Improving Implicit Recommender Systems with View Data

Jingtao Ding[1], **Guanghui Yu**[1], **Xiangnan He**[2], **Yuhan Quan**[1],
Yong Li[1], **Tat-Seng Chua**[2], **Depeng Jin**[1] and **Jiajie Yu**[3]
[1] Beijing National Research Center for Information Science and Technology (BNRist),
Department of Electronic Engineering, Tsinghua University
[2] School of Computing, National University of Singapore
[3] Beibei Inc
liyong07@tsinghua.edu.cn

Abstract

Most existing recommender systems leverage the primary feedback data only, such as the purchase records in E-commerce. In this work, we additionally integrate view data into implicit feedback based recommender systems (dubbed as *Implicit Recommender Systems*). We propose to model the pairwise ranking relations among purchased, viewed, and non-viewed interactions, being more effective and flexible than typical pointwise matrix factorization (MF) methods. However, such a pairwise formulation poses efficiency challenges 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 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 $10\% \sim 28.4\%$. Our implementation is available at: https://github.com/dingjingtao/View_enhanced_ALS.

1 Introduction

Recent research on recommendation has shifted from explicit ratings [Koren, 2010] to implicit feedback, such as purchases, clicks, and watches [Bayer *et al.*, 2017]. Distinct from explicit feedback like ratings, in implicit feedback data, the negative signal about user preference over items is naturally scarce, resulting in a one-class learning problem [Pan *et al.*, 2008; Zhao *et al.*, 2015a]. 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 [He *et al.*, 2016], 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 [Rendle *et al.*, 2009] that models partial missing data only.

In an online information system, in addition to the primary feedback data that is directly related with the business KPI, there are also other kinds of user feedback data available [Pan *et al.*, 2015; Tang *et al.*, 2016; Ding *et al.*, 2018]. 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. In this work, we aim to integrate the valuable view data into the state-of-the-art eALS method, so to enhance the performance of implicit recommender systems.

Nevertheless, it is non-trivial to integrate such intermediate feedback into 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. While an intuitive solution to assign the view interactions with an "intermediate label", *i.e.,* a value between 0 and 1, it is difficult to set a proper value for each interaction. Setting a uniform value for all intermediate interactions oversimplifies the problem, which is sub-optimal or may even adversely degrade the performance if the value is set improperly.

In this work, we make a novel technical contribution in integrating intermediate feedback into eALS. Instead of assigning a specific label value to do point-wise learning, we propose a new solution that models the relative preference orders among different interactions. Specifically, to encode the twofold semantics of view data, 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 point-wise regression over them become unaffordable [He *et al.*, 2016], not to mention the pairwise comparisons between viewed and non-viewed interactions. 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 designs of memoization strategies.

We summarize the contributions of the paper as follows.

1. We improve implicit recommender systems by incorporating view data, proposing a View-enhanced eALS (VALS) method that models the pairwise relations among purchased, viewed, and non-viewed interactions.
2. We propose a fast algorithm that solves the challenging VALS problem with a controllable time complexity that is determined by the number of observed interactions only.
3. We conduct extensive experiments on two real datasets, demonstrating that our method outperforms state-of-the-art view-aware recommender systems by a large margin.

2 Related Work

To improve implicit recommender systems with multiple feedback, two types of methods have been proposed.

Model-based. A typical method of this type is collective matrix factorization (CMF), which performs multiple relational learning by sharing information between models of different feedback [Singh and Gordon, 2008; Cheng *et al.*, 2014; Yuan *et al.*, 2014]. While CMF is originated for explicit rating prediction, it has been extended for implicit recommender systems as well [Krohn-Grimberghe *et al.*, 2012; Zhao *et al.*, 2015b; Cao *et al.*, 2017]. 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. In contrast, our VALS method learns the same user-item relation to indicate relative preference order among purchase and view data, which is more effective.

Learning-based. Several other methods integrate multiple types of feedback in the negative sampler of BPR [Lerche and Jannach, 2014]. 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 [Loni *et al.*, 2016]. A recent proposal also performs biased sampling from viewed but non-purchased items and observes significant performance improvements [Ding *et al.*, 2018]. However, these BPR-based solutions still suffer from the shortcomings of sampling-based methods, *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.

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, $\mathcal{R}$ 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 $\hat{r}_{ui} =< \mathbf{p}_u, \mathbf{q}_i >= \mathbf{p}_u^T \mathbf{q}_i$.

To learn user/item latent factors, [He *et al.*, 2016] developed an eALS method that introduces a weighted regression function, which assigns a zero r_{ui} value to missing entries with a confidence variable:

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

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 [Liu *et al.*, 2016]. The 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. Due to space limit, we leave out the update rule for user/item factors [He *et al.*, 2016].

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 Our View-enhanced eALS Method

Based on eALS, we consider how to effectively learn user preference from both purchase and view data. First of all, in order to differentiate their preference levels using the same user-item relation, 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. On the other hand, introducing the above pairwise relationship in view-enhaced objective function makes the original eALS learning elgorithm 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. For readability, we summarize the major notations throughout the paper in Table 1.

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.
$\mathcal{V}, \mathcal{V}_u, \mathcal{V}_i$	Similar notations for viewed interactions.
$\mathcal{RV}, \mathcal{RV}_u, \mathcal{RV}_i$	Similar 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.

Table 1: List of commonly used notations.

4.1 View-enhanced Objective Function

In E-commerce recommender systems, 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 $\hat{r}_{ui}$. However, 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, like the normally used squared loss $(r_{uv}-\hat{r}_{uv})^2$, 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.

Our view-enhanced objective function is then designed as:

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

Note that this objective function can be divided into three terms, where the first two represent the prediction error and regularizer in the former eALS solution, while the last 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 $\mathcal{R} \cap \mathcal{V} = \phi$.

4.2 Fast Learning Algorithm

As we mentioned above, the pairwise ranking loss between a viewed item and another one introduces nearly $|\mathcal{V}| \cdot N$ terms in L_{view} of Eq. (2), making the time cost become N times more if we directly use the original eALS method [He *et al.*, 2016]. To overcome this efficiency challenge, we develop a fast VALS learning algorithm that can speed-up this learning process by avoiding massive repeated computations in L_{view}. We detail this for user latent factors, while the counterpart for item factors can be achieved likewise and thus is omitted due to space limitation.

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 [He *et al.*, 2016], we only present the speed-up of L_{view} term. According to Eq. (2), we obtain the derivative of L_{view} *w.r.t.* p_{uf}:

$$\frac{\partial L_{\text{view}}}{\partial p_{uf}} = 2\sum_{v\in\mathcal{V}_u}\sum_{i\in\mathcal{R}_u} c_v(q_{if}-q_{vf})^2 \cdot p_{uf} + \tag{3}$$

$$2\sum_{v\in\mathcal{V}_u}\sum_{j\notin\mathcal{RV}_u} c_v(q_{jf}-q_{vf})^2 \cdot p_{uf} + \tag{4}$$

$$2\sum_{v\in\mathcal{V}_u}\sum_{i\in\mathcal{R}_u} c_v(\hat{r}_{ui}^f-\hat{r}_{uv}^f)(q_{if}-q_{vf}) + \tag{5}$$

$$2\sum_{v\in\mathcal{V}_u}\sum_{j\notin\mathcal{RV}_u} c_v(\hat{r}_{uj}^f-\hat{r}_{uv}^f)(q_{jf}-q_{vf}) - \tag{6}$$

$$2\sum_{v\in\mathcal{V}_u}\Big\{\sum_{i\in\mathcal{R}_u}\gamma_1 c_v(q_{if}-q_{vf}) - \sum_{j\notin\mathcal{RV}_u}\gamma_2 c_v(q_{jf}-q_{vf})\Big\}, \tag{7-8}$$

where $\hat{r}_{u\bullet}^f = \hat{r}_{u\bullet} - p_{uf}q_{\bullet f}$, i.e., the prediction without the component of latent factor f. Clearly, the bottleneck lies in the (4), (6) and (8) 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 above process by focusing on the (6) term in $\partial L_{\text{view}}/\partial p_{uf}$, as the rest can be done likewise. More specifically, we can obtain

$$\begin{aligned} &\sum_{v\in\mathcal{V}_u}\sum_{j\notin\mathcal{RV}_u} c_v(\hat{r}_{uj}^f-\hat{r}_{uv}^f)(q_{jf}-q_{vf}) = \sum_{v\in\mathcal{V}_u} c_v \cdot \sum_{j\notin\mathcal{RV}_u}\hat{r}_{uj}^f q_{jf} - \\ &\sum_{v\in\mathcal{V}_u} c_v q_{vf} \cdot \sum_{j\notin\mathcal{RV}_u}\hat{r}_{uj}^f - \sum_{v\in\mathcal{V}_u} c_v \hat{r}_{uv}^f \cdot \sum_{j\notin\mathcal{RV}_u} q_{jf} + \sum_{v\in\mathcal{V}_u} c_v \hat{r}_{uv}^f q_{vf} \cdot \sum_{j\notin\mathcal{RV}_u} 1. \end{aligned} \tag{9}$$

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, taking near $O(N)$ time. Therefore, we move forward to speed up the calculation of these terms,

$$\begin{aligned}\sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj} &= \sum_{j=1}^{N}\hat{r}^f_{uj} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} = \sum_{j=1}^{N}\sum_{k\neq f}p_{uk}q_{jk} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} \\ &= \sum_{k\neq f}p_{uk}\sum_{j=1}^{N}q_{jk} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}; \qquad (10)\\ \sum_{j\notin\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf} &= \sum_{k\neq f}p_{uk}\sum_{j=1}^{N}q_{jk}q_{jf} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf}.\end{aligned}$$

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 $\mathbf{D}^q = \sum_{i=1}^{N}\mathbf{q}_i$, and $\mathbf{E}^q$ cache as $\mathbf{E}^q = \sum_{i=1}^{N}\mathbf{q}_i\mathbf{q}_i^T$, which can be pre-computed and used in updating the latent factors for all users. Based on these two caches, Eq. (10) can be further evaluated as:

$$\sum_{k\neq f}p_{uk}d^q_k - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj} \text{ and } \sum_{k\neq f}p_{uk}e^q_{kf} - \sum_{j\in\mathcal{RV}_u}\hat{r}^f_{uj}q_{jf}, \qquad (11)$$

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

Overall, Eq. (9) can be calculated efficiently with the help of $\mathbf{D}^q$ and $\mathbf{E}^q$ caches. As for the rest 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 $O(K + |\mathcal{RV}_u|)$.

Similarly, for updating item latent factors, we can also achieve the time complexity of $O(K + |\mathcal{RV}_i|)$ by applying the above speed-up strategy. Algorithm 1 summarizes the accelerated algorithm for our VALS learner. Note that solving $\partial(L_{\text{eALS}} + L_{\text{Reg}})/\partial p_{uf}$ and $\partial(L_{\text{eALS}} + L_{\text{Reg}})/\partial q_{if}$ uses the original eALS method. Overall, one VALS iteration takes $O((M + N)K^2 + (|\mathcal{R}| + |\mathcal{V}|)K)$ time, which only depends on the number of observed interactions.

Note that some previous works [Takács and Tikk, 2012; Zhang *et al.*, 2017] have also used the squared form of loss function and learning method based on ALS or coordinate descent technique. However, compared to VALS, 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. Thus we plan to solve this issue in future work.

Algorithm 1: Fast VALS 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
```

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].

We take three steps for data preprocessing. First, we merge the repetitive purchases into one purchase with the earliest timestamp, as we aim to recommend novel items for a user to purchase. Second, we filter out users' views on those purchased items to avoid information leaking. Third, we filter out users and items with less than 12 and 16 purchase interactions, respectively, to alleviate the data sparsity issue. Table 2 summarizes the data statistics.

Dataset	Purchase#	View#	User#	Item#	Sparsity
Beibei	2,654,467	23,668,454	158,907	119,012	99.99%/99.87%
Tmall	352,768	1,585,225	28,059	32,339	99.96%/99.83%

Table 2: Statistics of the evaluation datasets.

Evaluation Methodology. In the evaluation, we adopt the *leave-one-out* protocol [Rendle *et al.*, 2009; He *et al.*, 2016], where the latest purchase interaction of each user is held out for testing and the models are trained on the remaining data. For the metrics, we employ *Hit Ratio* (HR) and *Normalized*

[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

	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	$[0.125, 0.25, 0.5, 1, 2, 4]$	1	2
β_1	MC-BPR	$[0.01, 0.05, 0.1, 0.5, 1, 5, 10]$	0.05	5
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

Table 3: Parameter exploration for baselines and optimal settings.

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 methods that only use primary purchase feedback, we choose:

- eALS [He *et al.*, 2016]. This is a state-of-the-art MF method for implicit recommender systems. We tuned the weight of missing data s_i.

- BPR [Rendle *et al.*, 2009]. 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 view feedback, we choose the following methods,

- MR-BPR [Krohn-Grimberghe *et al.*, 2012]. Applying CMF method to BPR, this method exerts the impact of viewing behavior on predicting purchases with a weight α.

- MC-BPR [Loni *et al.*, 2016]. 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.

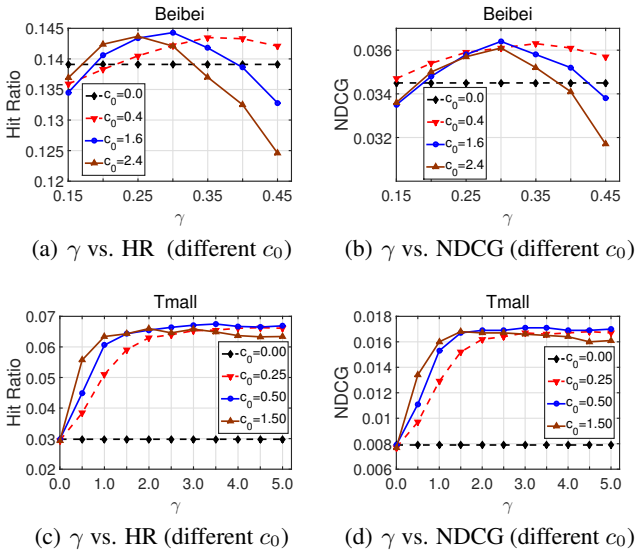
- MFPR [Liu *et al.*, 2017]. This is a most recent method that integrates multiple types of implicit feedback. We adapt it to our case by generating training item pairs in the same way as BPR, and use a fixed learning rate $\varepsilon_{\text{MFPR}}$ in SGD.

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 paper. 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 set λ as 0.001 for all methods for a fair comparison. Since the findings are consistent across the number of latent factors K, we report the results of $K = 32$ only.

Efficiency. Our experiment on the same machine (Intel Xeon 2.10 GHz CPU) shows that the actual training time per iteration for VALS on Beibei dataset is about 75s. When training set reduces to $(1/4, 1/2, 3/4)$ of the original size, the time is $(15s, 35s, 55s)$, corresponding to our theoretical analysis of linearity with the size of observed data.

5.2 Hyper-parameter Investigation

Our VALS 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



(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)

Figure 1: Impact of weighting parameters c_0 and margin values γ on VALS's performance.

view data. Without loss of generality, we set margin values $\{\gamma_1, \gamma_2\}$ uniformly as $\gamma_1 = \gamma_2 = \gamma$. 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 (γ, c_0), we conduct a grid search over these two parameters and report the mean values of both HR and NDCG in last 10 iterations.

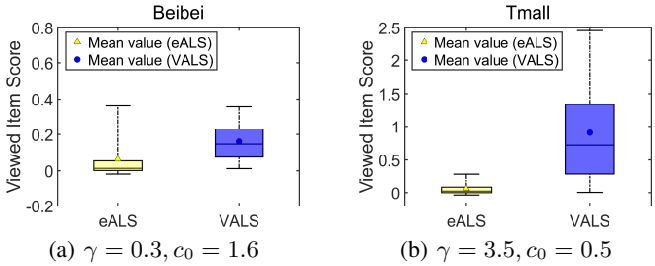
Figure 1 plots the prediction accuracy of VALS with different γ and c_0. First, we study the impact of margin value γ. For Beibei (Figure 1a and b), we observe that the best γ values are between 0.25 and 0.35 under different values of weight c_0. Since the prediction is optimized to indicate the preference level, this highlights the necessity of using an appropriate margin γ so as to control the preference level of viewed interactions, which is lower than purchased ones but higher than non-viewed ones. Surprisingly, for Tmall (Figure 1c and d), we observe that a relative large margin value ($\lambda = 3 \sim 5$) achieves better performance. According to our definition in Eq. (2), 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. For Beibei (Figure 1a and b), the peak performance is achieved when c_0 is 1.6; similarly for Tmall (Figure 1c and d), the optimal c_0 is 0.5. When c_0 becomes smaller or too large, the performance decreases in both cases, indicating the necessity to account for viewed interactions carefully.

According to above parameter exploration, we fix γ and c_0 according to the best performance evaluated by HR, i.e., $\gamma = 0.3, c_0 = 1.6$ for Beibei and $\gamma = 3.5, c_0 = 0.5$ for Tmall.

Methods	eALS		VALS		Improvement	
Datasets	HR	NDCG	HR	NDCG	ΔHR	ΔNDCG
Beibei	0.1388	0.0345	0.1443	0.0363	+3.96%	+5.22%
Tmall	0.0289	0.0079	0.0675	0.0171	+133.56%	+116.46%

Table 4: Performance improvement after integrating view data.



(a) $\gamma = 0.3, c_0 = 1.6$ (b) $\gamma = 3.5, c_0 = 0.5$

Figure 2: eALS versus VALS, in terms of quantiles (5%, 25%, 50%, 75%, 95%) and means of the predictions over viewed items.

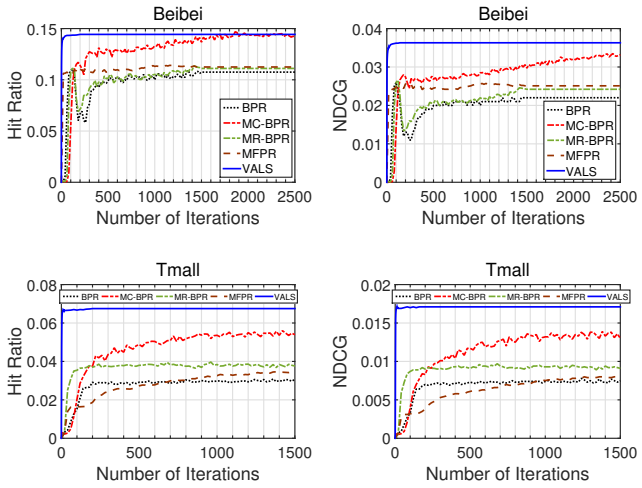


Figure 3: Prediction accuracy of VALS compared with other baseline methods in each iteration.

5.3 Performance Gain of View Data

Table 4 displays the performance of our VALS method compared with eALS, *w.r.t.* HR@100 and NDCG@100. We report their mean values in last 10 iterations. Clearly, after integrating view data as the intermediate feedback, our method significantly outperforms the original eALS that only leverages purchase data. For Beibei, the relative improvement in terms of HR and NDCG are 3.96% and 5.22%, respectively. Moreover, for Tmall, we observe an improvement of over 100% on both two metrics. This indicates that users' viewing behaviors in Tmall are much more valuable for learning a more accurate preference order among different items.

To further clarify the distinct performance gain on two datasets, we compare the predictions over viewed items between two datasets, using both eALS and VALS method to train the model. Figure 2 plots the distribution quantiles (5%, 25%, 50%, 75%, 95%) and means of viewed item scores, where the results of two 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, compared to that on Beibei (Figure 2a), VALS generates much higher scores for viewed items on Tmall (Figure 2b), i.e., 0.72 v.s. 0.15 and 0.91 v.s. 0.16 in terms of median and mean value, respectively. These higher scores of viewed items correspond to the fact that viewing behavior is more related to purchasing behavior in Tmall, which is the main reason behind the over 100% improvement of VALS method. Moreover, this also verifies our previous discussion of γ that the VALS model with a large γ is more likely to generate large predictions outside $(0, 1)$, as γ is set as 3.5 for Tmall.

5.4 Performance Comparison

Figure 3 shows the prediction accuracy of each method in each training iteration. Note that we only run VALS for 200 iterations on two datasets and run other methods except MC-BPR for 1500 iterations on Beibei dataset, which are enough for them to converge. We have the following **three key observations**. **First**, we see that VALS achieves 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 10.0% and 28.4%). Compared with MC-BPR, VALS mainly benefits from 1) margin-based design that controls the pairwise ranking between predictions of different feedback and 2) the whole-data based strategy of handling missing data. **Second**, 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. **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., 0.0330 v.s. 0.0345). It means that whole-data based strategy is a better candidate compared to sampling-based one when improving implicit recommender systems.

We notice that BPR-based methods show unusual spike in early iterations and 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. This may cause unstable performance because these popular items are ranked high in early iterations.

6 Conclusion

We study the problem of improving implicit recommender systems by integrating both purchase and view data. Based on the state-of-the-art eALS method we model user's viewed

interactions as an intermediate feedback between purchased and non-viewed interactions. To address the key efficiency challenge in optimization, we further develop a fast learning algorithm which efficiently learns parameters from the whole data instead of sampling negative instances. With these designs, our VALS method not only achieves higher accuracy, but also becomes practical for large-scale data.

This work has focused on collaborative filtering setting, which only leverages the feedback data and is mostly used in the candidate selection stage of industrial recommender systems. In future, we will focus more on the ranking stage, integrating view data into generic feature-based models, such as the expressive neural factorization machines [He and Chua, 2017]. In addition, our VALS method can also be applied to other domains of implicit recommender systems where various additional user feedback can be leveraged, like news [Agarwal *et al.*, 2012], online videos [Covington *et al.*, 2016] and social networks [Hong *et al.*, 2013].

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under 6171101425, 61621091 and 61673237, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology. This project is also part of NExT, supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative. The corresponding author is Xiangnan He.

References

[Agarwal *et al.*, 2012] Deepak Agarwal, Bee-Chung Chen, and Xuanhui Wang. Multi-faceted ranking of news articles using post-read actions. In *CIKM*, pages 694–703, 2012.

[Bayer *et al.*, 2017] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350, 2017.

[Cao *et al.*, 2017] Cheng Cao, Hancheng Ge, Haokai Lu, Xia Hu, and James Caverlee. What are you known for?: Learning user topical profiles with implicit and explicit footprints. In *SIGIR*, pages 743–752. ACM, 2017.

[Cheng *et al.*, 2014] Jian Cheng, Ting Yuan, Jinqiao Wang, and Hanqing Lu. Group latent factor model for recommendation with multiple user behaviors. In *SIGIR*, pages 995–998, 2014.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[Ding *et al.*, 2018] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. An improved sampler for bayesian personalized ranking by leveraging view data. In *WWW*, pages 13–14, 2018.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

[Hong *et al.*, 2013] Liangjie Hong, Aziz S Doumith, and Brian D Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *WSDM*, pages 557–566, 2013.

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

[Krohn-Grimberghe *et al.*, 2012] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*, pages 173–182, 2012.

[Lerche and Jannach, 2014] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. In *RecSys*, pages 353–356, 2014.

[Liu *et al.*, 2016] An-An Liu, Wei-Zhi Nie, Yue Gao, and Yu-Ting Su. Multi-modal clique-graph matching for view-based 3d model retrieval. *IEEE Transactions on Image Processing*, 25(5):2103–2116, 2016.

[Liu *et al.*, 2017] Jian Liu, Chuan Shi, Binbin Hu, Shenghua Liu, and S Yu Philip. Personalized ranking recommendation via integrating multiple feedbacks. In *PAKDD*, pages 131–143, 2017.

[Loni *et al.*, 2016] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Bayesian personalized ranking with multi-channel user feedback. In *RecSys*, pages 361–364, 2016.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[Pan *et al.*, 2015] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, 73:173–180, 2015.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Singh and Gordon, 2008] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[Takács and Tikk, 2012] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *RecSys*, pages 83–90, 2012.

[Tang *et al.*, 2016] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. An empirical study on recommendation with multiple types of feedback. In *KDD*, pages 283–292, 2016.

[Yuan *et al.*, 2014] Ting Yuan, Jian Cheng, Xi Zhang, Shuang Qiu, Hanqing Lu, et al. Recommendation by mining multiple user behaviors with group sparsity. In *AAAI*, pages 222–228, 2014.

[Zhang *et al.*, 2017] Yan Zhang, Defu Lian, and Guowu Yang. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *AAAI*, pages 1669–1675, 2017.

[Zhao *et al.*, 2015a] Tong Zhao, Julian McAuley, and Irwin King. Improving latent factor models via personalized feature projection for one class recommendation. In *CIKM*, pages 821–830, 2015.

[Zhao *et al.*, 2015b] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. Improving user topic interest profiles by behavior factorization. In *WWW*, pages 1406–1416, 2015.