s	0.3 s	0.6 s	#
50%	4 s	25 s	35 s	0.2 s	0.6 s	1.0 s	#
75%	6 s	41 s	55 s	0.3 s	0.8 s	1.5 s	#
100%	8 s	54 s	75 s	0.4 s	1.3 s	2.1 s	3,768 s

(b) Varying K

Dataset	Beibei			Tmall		
K	eALS	PointVALS	PairVALS	eALS	PointVALS	PairVALS
16	5 s	29 s	47 s	0.3 s	1.0 s	1.9 s
32	8 s	54 s	75 s	0.4 s	1.3 s	2.1 s
64	14 s	75 s	149 s	1.2 s	2.4 s	4.3 s
128	27 s	154 s	334 s	3.3 s	5.4 s	11.0 s

iteration[4] is shown in Tables 4(a) and 4(b), in terms of the observed data size and number of latent factors, respectively.

As can be seen in Table 4(a), with the increase of observed data size, both PointVALS and PairVALS increase almost linearly, corresponding to our theoretical analysis that the time complexity should be determined by the number of observed data, i.e., $O((|\mathcal{R}| + |\mathcal{V}|)K)$ part. Besides, as PointVALS and PairVALS have the same analytical time complexity, their actual running time are in the same magnitude and the minor difference can be caused by some implementation details, such as the data structures and caches used. Here, for comparison, we also list the training time per iteration of RankALS, which theoretically costs $O((|\mathcal{R}| + |\mathcal{V}|)K^2 + (M + N)K^3)$. It can be observed that RankALS is computationally inefficient even on a rather small dataset (Tmall). Due to the potential huge time cost, we do not evaluate RankALS on Beibei dataset.

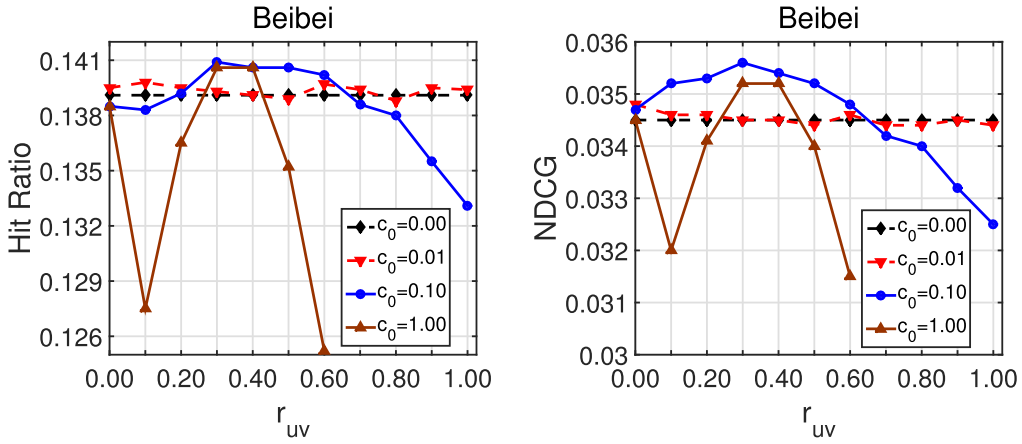
As for the impact of latent factors, result in Table 4(b) does not indicate a linearity to K^2, due to the fact that $(M + N)K$ and $(|\mathcal{R}| + |\mathcal{V}|)$ are in the same magnitude. In summary, by comparing to the eALS, we demonstrate that both PointVALS and PairVALS integrate the additional view data in an efficient way, making these two methods scalable and practicable for large-scale data.

Later, we focus on the recommendation accuracy of the proposed methods. We first study the impact of parameters. Then, we compare our auxiliary-enhanced solution with original eALS method to demonstrate the performance gain. After that, we compare with other methods that also integrate both primary and auxiliary data. Finally, we demonstrate the advantage of whole-data-based learning strategy by comparing with sampling-based variations of our proposed methods.

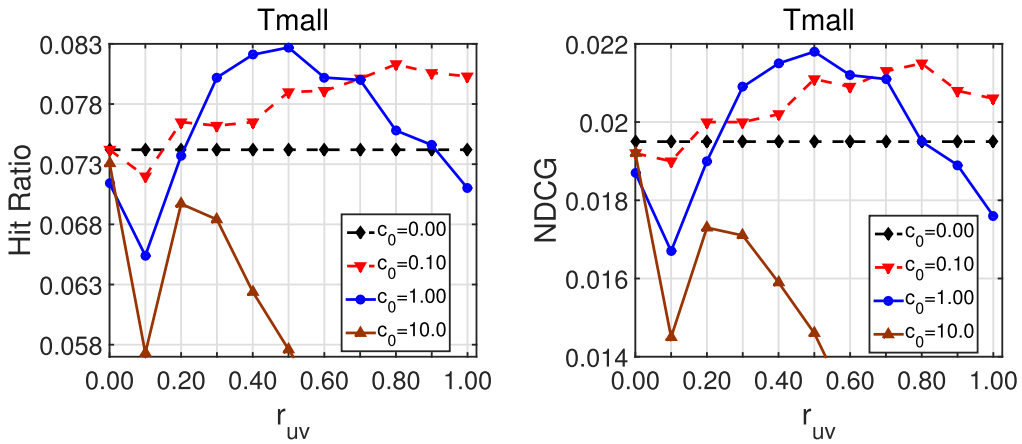
5.3 Hyper-parameter Investigation

In this part, we focus on the hyper-parameter investigation of proposed methods on both Beibei and Tmall. Although two types of auxiliary data are available in Tmall, i.e., users' views and collects, for simplicity, we only detail the process of investigating view-related hyper-parameters.

[4]Obtained by averaging over results of ten times of repeated experiments.



(a) r_{uv} vs. HR (different c_0) (b) r_{uv} vs. NDCG (different c_0)

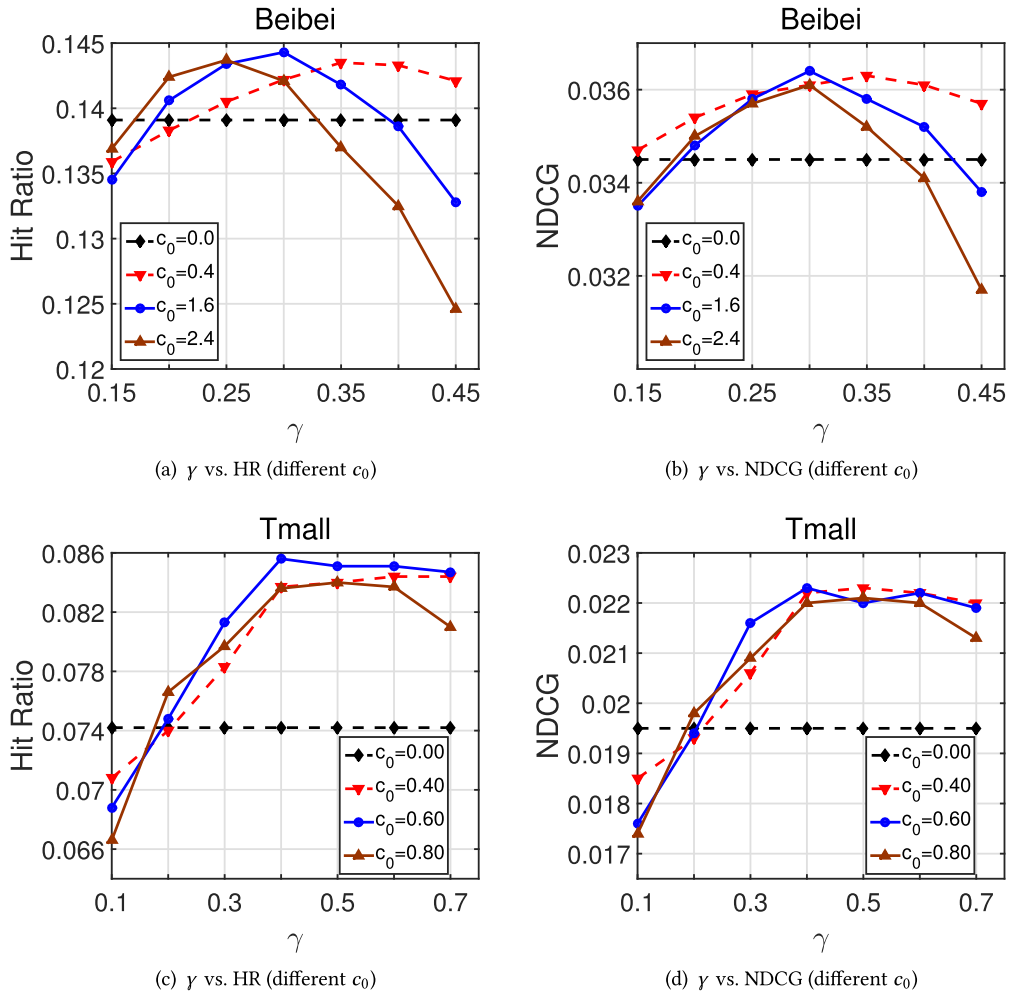


(c) r_{uv} vs. HR (different c_0) (d) r_{uv} vs. NDCG (different c_0)

Fig. 2. Impact of weighting parameters c_0 and label values r_{uv} on PointVALS's performance.

5.3.1 PointVALS. The PointVALS has two parameters: r_{uv}, which is the label value assigned to predictions on viewed items, and c_i that determines the weight of view data. Similar to weight of missing data s_i, we set a uniform weight distribution (i.e., $c_i = c_0/N$) and leave the item-dependent weighting strategy to the future work. To find the best setting for (r_{uv}, c_0), we conduct a grid search over these two parameters and decide the best setting by mean values of both HR and NDCG in last 10 iterations.

Figure 2 plots the prediction accuracy of PointVALS with different r_{uv} and c_0. The actual tuning range of c_0 is [0,0.005,0.01,0.05,0.1,0.5,1] (Beibei) and [0,0.05,0.1,0.5,1,5,10]] (Tmall), respectively, while we only visualize four of them from better illustration. First, we study the impact of label value r_{uv}. For Beibei (Figures 2(a) and 2(b)), we observe that the best r_{uv} values are between 0.2 and 0.4 under different values of weight c_0. When r_{uv} is chosen smaller or larger, the performance is degraded and even worse than that without considering view data, i.e., $c_0 = 0$. As users' viewed interactions indicate an intermediate preference level, it is reasonable to set r_{uv} between that of non-viewed interaction (i.e., 0) and purchased interaction (i.e., 1). Similarly, for Tmall (Figures 2(c)



(a) γ vs. HR (different c_0)

(b) γ vs. NDCG (different c_0)

(c) γ vs. HR (different c_0)

(d) γ vs. NDCG (different c_0)

Fig. 3. Impact of weighting parameters c_0 and margin values γ on PairVALS's performance.

and 2(d)), the best r_{uv} values is still in $(0, 1)$, yet bigger than those for Beibei. Specifically, it is 0.8 when $c_0 = 0.1$ and 0.5 when $c_0 = 1.0$, respectively. As for the impact of c_0, both results for Beibei and Tmall highlight the importance of weighting view data appropriately, which should be neither under-weighted ($c_0 = 0.01$, Beibei) nor over-weighted ($c_0 = 10$, Tmall).

According to above parameter exploration, we fix r_{uv} and c_0 according to the best performance evaluated by both HR and NDCG, i.e., $r_{uv} = 0.3, c_0 = 0.1$ for Beibei and $r_{uv} = 0.5, c_0 = 1.0$ for Tmall.

5.3.2 PairVALS. Similar to the PointVALS, the PairVALS also has two parameters: $\{\gamma_1, \gamma_2\}$, which are the margin values to control pairwise ranking between viewed items and other items, and c_i that determines the weight of view data. Without loss of generality, we set margin values $\{\gamma_1, \gamma_2\}$ uniformly as $\gamma_1 = \gamma_2 = \gamma$. Similarly, we also set $c_i = c_0/N$. The grid search results over these two parameters are reported in Figure 3. The actual tuning range of c_0 is [0,0.4,0.8,1.6,2.4,3.2] (Beibei) and [0,0.2,0.4,0.6,0.8,1] (Tmall), respectively, while we only visualize four of them from better illustration.

First, we study the impact of margin value γ. For Beibei (Figures 3(a) and 3(b)), we observe that the best γ values are between 0.25 and 0.35 under different values of weight c_0. Since the prediction is optimized to indicate the preference level, this highlights the necessity of using an appropriate margin γ to control the preference level of viewed interactions, which is lower than purchased ones but higher than non-viewed ones. Surprisingly, for Tmall (Figures 3(c) and 3(d)), we observe that a relative large margin value ($\lambda = 0.4 \sim 0.7$) achieves better performance. According to our definition in Equation (6), purchased items are optimized to be predicted as 1, while optimized predictions for non-viewed items are 0. When the margin γ between a viewed item and a purchased/non-viewed item is large, it is more likely for a viewed item to be predicted outside the (0, 1), while the optimized prediction should be inside. Therefore, a large γ observed on Tmall dataset indicates higher possibility that the preference level of viewed interactions is close to purchased ones or non-viewed ones. Then, we investigate the best setting of weight c_0 by comparing the peaks of different curves in each sub-figure. For Beibei (Figures 3(a) and 3(b)), the peak performance is achieved when c_0 is 1.6; similarly for Tmall (Figures 3(c) and 3(d)), the optimal c_0 is 0.6. When c_0 becomes smaller or too large, the performance decreases in both cases, indicating the necessity to account for viewed interactions carefully.

Finally, for PairVALS, we fix γ and c_0 according to the best performance evaluated by both HR and NDCG, i.e., $\gamma = 0.3, c_0 = 1.6$ for Beibei and $\gamma = 0.4, c_0 = 0.6$ for Tmall.

5.3.3 Tuning with Multiple Types of Auxiliary Data. When coping with more than one auxiliary feedbacks, (e.g., view and collect feedbacks in Tmall), we tune the hyper-parameters of each feedback in sequential instead of a single grid search, to speedup the process. More specifically, when tuning collect-related hyper-parameters, we fix the previous tuned view-related hyper-parameters and only investigate $\{c_0', r_{uv}'\}$ (PointAALS) or $\{c_0', \gamma'\}$ (PairAALS). c_0' and r_{uv}' denotes the weight and the pointwise label value of collect data, while γ' represents the expected margin between the model predications of collected and viewed items. Finally, we find the best setting as $c_0' = 1, r_{uv}' = 0.6$ and $c_0' = 0.01, \gamma' = 0.2$, respectively. To ease above time-consuming tuning process, one can also consider other automatic hyper-parameter optimization tools like Optuna [1].

5.4 Performance Gain of Auxiliary Data

Table 5 displays the performance of our PointVALS and PairVALS methods compared with eALS, *w.r.t.* HR@100 and NDCG@100. We report both the mean values and standard variances of ten repeated experiments. To highlight the importance of modeling viewing behaviors as the intermediate feedback rather than a positive signal, we also compare with two eALS methods: (1) uses auxiliary data only and (2) uses both primary and auxiliary data equivalently. i.e., predicting future purchases based on previous views. The former can be easily implemented by setting purchase weight values $\omega_{ui} = 0$ and view label values $r_{uv} = 1$, while the latter refer to the case when $c_0 = 1$ and $r_{uv} = 1$.

It is noteworthy that, compared with using the purchase data only, combining both the purchase and view data together to train an eALS model will degrade the performance by 38.57% (Beibei) and 3.57% (Tmall) on average, which demonstrates that auxiliary behavior cannot be directly considered as the positive signal. Clearly, after integrating view data as the intermediate feedback, our proposed methods significantly outperforms the eALS (purchase). For Beibei, the relative improvements with PointVALS in terms of HR and NDCG are 1.36% and 2.89%, respectively. As the PairVALS models the pairwise ranking relations among purchased, viewed and non-viewed interactions, it achieves the improvements of 3.52% and 4.91%, which further outperform those of PointVALS by 2.12% and 1.97%, respectively. Similarly, for Tmall, the relative improvements of

Table 5. Performance Improvement After Integrating Auxiliary Data

(a) Beibei

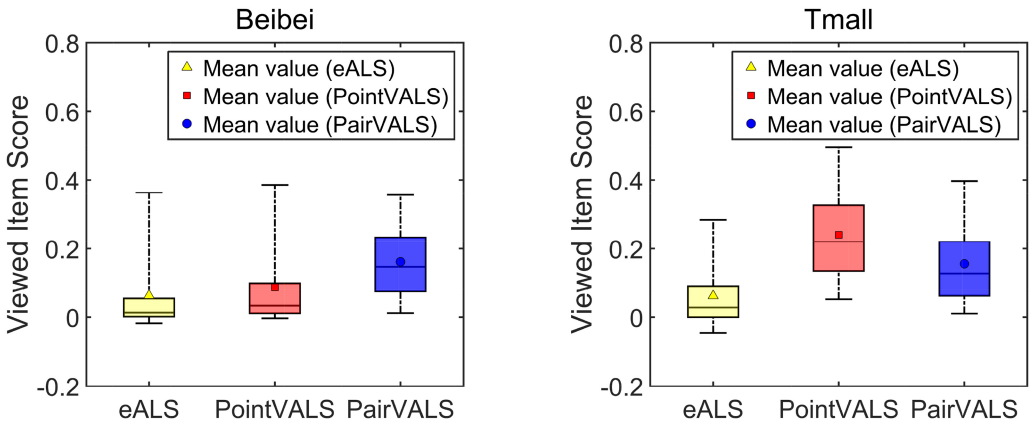
Data Used	Methods	HR	NDCG	ΔHR	ΔNDCG
purchase	**eALS**	0.1393±0.0005	0.0346±0.0001	-	-
view	**eALS**	0.0636±0.0003	0.0141±0.0001	−54.34%	−59.25%
purchase + view	**eALS**	0.0860±0.0005	0.0208±0.0001	−38.26%	−39.88%
	PointVALS	0.1412±0.0002	0.0356±0.0001	+1.36%	+2.89%
	PairVALS	0.1442±0.0003	0.0363±0.0001	+3.52%	+4.91%

(b) Tmall

Data Used	Methods	HR	NDCG	ΔHR	ΔNDCG
purchase	**eALS**	0.0742±0.0004	0.0194±0.0002	-	-
view	**eALS**	0.0391±0.0002	0.0085±0.0002	−47.30%	−56.19%
purchase + view	**eALS**	0.0712±0.0004	0.0188±0.0002	−4.04%	−3.09%
	PointVALS	0.0836±0.0008	0.0219±0.0001	+12.67%	+12.89%
	PairVALS	0.0847±0.0008	0.0222±0.0001	+14.15%	+14.43%
purchase + view + collect	**eALS**	0.0735±0.0006	0.0195±0.0002	−0.94%	−0.52%
	PointAALS	0.0871±0.0005	0.0228±0.0001	+17.39%	+17.53%
	PairAALS	0.0867±0.0007	0.0227±0.0003	+16.85%	+17.01%

proposed methods are about 12.67%–14.43%, in terms of both HR and NDCG. This indicates that users' viewing behaviors in Tmall are much more valuable for learning a personal and accurate preference order among different items, corresponding to our previous observation that users in Tmall are less likely to view those popular item. Moreover, when we further integrate the collect data in Tmall, the relative improvements increase to 16.85%–17.53%. This observation not only indicates the value of collect data in terms of learning user preference but also highlights the generality of our proposed methods when multiple types of auxiliary feedback are available. Last but not least, the proposed pairwise method outperforms the pointwise one with single auxiliary feedback, while the opposite is observed with two auxiliary feedbacks. We believe this difference comes from the fact that collect data is quite sparse in tmall, about two collects per user. Considering the insufficient pairwise learning between collecting behaviors and other two behaviors, i.e., purchasing and viewing behaviors, it may be more effective to directly assign a uniform label value for all collected interactions.

To further clarify the difference between these methods, we compare the predictions over viewed items between two datasets, using eALS, PointVALS and PairVALS method to train the model. Figure 4 plots the distribution quantiles (5%, 25%, 50%, 75%, 95%) and means of viewed item scores, where the results of three methods are presented together. Clearly, the viewed items are both predicted to have a near 0 score on two datasets when trained with eALS, as they are considered as negative instances. However, when comparing PointVALS and PairVALS, we observe that the distribution of predictions with PairVALS is more consistent between two dataset, where median and mean value are $(0.15, 0.16)$ for Beibei (Figure 4(a)) and $(0.13, 0.16)$ for Tmall (Figure 4(b)), respectively. On contrary, with PointVALS, the prediction values are significantly different between Beibei and Tmall. The corresponding median and mean value are $(0.03, 0.09)$ and $(0.25, 0.27)$, respectively. The above distinct observations demonstrate the superiority of PairVALS method, i.e., the consistency of learned prediction values over viewed items.



(a) $r_{uv} = 0.3, \gamma = 0.3$ (b) $r_{uv} = 0.5, \gamma = 0.4$

Fig. 4. eALS versus PointVALS and PairVALS, in terms of quantiles (5%, 25%, 50%, 75%, 95%) and means of the predictions over viewed items.

5.5 Performance Comparison

After investigating performance improvement of considering auxiliary behaviors, we further investigate *Do our proposed methods outperform state-of-art auxiliary-enhanced recommender systems.*

First, except RankALS and NMTR that converge within 10–20 iterations, We show the prediction accuracy of other methods during the training process in Figure 5. In terms of the number of iterations, SGD-based methods will converge slower compared to eALS-based methods, which is reasonable considering that the latter optimize their parameters based on the whole data in each iteration. Besides, when comparing between iterations, we observe that SGD-based methods are less stable than those eALS-based methods. We notice that the former first show an unusual spike in early iterations and then suffer from a performance degradation with more iterations on Beibei dataset, which might be caused by some regularities in the data. For example, Beibei dataset is highly popularity-skewed—the top-1% items contributed almost 50% of purchases, as illustrated in Figure 1(a). This may cause unstable performance, because these popular items are ranked high in early iterations.

In the following, we focus on comparing the performance metrics and report both the mean values and standard variances of ten repeated experiments in Table 6. Note that we do not evaluate RankALS on Beibei due to its infeasibility *w.r.t.* time consumption. For Tmall with both viewing and collecting behaviors as auxiliary data, we do not list the performance of MR-BPR and MFPR as they are not competitive in previous experiments. According to Table 6, we have the following three key observations. First, we see that proposed PointVALS and PairVALS achieve the best performance after convergence. All improvements are statistically significant evidenced by the one-sample paired t-test ($p < 0.01$). The best baseline is MC-BPR. Except for a close HR metric on Beibei, VALS outperforms it by a large margin (on average, the relative improvement for Beibei and Tmall is 6.43% and 6.75%). Compared with MC-BPR, VALS mainly benefits from (1) the objective function that explicitly learns user preference from auxiliary data and (2) the whole-data-based strategy of handling missing data. Third, MR-BPR and MFPR achieve higher performance over the vanilla BPR that only leverages purchase data, while the relative improvement is quite insignificant when compared with MC-BPR and VALS. This highlights the necessity of exploiting different preference levels between purchase and view data, which is lacked in MR-BPR and MFPR. Similarly, NMTR also performs less competitive even though it follows a multi-task learning manner to

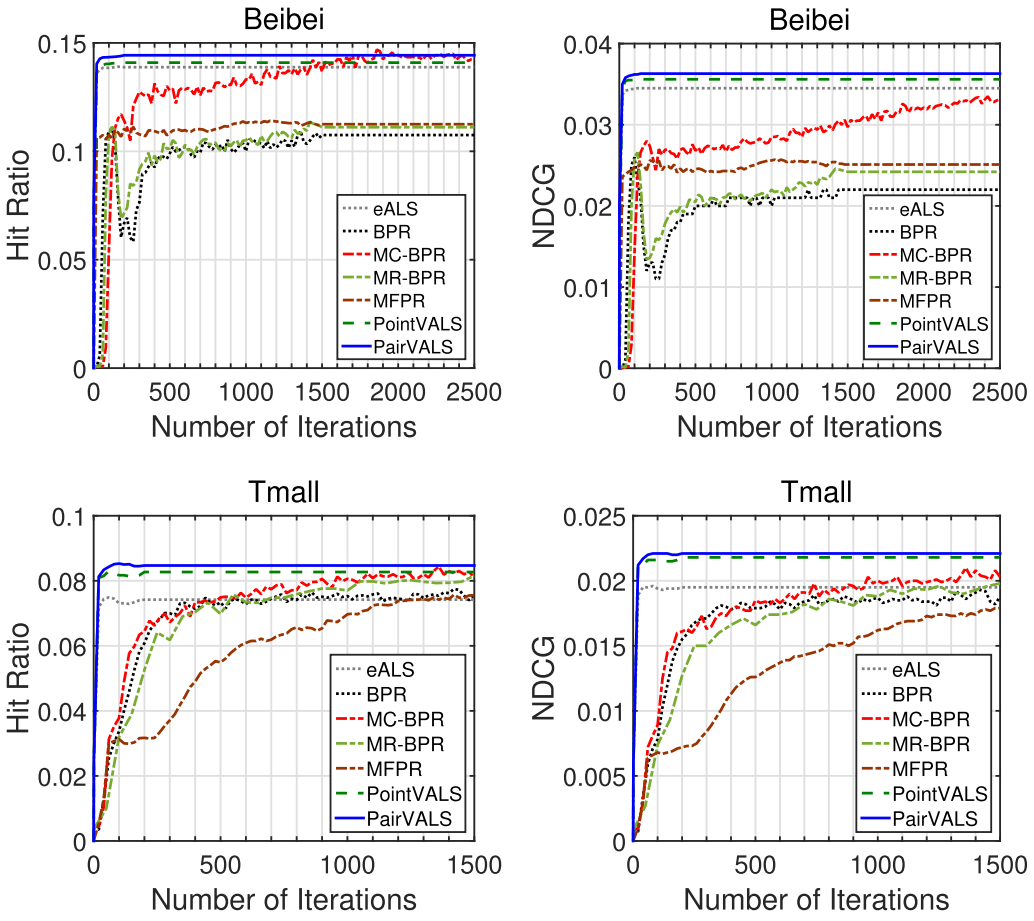


Fig. 5. Performance of PointVALS and PairVALS compared with other baseline methods in each iteration (only purchased and viewed interactions are considered).

capture the cascading relationship among users' views, collects and purchases. This demonstrates the importance of explicitly distinguishing the preference levels reflected by different behaviors. Finally, VALS maintains both accuracy and fidelity, which is an inherent advantage of using whole-data-based learning strategy. Comparatively, although MC-BPR outperforms the vanilla BPR on the Beibei dataset evaluated by both metrics, we find it obtains a higher HR but a lower NDCG score when compared to eALS that does not integrate view data (i.e., 0.0323 vs. 0.0346). It means that whole-data-based strategy is a better candidate compared to sampling-based one when improving implicit recommender systems. However, for RankALS that also adopts the whole-data-based strategy, we do not observe the similar performance as other eALS-based methods, indicating that it is not suitable for incorporating multiple types of user feedback.

Practical recommender systems typically have two stages: (1) candidate selection that selects hundreds of items that might be of interest to a user, and (2) ranking that re-ranks the candidates to show top a few results. Since we only use the interaction data instead of other side features in proposed methods, it is more suitable for them to be applied in the candidate selection stage that requires a high recall. Therefore, evaluation with a large K of hundreds is suitable, and we set the K as 100 in previous experiments. As for the ranking stage, we also evaluate the performance on Tmall

Table 6. Performance Comparison between Baseline Methods

(a) Beibei

Data Used	Methods	HR	NDCG	ΔHR	ΔNDCG
purchase only	**eALS**	0.1393±0.0005	0.0346±0.0001	+3.52%	+ 4.91%
	BPR	0.1065±0.0020	0.0217±0.0005	+35.40%	+67.28%
purchase + view	**MR-BPR**	0.1123±0.0010	0.0244±0.0002	+28.41%	+48.77%
	MFPR	0.1132±0.0006	0.0254±0.0002	+27.39%	+42.91%
	MC-BPR	0.1435±0.0014	0.0323±0.0007	+0.49%	+12.38%
	NMTR	0.1161±0.0004	0.0280±0.0002	+24.20%	+29.64%
proposed	**PointVALS**	0.1412±0.0002	0.0356±0.0001	+2.12%	+1.97%
	PairVALS	**0.1442**±0.0003	**0.0363**±0.0001	-	-

(b) Tmall

Data Used	Methods	HR	NDCG	ΔHR	ΔNDCG
purchase only	**eALS**	0.0742±0.0004	0.0194±0.0002	+17.39%	+17.53%
	BPR	0.0759±0.0010	0.0189±0.0004	+14.76%	+20.63%
purchase + view+collect	**MC-BPR**	0.0830±0.0018	0.0210±0.0003	+4.94%	+8.57%
	NMTR	0.0785±0.0008	0.0192±0.0001	+10.96%	+18.75%
	RankALS	0.0676±0.0005	0.0164±0.0001	+28.85%	+39.02%
proposed	**PointAALS**	**0.0871**±0.0005	**0.0228**±0.0001	-	-
	PairAALS	0.0867±0.0007	0.0227±0.0003	+0.46%	+0.44%

The best results are bold and the second to the best are underlined. ΔHR/ΔNDCG are calculated in terms of the best result.

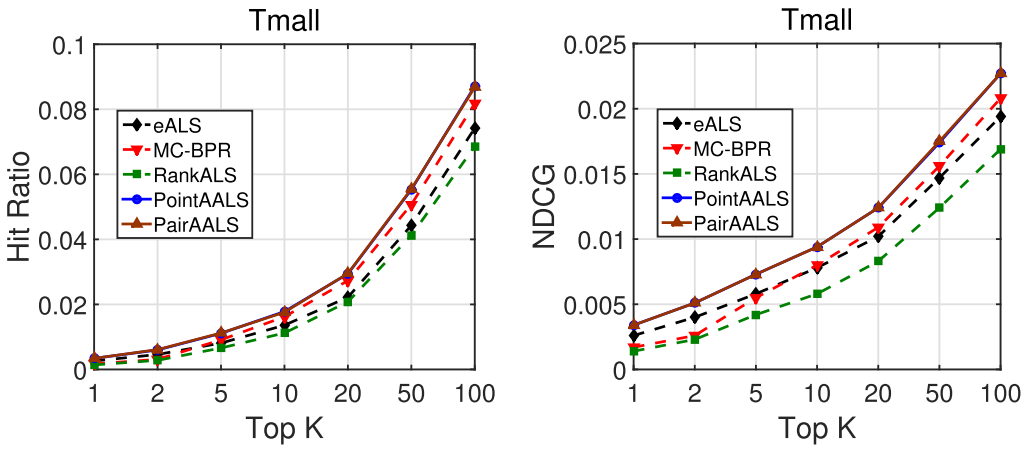


Fig. 6. Performance comparison in terms of HR and NDCG evaluated by different values of K.

w.r.t. HR@K / NDCG@K ($K \in \{1, 2, 5, 10, 20, 50, 100\}$) in Figure 6. Similar to the previous result, our proposed PointVALS and PairVALS outperform other state-of-the-art baselines in all cases.

5.6 The Advantage of Whole-data-based Learning Strategy

In this part, we further demonstrate the advantage of whole-data-based learning strategy in VALS methods, by comparing with a PairVALS variant that optimizes the same objective function but

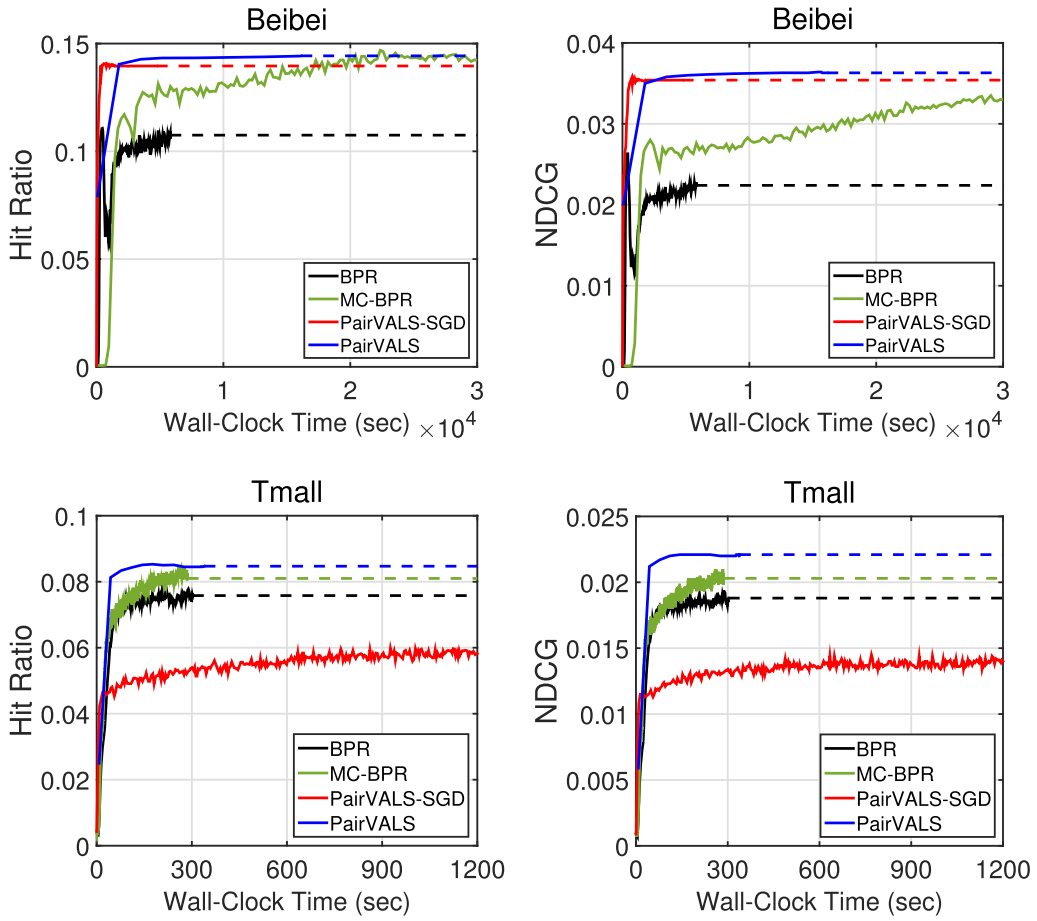


Fig. 7. Performance of PairVALS and PairVALS-SGD compared with other baseline methods in wall-clock time.

adopts a sampling-based learning strategy, i.e., SGD learning algorithm. Figure 7 illustrates the performance curve of PairVALS, its sampling-based variant (denoted as PairVALS-SGD) and other two sampling-based methods in wall-clock time. As different methods converge at different time, we add dotted line in the figure for better comparison. Surprisingly, we observe that PairVALS not only performs best but also requires less time for convergence. As for PairVALS-SGD, it performs quite competitively in Beibei, outperforming other sampling-based methods, but performs the worst in Tmall. Therefore, above results indicate that whole-data-based learning strategy is essential for better incorporating users' preference signal encoded in rich feedback data.

Generally, efficient learning from the whole data requires a least-square-based loss and a prediction model satisfying "K-separable" property [3], such as MF, FM, or even Tucker Decomposition. Similar idea has been proposed in wide areas, including training implicit recommender systems with batch gradient descent learner [50], efficiently learning from non-displayed events in counterfactual CTR prediction [49] and accelerating the batch learning of a NN-based recommender system that has a MF-based prediction layer [6].

Summary: Through extensive experiments on two real-world datasets, we have demonstrated both efficiency and efficacy of our proposed PointVALS and PairVALS models. These two models

integrate the additional auxiliary data in an efficient way, with a time cost that is empirically linear to the data size. Moreover, with much less iterations required in training process, the overall training time is significantly shorter than other sampling-based methods. In terms of recommendation performance, both PointVALS and PairVALS solve the difficulty of learning user preference from auxiliary data, which can result in a poor performance by directly considering them as positive labels. Therefore, our proposed methods outperform state-of-the-art auxiliary-enhanced methods by a large margin, about 6.43%~6.75%.

6 CONCLUSION

We study the problem of improving implicit recommender systems by integrating both primary and auxiliary data. Based on the state-of-the-art eALS method, we model user's auxiliary interactions as an intermediate feedback between primary and other non-observed interactions. Besides the intuitive way of modeling through pointwise regression, we also propose a pairwise ranking model, which is much more flexible but causes the efficiency problem in optimization. To address this key challenge, we further develop a fast-learning algorithm, which efficiently learns parameters from the whole data instead of sampling negative instances. With these designs, our proposed auxiliary-enhanced eALS methods not only achieve higher accuracy but also become practical for large-scale data. Moreover, it can be extended into more general cases where multiple types of auxiliary feedback data are available.

This work has focused on the collaborative filtering setting, which only leverages the feedback data and is mostly used in the candidate selection stage of industrial recommender systems [45]. In future, we will focus more on the ranking stage [30, 57], integrating auxiliary data into generic feature-based models, such as the expressive neural factorization machines [13] and the more explainable tree-enhanced embedding model [44]. Since our MF-based methods are generally used to learn users' long-term preference, we can also focus on modeling users' short-term preference, explicitly integrating the pairwise ranking relations among multiple types of feedback data, which is not considered in recent works on using multi-behavior data for next-item recommendation [21, 24]. Moreover, we also plan to investigate the impact of auxiliary feedback in location recommendation problems [2, 11, 34, 37].

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the KDD*. 2623–2631.
[2] Mohammad Aliannejadi and Fabio Crestani. 2018. Personalized context-aware point of interest recommendation. *ACM Trans. Info. Syst.* 36, 4 (2018), 45.
[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the WWW*. 1341–1350.
[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the NIPS*. 2787–2795.
[5] Cheng Cao, Hancheng Ge, Haokai Lu, Xia Hu, and James Caverlee. 2017. What are you known for?: Learning user topical profiles with implicit and explicit footprints. In *Proceedings of the SIGIR*. ACM, 743–752.
[6] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *Proceedings of the SIGIR*. 225–234.
[7] Jian Cheng, Ting Yuan, Jinqiao Wang, and Hanqing Lu. 2014. Group latent factor model for recommendation with multiple user behaviors. In *Proceedings of the SIGIR*. 995–998.
[8] Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. 2015. Dynamic matrix factorization with priors on unknown values. In *Proceedings of the KDD*. 189–198.
[9] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced negative sampling for recommendation with exposure data. In *Proceedings of the IJCAI*. 2230–2236.
[10] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving implicit recommender systems with view data. In *Proceedings of the IJCAI*. 3343–3349.

[11] Jingtao Ding, Guanghui Yu, Yong Li, Depeng Jin, and Hui Gao. 2019. Learning from hometown and current city: Cross-city POI recommendation via interest drift and transfer learning. *Proc. ACM Interact. Mobile Wear. Ubiq. Technol.* 3, 4 (2019), 131.
[12] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *Proceedings of the ICDE*. 1554–1557.
[13] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the SIGIR*. 355–364.
[14] Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014. Predicting the popularity of web 2.0 items based on user comments. In *Proceedings of the SIGIR*. 233–242.
[15] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the SIGIR*. 549–558.
[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the ICDM*. 263–272.
[17] Saikishore Kalloori, Tianyu Li, and Francesco Ricci. 2019. Item recommendation by combining relative and absolute feedback data. In *Proceedings of the SIGIR*. ACM, 933–936.
[18] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the KDD*. 426–434.
[19] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.
[20] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the WSDM*. 173–182.
[21] Duc Trong Le, Hady Wirawan Lauw, and Yuan Fang. 2018. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *Proceedings of the IJCAI*. 3414–3420.
[22] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *Proceedings of the WWW*. 85–96.
[23] Lukas Lerche and Dietmar Jannach. 2014. Using graded implicit feedback for bayesian personalized ranking. In *Proceedings of the RecSys*. 353–356.
[24] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the KDD*. 1734–1743.
[25] Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. 2018. Scalable content-aware collaborative filtering for location recommendation. *IEEE Trans. Knowl. Data Eng.* 30, 6 (2018), 1122–1135.
[26] Defu Lian, Kai Zheng, Yong Ge, Longbing Cao, Enhong Chen, and Xing Xie. 2018. GeoMF++: Scalable location recommendation via joint geographical modeling and matrix factorization. *ACM Trans. Info. Syst.* 36, 3 (2018), 33.
[27] Jian Liu, Chuan Shi, Binbin Hu, Shenghua Liu, and S. Yu Philip. 2017. Personalized ranking recommendation via integrating multiple feedbacks. In *Proceedings of the PAKDD*. 131–143.
[28] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the RecSys*. 361–364.
[29] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2019. Top-N recommendation with multi-channel positive feedback using factorization machines. *ACM Trans. Info. Syst.* 37, 2 (2019), 15.
[30] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *Proceedings of the SIGIR*. 1137–1140.
[31] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the ICDM*. 502–511.
[32] Weike Pan, Qiang Yang, Wanling Cai, Yaofeng Chen, Qing Zhang, Xiaogang Peng, and Zhong Ming. 2019. Transfer to rank for heterogeneous one-class collaborative filtering. *ACM Trans. Info. Syst.* 37, 1 (2019), 10.
[33] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. 2015. Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowl.-Based Syst.* 73 (2015), 173–180.
[34] Tieyun Qian, Bei Liu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2019. Spatiotemporal representation learning for translation-based POI recommendation. *ACM Trans. Info. Syst.* 37, 2 (2019), 18.
[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the UAI*. 452–461.
[36] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the SIGIR*. 635–644.
[37] Lei Shi, Wayne Xin Zhao, and Yi-Dong Shen. 2017. Local representative-based matrix factorization for cold-start recommendation. *ACM Trans. Info. Syst.* 36, 2 (2017), 22.
[38] Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the KDD*. 650–658.

[39] Nathan Srebro, Jason Rennie, and Tommi S. Jaakkola. 2005. Maximum-margin matrix factorization. In *Proceedings of the NIPS*. 1329–1336.
[40] Gábor Takács and Domonkos Tikk. 2012. Alternating least squares for personalized ranking. In *Proceedings of the RecSys*. 83–90.
[41] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. 2016. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the KDD*. 283–292.
[42] Maksims Volkovs and Guang Wei Yu. 2015. Effective latent models for binary feedback in recommender systems. In *Proceedings of the SIGIR*. 313–322.
[43] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the RecSys*. ACM, 86–94.
[44] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the WWW*. 1543–1552.
[45] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in E-commerce. In *Proceedings of the WSDM*. 619–627.
[46] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, Bolong Zheng, and Xiangyang Luo. 2019. A context-aware user-item representation learning for item recommendation. *ACM Trans. Info. Syst.* 37, 2 (2019), 22.
[47] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for Top-N recommendation. *ACM Trans. Info. Syst.* 37, 3 (2019), 33.
[48] Chunfeng Yang, Huan Yan, Donghan Yu, Yong Li, and Dah Ming Chiu. 2017. Multi-site user behavior modeling and its application in video recommendation. In *Proceedings of the SIGIR*. 175–184.
[49] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving ad click prediction by considering non-displayed events. In *Proceedings of the CIKM*. 329–338.
[50] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Chua Tat-Seng, and Joemon M. Jose. 2018. fBGD: Learning embeddings from positive unlabeled data with BGD. In *Proceedings of the UAI*.
[51] Ting Yuan, Jian Cheng, Xi Zhang, Shuang Qiu, Hanqing Lu, et al. 2014. Recommendation by mining multiple user behaviors with group sparsity. In *Proceedings of the AAAI*. 222–228.
[52] Wei Zhang and Jianyong Wang. 2015. Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. In *Proceedings of the CIKM*. 1221–1230.
[53] Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *Proceedings of the AAAI*. 1669–1675.
[54] Tong Zhao, Julian McAuley, and Irwin King. 2015. Improving latent factor models via personalized feature projection for one class recommendation. In *Proceedings of the CIKM*. 821–830.
[55] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H. Chi. 2015. Improving user topic interest profiles by behavior factorization. In *Proceedings of the WWW*. 1406–1416.
[56] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. 2019. Collaborative metric learning with memory network for multi-relational recommender systems. In *Proceedings of the IJCAI*. 4454–4460.
[57] Yu Zhu, Junxiong Zhu, Jie Hou, Yongliang Li, Beidou Wang, Ziyu Guan, and Deng Cai. 2018. A brand-level ranking system with the customized attention-GRU model. In *Proceedings of the IJCAI*. 3947–3953.

Received March 2019; revised September 2019; accepted November 2019