The Framework of Personalized Ranking on Poisson Factorization

Li-Yen Kuo, Chung-Kuang Chou, and Ming-Syan Chen, *Life Fellow, IEEE*

Abstract—Matrix factorization (MF) has earned great success on recommender systems. However, the common-used regression-based MF not only is sensitive to outliers but also unable to guarantee that the predicted values are coordinate with the user preference orders, which is the basis of common measures of recommender systems, e.g. nDCG. To overcome the aforementioned drawbacks, we propose a framework for personalized ranking of Poisson factorization that utilizes learning-to-rank based posteriori instead of the classical regression-based ones. Owing to the combination, the proposed framework not only preserves user preference but also performs well on a sparse matrix. Since the posteriori that combines learning to rank and Poisson factorization does not follow the conjugate prior relationship, we estimate variational parameters approximately and propose two optimization approaches based on variational interference. As long as the used learning-to-rank model has the 1st and 2nd order partial derivatives, by exploiting our framework, the proposed optimizing algorithm can maximize the posteriori whichever the used learning-to-rank model is. In the experiment, we show that the proposed framework outperforms the state-of-the-art methods and achieves promising results on consuming log and rating datasets for multiple recommendation tasks.

Index Terms—Poisson factorization, learning to rank, recommender systems.

1 Introduction

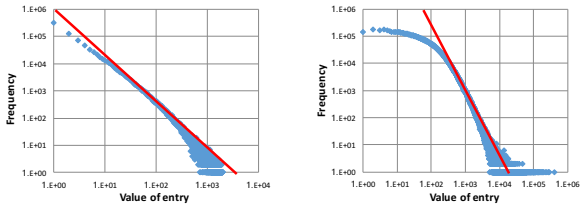
In recent years, personalized recommender systems which find desirable items (e.g., films, songs, books) for each user have been an important research topic because of their key roles in e-commerce. The principal goal of recommender systems is to retrieve items which have not been consumed by the target user yet but will be consumed in the future. To tackle this, collaborative filtering (CF), based on the assumption that a user would consume an item consumed by other users with similar preference, conducts satisfying performance by utilizing user-item consuming logs and has been one of the most successful techniques. By regarding user-item relationships as a *utility matrix*, matrix factorization (MF), which estimates a predicted score of an entry by computing the inner product of two low-dimensional latent factors, is widely used for collaborative filtering in recommender systems to retrieve the desirable unconsumed items, i.e., zero entries with high predicted scores.

Given a utility matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$, MF models user preferences and item attributes by two low-rank matrices $\boldsymbol{\theta} \in \mathbb{R}^{M\times K}$ and $\boldsymbol{\beta} \in \mathbb{R}^{N\times K}$ respectively with the goal that the predicted score computed by the inner product of two latent factors $\boldsymbol{\theta}_u \in \mathbb{R}^K$ and $\boldsymbol{\beta}_i \in \mathbb{R}^K$ should be as close to its corresponding ground-truth value x_{ui} as possible for each user-item entry (u, i). To measure the distance between x_{ui} and $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$, the square-loss, i.e., $(x_{ui} - \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)^2$, is widely used. When assuming that each entry in utility matrix $\mathbf{X}$ is on a Gaussian, we can generally define the objective of matrix factorization as a log likelihood estimation, where the ground-truth value x_{ui} is an observation and its corresponding predicted score $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ can be an estimator for each user-item entry (u, i), i.e., $p(x_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$. We call these methods regression-based methods.

Although regression-based MF methods [2], [3], [4], [6], [17] have earned great success, the widely-used regression-based objective functions, namely, square-loss minimization, cannot guarantee that the predicted values are in line with user preference orders. For example, given four items in a system, consider a ground-truth consuming vector $\mathbf{x}_u = [5, 4, 3, 2]$, where the first entry means that user u gives item 1 rating 5 or user u consumes item 1 five times. The preference order of u is: item 1 $\succ$ item 2 $\succ$ item 3 $\succ$ item 4. Consider two predictions for $\mathbf{x}_u$: prediction 1 $\hat{\mathbf{x}}_u^{(1)} = [5, 4, 2, 3]$ and prediction 2 $\hat{\mathbf{x}}_u^{(2)} = [4, 3, 2, 1]$. The square-loss of prediction 1 is 2[1], which is smaller than that of prediction 2, which equals 4, and thus prediction 1 will be selected as the solution. However, prediction 2 is more preferable since it can preserve the preference order of user u. Since recommender systems are usually evaluated through measures in learning to rank, such as precision@Z, recall@Z and nDCG@Z, all of which measure preference orders of users, an objective function based on a ranking criterion can obtain more desirable results.

In addition, users have biases, so one user's rating is often not compatible with another user. For instance, considering rating behaviors $\mathbf{x}_u = [1, 1, 1, 5]$ and $\mathbf{x}_v = [1, 5, 5, 5]$, the differences of ratings between item 1 and item 5 for two users u and v are equal[2], whereas their rating distributions are absolutely different. Item 4 is more desirable by user u than user v though both users give rating 5 to item 4. In contrast, the preference order can describe the condition easily: item 4 $\succ$ {item 1, item 2, item 3} for user u and {item 2, item 3, item 4} $\succ$ item 1 for user v. More specifically, in $\mathbf{x}_u$, preference order only represents that item 4 should be ranked higher than the others and neglects the ranking positions of the others. Thus, a loss function based on

1. The square-loss of $\hat{\mathbf{x}}_u^{(1)}$: $(5-5)^2+(4-4)^2+(2-3)^2+(3-2)^2 = 2$.
2. For both users u and v, the difference of rating between item 1 and item 4 is $5 - 1 = 4$.



(a) Last.fm1K (b) Last.fm360K

Fig. 1. User-item consuming behaviors in real-life follow power-law distributions approximately, where the red lines represent the regression lines to fit the exponents.

11	13	0	0
10	8	0	0
0	0	8	20
0	0	**100**	10
0	0	8	12

=

0	1.94
0	1.45
8.07	19.07
100.5	0
8.05	11.12

×

0	0	0.995	0.01
0.079	0.081	0	0.99

(a) A utility matrix $\mathbf{X}$ with an outlier

11	13	0	0
10	8	0	0
0	0	8	20
0	0	12	10
0	0	8	12

=

0	16.99
0	12.71
21.21	0
14.83	0
14.41	0

×

0	0	0.528	0.85
0.698	0.717	0	0

(b) A utility matrix $\mathbf{X}'$ without outliers

Fig. 2. Outlier values in a utility matrix may disturb the result of factorization.

preference order can prune unnecessary objectives, i.e., the predicted scores of item 1, item 2 and item 3 should be the same, and focus on critical ones, i.e., item 4 should be ranked before the others.

Moreover, the consuming frequency on items follows a power-law distribution approximately in real-world datasets [25]. Figure 1 shows that the value of an entry follows a power-law distribution approximately in real-world datasets. In other words, both of the two datasets comprise plenty of entries with very large values, which are called outlier entries in the rest of the paper. This phenomenon exists commonly in implicit count data. Hence, the effect of outliers is unavoidable. When the ground-truth value of an outlier entry x_{ui} is far from its predicted score $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$, the corresponding loss will be much higher than other entries. As a result, outliers will be dominant in a regression-based objective function and thus hinder an optimizer from fitting other entries. In Figure 2(a), for instance, the first latent factor is dominated by the outlier, $\mathbf{X}(4,3) = 100$. On the contrary, the factorization result in Figure 2(b) is more reasonable because of the lack of outliers in $\mathbf{X}'$. To the best of our knowledge, the state-of-the-art MF methods that tackle the sparsity of a utility matrix do not consider outliers and user preference simultaneously.

To mitigate the aforementioned drawbacks, in this paper, we propose the framework for personalized ranking on Poisson factorization, which maximizes the posterior estimation of personalized ranking. We view the MF problem on a sparse matrix as three objectives: *user preference order*, *insensitivity to outliers* and *separation between observed and unobserved entries*. To the first and the second ends, the proposed framework utilizes a ranking-based objective function instead of a regression-based one. To the third end, we introduce *probability of observation* to keep all the predicted values of nonzero entries positive without raising computational cost. If we directly considered that nonzeros should be ranked higher than zeros, the pair-wise the relationship would raise the computational cost drastically. Hence, by introducing the third objective, the proposed framework can not only utilize LTR-based objective but also inherit the fast inference of HPF. Moreover, since the latent variables are tangled, we utilize variational inference (VI) to approximate the posterior estimation. To tackle the dissatisfaction of the conjugate prior relationship of the proposed leaning-to-rank based posteriori, which is called *unconjugacy* in the rest of the paper, we approximate the posteriori as a quadratic form in terms of Poisson variables by utilizing Taylor's series, and thus the estimation for the variational parameters of the Poisson variables can be approximated by introducing the Lambert W function. Notice that our framework does not specify the learning-to-rank (LTR) model. In other words, as long as the employed learning-to-rank model has the 1st and 2nd order partial derivatives, by utilizing our framework, the proposed optimizing algorithm can maximize the posteriori whichever the utilized LTR model is.

Compared with PF, our framework has two major advantages. First, the objective function based on personalized ranking can be viewed as a relaxation of the regression-based objective function, and hence our framework can achieve promising results by neglecting unnecessary objectives. In other words, the proposed framework fits orders rather than values. Second, our framework is much more robust and more insensitive to outlier entries in a utility matrix. The contributions in the paper are listed as follows.

- We propose a personalized ranking framework on PF, which combines the advantages of learning to rank and PF. Our framework is insensitive to outliers on a sparse utility matrix.
- Our framework does not specify the LTR model. The only requirement is that a model should have continuous gradient, i.e., existing 1st and 2nd order partial derivatives.
- Our framework can be used on the traditional matrix factorization problem as well as the LTR problem. Even if only user preference orders are known with the lack of consuming logs, our framework still works.
- To solve the unconjugacy of the posteriori, we approximate variational parameters by introducing the Lambert W function and two optimization approach based on VI and stochastic VI respectively are proposed.
- In the experiment, four LTR models applied to our framework outperforms the state-of-the-art method in terms of precision, recall and F_1 score for recommendation tasks on implicit count and explicit rating data.

The remainder of the paper is organized as follows. In Section 2, we discuss the related works including matrix factorization and learning to rank. In Section 3, we define the posteriori of the framework and propose two updating approaches based on variational inference. In Section 4, we select four multiple models as examples and demonstrate how to apply them to our framework. In Section 5, an empirical analysis is conducted, followed by our conclusion in Section 6.

2 Related Works

Our work has connections to existing works in matrix factorization and learning to rank. The two subjects are introduced briefly in the section.

2.1 Matrix Factorization

In recommender systems, Matrix Factorization (MF) outputs two low-rank matrices by extracting K latent factors underlying a utility matrix to retrieve items that the target user has not consumed yet before but will consume in the future. Since most of real-world consuming data are non-negative, Non-negative Matrix Factorization (NMF) [17] is proposed to conform with this naturality. Lee and Seung [8] propose an efficient algorithm for NMF. Salakhutdinov and Mnih [6] present Probabilistic Matrix Factorization (PMF), which models the conditional probabilities of the utility matrix to control the model capacity automatically by introducing prior parameters. However, owing to the Gaussian-based likelihood that pays equal emphasis on a nonzero entry and a zero entry, the model will place more total emphasis on zero entries and thus fails to fit nonzero entries [4].

Since real-world utility matrices are usually sparse, many methods are proposed to reduce the impact of sparsity on the performance of MF. Bayesian Personalized Ranking (BPR) [9] considers a user's consuming as a ranking problem where observed data should be ranked higher than unobserved ones. Gopalan et al. [4] introduce a model, called Hierarchical Poisson Factorization (HPF), which factorizes user-item consuming over Poisson distributions. Non-parametric PF [3] controls the dimensionality of latent factors automatically. In addition, Johnson [10] proposes logistic matrix factorization which models the probability of user-item consuming by logistic function. Compared with these regression-based methods, the proposed framework not only preserves user preference orders but also is more robust to outliers than those models.

Deep-learning based approaches [26], [27] utilize multiple layers to abstract sparse input vectors non-linearly to obtain user and item latent vectors, where each entry is assumed to be over a Gaussian. As such, these methods can express more complicated data. Even so, compared with these methods, PRPF has three advantages. First, Poisson-based posteriori can output more reasonable latent vectors to capture distributions of user activity and item popularity. Second, compared with Gaussian likelihood, Poisson-based methods can down-weight the effect of zeros. Last, updating a Poisson-based model is much faster than updating a deep-learning-based method since the Gaussian likelihood considers all the entries in a matrix so that they learn by stochastic-based algorithms to accelerate convergence.

2.2 Learning To Rank

Ranking is a critical problem in recommender systems as well as information retrieval (IR). How to learn the underlying model such that the personalized output is corresponding to a user's preference has been a major research topic in IR, which is well-known as *learning to rank* (LTR). LTR can be categorized into three groups, namely, point-wise, pair-wise and list-wise. Point-wise LTR, similar to regression, regards the relevance degree as real numbers. Since point-wise LTR focuses on single item, the relative order between items cannot be naturally considered in the learning process. Pair-wise LTR considers the relative order between two items. RankNet [11], a pair-wise LTR method, uses cross-entropy as the loss function and models the ranking probability by the logistic function. RankBoost [12] adopts AdaBoost as the classifier over item pairs.

Compared with pair-wise LTR, some list-wise LTR models can put more emphasis on top-ranked items rather than treat all the items equally. IR measures such as nDCG are reflexes of this concept. In nDCG, the weights of low-ranked items will be discounted. The loss function of list-wise LTR can be separated into two types: list-wise ranking loss [23], e.g., Plackett-Luce model, or IR-evaluation-based loss [22]. Since most IR measures are discontinuous, LambdaRank [13] only computes the gradients of the objective to bypasses the discontinuousness of IR measures and thus makes the gradient adaptive to optimize IR measures such as nDCG, MAP and MRR. LambdaMART [15] combines LambdaRank and MART [14], which is a boosted tree model of which the output is the combination of the outputs of regression trees. Valizadegan et al. [16] propose a probabilistic framework that optimizes the expectation of nDCG over all the possible permutations of documents. On the other hand, ListNet [23] applies Plackett-Luce model to LTR. ListNet uses the KL-divergence between the permutation probability distributions of the desired model and the ideal one to define its list-wise ranking loss with the cost of high computational complexity. To alleviate the computational cost, ListMLE [24] uses a negative log-likelihood as the ranking loss to replace KL-divergence. The computational complexity can be greatly reduced compared with ListNet. Notice that LTR-based MF methods are based on list-wise ranking loss rather than IR measures because of the need of the gradient of a loss function.

Recently, the problem of regression-based MF is addressed and several LTR-based MF methods [18], [20] are proposed. However, neither ListRank-MF nor ListPMF tackles the sparsity of a utility matrix well. In our model, the combination of LTR-based objective function and HPF can alleviate the impact of the sparsity.

3 Personalized Ranking of Poisson Factorization

In the section, we propose the framework of personalized ranking on Poisson factorization. We firstly define ranking criterion and probability of observation. The posteriori of the proposed framework is proposed based on the two definitions and HPF. We then introduce a method for approximation by variational inference to seek the solutions to the proposed model efficiently. We furthermore propose a user-wise stochastic mini-batch variational inference to apply the model to massive datasets and improve the efficiency of training. Our notations are summarized as Table 1.

3.1 Ranking Criterion

The proposed framework uses a probabilistic model based on pair-wise learning to rank to achieve the two propositions described as follows.

TABLE 1
Notation Table

Symbol	Description
$\mathcal{U}$	user set, $\|\mathcal{U}\| = M$
$\mathcal{I}$	item set, $\|\mathcal{I}\| = N$
K	dimensionality of latent factors
$\mathbf{X}$	utility matrix, $\mathbf{X} \in \mathbb{R}^{M \times N}$
x_{ui}	the consuming of item i by user u, $x_{ui} = \mathbf{X}(u, i)$
$\mathcal{X}$	observed entry set, $\mathcal{X} = \{x_{ui} \| x_{ui} > 0\}$
$\hat{x}_{ui}$	the prediction of x_{ui}, $\hat{x}_{ui} = \hat{\mathbf{X}}(u, i)$
$\boldsymbol{\theta}_u$	the row vector of $\boldsymbol{\theta}$ with respect to user u
θ_{uk}	the k-th factor of $\boldsymbol{\theta}_u$, $\boldsymbol{\theta}(u, k) = \theta_{uk}$
$\mathcal{I}_u$	set of items consumed by user u, $\{i \| x_{ui} > 0\}$
$\succ_u$	ground-truth user preference order w.r.t. user u
π_u	item permutation w.r.t. user u
δ	parameter to rescale preference probability, $\delta \in \mathbb{R}^+$
α	parameter to balance nonzero/zero entries, $\alpha \in \mathbb{R}^+$
C	a constant, $C \in \mathbb{R}^+$

PROPOSITION 1. *User preference order.* The inference by a good MF model should preserve the preference order for each user.

PROPOSITION 2. *Insensitivity to outliers.* In consuming log datasets, such as listening count, a user may consume certain items frequently. These entries with enormous values in a utility matrix may impair fitting other entries. Only the MF method that is insensitive to the outliers can model consuming logs well.

The personalized total ranking $\succ_u \in \mathcal{I}^2$ of all items for user u must satisfy the properties of a *strict total order* for all h, i and j in $\mathcal{I}$:

$$\begin{aligned} &i \succ_u j \Leftrightarrow x_{ui} > x_{uj}, && \text{(asymmetry)} \\ &h \succ_u i \wedge i \succ_u j \Rightarrow h \succ_u j, && \text{(transitivity)} \\ &x_{ui} \neq x_{uj} \Rightarrow i \succ_u j \vee j \succ_u i. && \text{(totality)} \end{aligned}$$

Unlike the work [9], the proposed framework does not use a non-strict total order. According to the *antisymmetry* of the non-strict total order, given that $x_{ui} = x_{uj}$, there are three possible conditions: $i \succeq_u j$, $j \succeq_u i$ and $i \succeq_u j \wedge j \succeq_u i$. In other words, $x_{ui} = x_{uj}$ can represent any possible order between items i and j. Indeed, given that $x_{ui} = x_{uj}$, we do not know which item user u prefers or whether u likes them equally. A feasible way to tackle this fact is to neglect it and only consider the cases $x_{ui} > x_{uj}$ and $x_{uj} > x_{ui}$. Hence, we apply the *asymmetry* relation to convert personal consuming logs (i.e., a row in a utility matrix) to a personalized total ranking. In other words, only the item pairs that satisfy properties of a strict total order are considered in the proposed framework.

Next, we will define the probabilistic ranking criterion between two items. Let Ω be the parameters of an arbitrary model and $\mathcal{I}_u$ be the item set where user u has consumed. To find the best personalized ranking for all items in $\mathcal{I}_u$ with respect to user u, we define the Bayesian formulation by maximizing the posterior probability

$$p(\Omega | \succ_u) \propto p(\succ_u | \Omega) p(\Omega). \tag{1}$$

Since all users are presumed to behave independently, the ranking of items for a specific user is also independent of the ranking of items for any other user. Thus, considering all users' consuming behaviors, we have the joint probability

$$\prod_{u \in \mathcal{U}} p(\Omega | \succ_u) \propto \prod_{u \in \mathcal{U}} p(\succ_u | \Omega) p(\Omega), \tag{2}$$

where $\mathcal{U}$ denotes the user set. The personalized total ranking $\succ_u$ can be represented as pair-wise or list-wise expressions.

In pair-wise expression, one focuses on the permutation order between two items. Let $\mathcal{R} \in \{0, 1\}^{M \times N \times N}$ be the user preference order, where entry $r_{uij} \in \mathcal{R}$ denotes the permutation order with respect to user u in the ground-truth data and can be defined as

$$r_{uij} = \begin{cases} 1 & \text{, if } i \succ_u j, \\ 0 & \text{, otherwise.} \end{cases} \tag{3}$$

Figure 3 shows an example of user preference order $\mathcal{R}$. Notice that only consumed items are considered to construct $\mathcal{R}$. The joint likelihood in Eq. (2) can be defined as

$$\prod_{u \in \mathcal{U}} p(\succ_u | \Omega) \propto \prod_{u \in \mathcal{U}} \prod_{i,j \in \mathcal{I}_u} p(i \succ_u j | \Omega)^{r_{uij}}. \tag{4}$$

On the other hand, in list-wise expression, one focuses on the ranking position of an item. In other words, to consider the ranking relationship of a certain item in a list-wise way, the number of items that succeed the target item is considered. We define personalized permutation π_u, where $\pi_u(i)$ denotes the ranking position of item i. Thus, $\pi_u(i)$ can be written as

$$\pi_u(i) = |\{j | j \succ_u i\}| + 1 = \sum_{j \in \mathcal{I}_u} r_{uji} + 1. \tag{5}$$

Since pair-wise expression and list-wise one are equivalent, for convenience, we use π_u to represent the ground-truth user preference order. In Section 4, we will show that the two expressions lead to various LTR models.

Our goal is to find an underlying model with parameters Ω for fitting the ordering with respect to user u, namely, π_u. In addition to the ranking criterion, the proposed framework uses logistic function to model the observation of nonzero entries and uses HPF [4] as the underlying model, both of which will be introduced in the following sections.

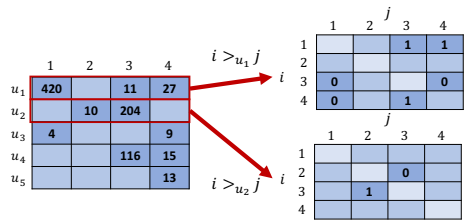


Fig. 3. An example of user preference orders.

3.2 Probability of Observation

In addition to the aforementioned ranking criterion, which is used to model the ordering of observed items for each user, we will propose probability of observation to achieve the following proposition.

PROPOSITION 3. *Separation between observed and unobserved entries.* Most of observed data are desirable for a

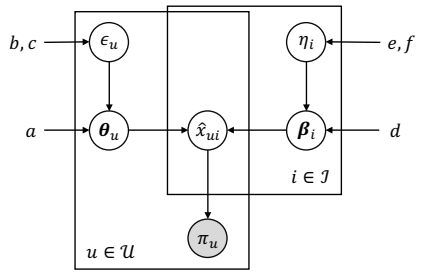


Fig. 4. The Bayesian graphical model of the proposed framework.

user, while most of unobserved ones cannot attract a user's attention. In the light of this, we regard observed data as positive and regard unobserved ones as zero.

A LTR model only focuses on the relative values among observed entries based on their ranking positions so that the ranking relationship may hold even if the values of some observed entries are close to zero. To keep all observed entries positive, we introduce probability of observation $p(\hat{x}_{ui} > 0)^\alpha$, where α denotes a tuning parameter to balance observed (nonzero) and unobserved (zero) entries. We use logistic function to represent the probability of observation, which is defined as

$$p(\hat{x}_{ui} > 0) = \sigma(\hat{x}_{ui}) = \frac{1}{1 + e^{-\hat{x}_{ui}}}. \tag{6}$$

The probability will be introduced to the posteriori of the proposed framework so as to make observed entries positive during training. Notice that we only introduce the probability of observation for observed entries since unobserved entries have not been consumed yet and only few of them would be consumed in the future. Moreover, from our observations, the mean of the prior of $\hat{x}_{ui}$ is close to zero in the beginning of the training phase because we set the mean of the prior of $\hat{x}_{ui}$ to a small value practically. Therefore, the prior of $\hat{x}_{ui}$ will make $\hat{x}_{ui}$ also close to zero without imposing an auxiliary constraint. In the light of this, introducing the probability of observation can prevent the observed entries in $\boldsymbol{\theta}$ or $\boldsymbol{\beta}$ from being close to zero practically. The underlying concept is to separate nonzero entries from zero ones. Unlike other works [28], [29], [30], we introduce the probability of observation in the proposed framework to make the model focus on the user preference order of nonzero entries and neglect the ranking relationships between nonzero and zero entries. As such, our setting will not raise the updating cost much. Hence, the proposed framework can not only utilize LTR-based objective but also inherit the fast inference of HPF.

3.3 Proposed Posterior Probability

The proposed ranking criterion only considers the items that have been consumed, that is, only the observed entries in utility matrix $\mathbf{X}$ are considered. To tackle the sparsity of $\mathbf{X}$, we use hierarchical Poisson factorization [4] as the underlying model.

Given a user set $\mathcal{U}$ with size M and an item set $\mathcal{I}$ with size N, matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ denotes a utility matrix, where entry x_{ui} represents the relationship between user u and item i. In an online music service, x_{ui} may be viewed as the frequency that user u listened to item i, which can be a song or an artist. Let $\mathcal{X}$ be the set that consists of all observed, i.e., nonzero, entries. The row vector of $\boldsymbol{\theta}$ corresponding to user u, denoted by $\boldsymbol{\theta}_u$, represents the preference of user u. Likewise, $\boldsymbol{\beta}_i$, the row vector of $\boldsymbol{\beta}$ corresponding to item i, denotes the attribute of item i. Then we use $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ to denote the inner product of vectors $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}_i$.

The generative process of the proposed framework is shown as follows.

- For each user u:
  - Draw user activity $\epsilon_u \sim \text{Gamma}(b, c)$.
  - Draw user preference $\boldsymbol{\theta}_u \sim \text{Gamma}(a, \epsilon_u)$.
- For each item i:
  - Draw item currency $\eta_i \sim \text{Gamma}(e, f)$.
  - Draw item attribute $\boldsymbol{\beta}_i \sim \text{Gamma}(d, \eta_i)$.
- For each user u and item i, draw prediction $\hat{x}_{ui} \sim \text{Poisson}(\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$.
- For each permutation w.r.t user u for each user u $\succ_u \sim p(\pi_u | \{\hat{x}_{ui} | i \in \mathcal{I}_u\}) \prod_{i \in \mathcal{I}_u} \sigma(\hat{x}_{ui})^\alpha$.

Figure 4 shows the Bayesian graphical model of the proposed framework. Considering the posteriori of ranking criteria in Eq. (2) and the probability of observation in Eq. (6), the posteriori can be defined as

$$\begin{aligned}\prod_{u=1}^{M} p(\succ_u | \Omega) \propto & \prod_{u=1}^{M} p(\pi_u | \{\hat{x}_{ui} | i \in \mathcal{I}_u\}) \\ & \prod_{u=1}^{M} \prod_{i \in \mathcal{I}_u} p(\hat{x}_{ui} > 0)^\alpha \prod_{u=1}^{M} \prod_{i=1}^{N} p(\hat{x}_{ui} | \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i) \\ & \prod_{u=1}^{M} p(\boldsymbol{\theta}_u | a, \epsilon_u) p(\epsilon_u | b, c) \\ & \prod_{i=1}^{N} p(\boldsymbol{\beta}_i | d, \eta_i) p(\eta_i | e, f),\end{aligned} \tag{7}$$

where α denotes the scale factor to balance the observed and unobserved data. The log-posteriori in terms of $\hat{x}_{ui}$ can be written as

$$\begin{aligned}\log p(\hat{x}_{ui} | \pi_u, \boldsymbol{\theta}_u, \boldsymbol{\beta}_i) &\propto \mathcal{F}_{ui}(\hat{x}_{ui}) \\ &= f_{ui}(\hat{x}_{ui}) + \log p(\hat{x}_{ui} | \boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i),\end{aligned} \tag{8}$$

where

$$f_{ui}(\hat{x}_{ui}) = \log p(\pi_u | \{\hat{x}_{ui} | i \in \mathcal{I}_u\}) + \alpha \log p(\hat{x}_{ui} > 0), \tag{9}$$

which represents the log likelihood that variable $\hat{x}_{ui}$ follows the permutation π_u. There are plenty of models to represent the personalized permutations denoted by $p(\pi_u | \{\hat{x}_{ui} | i \in \mathcal{I}_u\})$. In section 4, we will show four LTR models (one for point-wise, one for pair-wise, two for list-wise) on Poisson factorization.

The goal of the proposed framework is to infer the zero entries. Since $\hat{x}_{ui}$ is only used to learn user preference order, we only record the variables of nonzero entries, i.e., $\{\hat{x}_{ui} | (u, i) \in \mathcal{X}\}$, to reduce the memory cost. In other words, prediction matrix $\hat{\mathbf{X}}$ is recorded as a sparse matrix practically. Hence, prediction $\hat{x}_{ui}$ cannot be regarded as the inference for zero entry (u, i). We use $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ to obtain the

TABLE 2
Latent Variables, Variational Parameters and Their Complete Conditionals

Latent Variable	Type	Complete Conditional	Variational Parameter
ϵ_u	Gamma	$b+Ka, c+\sum_k \theta_{uk}$	$\tilde{\epsilon}_u^{shp}, \tilde{\epsilon}_u^{rte}$
η_i	Gamma	$e+Kd, f+\sum_k \beta_{ik}$	$\tilde{\eta}_i^{shp}, \tilde{\eta}_i^{rte}$
θ_{uk}	Gamma	$a+\sum_i \hat{x}_{ui}\varphi_{uik}, \epsilon_u+\sum_i \beta_{ik}$	$\tilde{\theta}_{uk}^{shp}, \tilde{\theta}_{uk}^{rte}$
β_{ik}	Gamma	$d+\sum_u \hat{x}_{ui}\varphi_{uik}, \eta_i+\sum_u \theta_{uk}$	$\tilde{\beta}_{ik}^{shp}, \tilde{\beta}_{ik}^{rte}$
$\hat{\mathbf{x}}_{ui}$	Mult	$\log\theta_{uk}+\log\beta_{ik}$	$\boldsymbol{\varphi}_{ui}$
$\hat{x}_{ui}$	Poisson	Eq. (14)	$\tilde{x}_{ui}$

inference, namely, predicted score, for the distribution of zero entry (u,i) rather than using $\hat{x}_{ui}$ directly since $\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ is the low-rank approximation, while $\hat{x}_{ui}$ is actually not.

Notice that, in our framework, the only requirement for $p(\pi_u|\{\hat{x}_{ui}|i\in\mathcal{I}_u\})$ is that the likelihood must be concave, which helps the approximated updating of prediction $\hat{x}_{ui}$. More detail will be stated in Section 3.5. Since all the variables except for prediction $\hat{x}_{ui}$ follow the conjugate prior relationship, we focus on the approximation of the posteriori in terms of $\hat{x}_{ui}$ in the rest of the section.

3.4 Variational Inference

Since the variables are tangled, we use variational inference to optimize the posteriori. Variational inference (VI) is a method for approximating posterior distributions, indexed by free variational parameters, by minimizing the KL divergence from the true posteriors to the approximate ones [5], [19]. Thus, the inference problem can be transformed into an optimization problem. The simplest variational family of distributions is the *mean field family*. In this family, each latent variable is independent and governed by its own variational parameter. The latent variables, their variational parameters and complete conditionals are shown in Table 2.

The latent variables in the model are listed as follows: user activity ϵ_u, item currency η_i, user preference θ_{uk}, item attribute β_{ik}, and the user-item prediction contributed by the k-th factor $\hat{x}_{uik}=\theta_{uk}\beta_{uk}$. $\hat{\mathbf{x}}_{ui}=[\hat{x}_{ui1},...,\hat{x}_{uiK}]\in\mathbb{R}^K$ is represented as a K-vector of counts and can be modeled as a multinomial $\mathrm{Mult}(\hat{x}_{ui},\boldsymbol{\varphi}_{ui})$, where $\hat{x}_{ui}=\sum_{k=1}^K \hat{x}_{uik}$. The mean-field family considers that the variables are independent and each variable is governed by its own distribution. The variational family of the proposed framework is

$$q(\boldsymbol{\theta},\boldsymbol{\beta},\epsilon,\eta,\widehat{\mathbf{X}}) = \prod_{u=1}^M\prod_{k=1}^K q(\theta_{uk}|\tilde{\theta}_{uk}) \prod_{i=1}^N\prod_{k=1}^K q(\beta_{ik}|\tilde{\beta}_{ik}) \prod_{u=1}^M q(\epsilon_i|\tilde{\epsilon}_u) \prod_{u=1}^M\prod_{i\in\mathcal{I}_u} q(\hat{\mathbf{x}}_{ui}|\boldsymbol{\varphi}_{ui})q(\hat{x}_{ui}|\tilde{x}_{ui}) \prod_{i=1}^N q(\eta_i|\tilde{\eta}_i).$$

All the variables except for $\hat{x}_{ui}$ can be estimated easily by VI since these variables follow the conjugate prior relationship. Next, we will show how to approximate the expectation of $\tilde{x}_{ui}$, that is, the variational variable of $\hat{x}_{ui}$. According to the definition of *evidence lower bound* (ELBO), the negative KL divergence from the posteriori in terms of $\hat{x}_{ui}$ in Eq. (8) to the variational probability q is

$$\begin{aligned}\mathcal{L}(\hat{x}_{ui}) \sim& \mathbb{E}_q[\log p(\hat{x}_{ui}|\pi_u,\boldsymbol{\theta}_u,\boldsymbol{\beta}_i)] - \mathbb{E}_q[\log q(\hat{x}_{ui}|\tilde{x}_{ui})]\\ =& \mathbb{E}_q[f_{ui}(\hat{x}_{ui})] + \mathbb{E}_q[\log p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i)] - \\ & \mathbb{E}_q[\log q(\hat{x}_{ui}|\tilde{x}_{ui})].\end{aligned}$$

Since the conditionals $\log p(\tilde{x}_{ui}|\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i)$ and $\log q(\hat{x}_{ui}|\tilde{x}_{ui})$ are both assumed to be Poisson distributions, $\mathcal{L}(\hat{x}_{ui})$ can be written as

$$\begin{aligned}\mathcal{L}(\tilde{x}_{ui}) \sim& \mathbb{E}_q[f_{ui}(\hat{x}_{ui})] + \\ & (\log\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i - \log\tilde{x}_{ui})\mathbb{E}_q[\hat{x}_{ui}] - (\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i - \tilde{x}_{ui}).\end{aligned} \quad (10)$$

Recall that function f_{ui} is the combination between LTR-based probability and the probability of observation. Since the LTR-based part of f_{ui} can be any form (e.g., logistic function for pair-wise LTR, Plackett-Luce model for list-wise LTR), $f_{ui}(\hat{x}_{ui})$ is not polynomial so that the expectation of $f_{ui}(\hat{x}_{ui})$ over a Poisson distribution cannot be computed directly. To solve this, we use Taylor expansion to approximate $f_{ui}(\hat{x}_{ui})$ in a polynomial form. As such, the expectation of the approximation can be computed easily.

Hence, the approximation by a Taylor series with respect to $\hat{x}_{ui}$ around *approximation point* s_{ui} can be written as

$$\begin{aligned}\mathbb{E}_q[f_{ui}(\hat{x}_{ui})] \approx& \frac{f''_{ui}(s_{ui})}{2}\mathbb{E}_q[\hat{x}_{ui}^2] + \\ & (f'_{ui}(s_{ui}) - s_{ui}f''_{ui}(s_{ui}))\mathbb{E}_q[\hat{x}_{ui}] + \\ & f_{ui}(s_{ui}) - s_{ui}f'_{ui}(s_{ui}) + \frac{s_{ui}^2}{2}f''_{ui}(s_{ui}).\end{aligned} \quad (11)$$

We will introduce a method to find an adaptive s_{ui} in the next section. Since we have $\mathbb{E}_q[\hat{x}_{ui}]=\tilde{x}_{ui}$ and $\mathbb{E}_q[\hat{x}_{ui}^2]=\tilde{x}_{ui}(\tilde{x}_{ui}+1)$, Eq. (10) can be written as

$$\begin{aligned}\mathcal{L}(\tilde{x}_{ui}) \sim& f_{ui}(s_{ui}) + (\tilde{x}_{ui}-s_{ui})f'_{ui}(s_{ui}) + \frac{(\tilde{x}_{ui}-s_{ui})^2}{2}f''_{ui}(s_{ui}) \\ & + (\log\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i - \log\tilde{x}_{ui} + 1)\tilde{x}_{ui} - \boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i.\end{aligned}$$

We use coordinate ascent to optimize $\mathcal{L}(\hat{x}_{ui})$. The gradient in terms of $\tilde{x}_{ui}$ can be

$$\frac{\partial\mathcal{L}(\tilde{x}_{ui})}{\partial\tilde{x}_{ui}} = f''_{ui}(s_{ui})\tilde{x}_{ui} - \log\tilde{x}_{ui} + h_{ui}(s_{ui}) = 0, \quad (12)$$

where

$$h_{ui}(s_{ui}) = f'_{ui}(s_{ui}) + (\frac{1}{2} - s_{ui})f''_{ui}(s_{ui}) + \log\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i. \quad (13)$$

We can use Lambert W function to approximate the solution of the equation in Eq. (12). While $\tilde{x}_{ui}$ has no solutions, $\tilde{x}_{ui}$ does not locate in the range of a Poisson distribution, i.e., $(0,\infty)$. In this situation, we set $\tilde{x}_{ui}$ to 1 since x_{ui} is positive and its prediction $\hat{x}_{ui}$ should be larger than 1. As such, we solve Eq. (12) as

$$\tilde{x}_{ui} = \begin{cases} \frac{-1}{f''_{ui}(s_{ui})}W(-f''_{ui}(s_{ui})e^{h_{ui}(s_{ui})}) & \text{, if } f''_{ui}e^{h_{ui}} < \frac{1}{e}, \\ 1 & \text{, otherwise,}\end{cases} \quad (14)$$

where $W(\cdot)$ denotes the Lambert W function and there are effective algorithms to approximate it. Practically, we can use fewer iterations to compute a rough solution of Eq. (14) to speed up the variational inference procedure.

So far, we have introduced a method to approximate the posteriori $\mathcal{L}(\tilde{x}_{ui})$ in terms of $\hat{x}_{ui}$ as a quadratic form by Taylor's expansion since the expectation of a polynomial over a Poisson distribution can be easily computed. By letting the partial derivative of the approximation be zero, we can convert the optimization problem into a Lambert W function which it exists efficient algorithms to evaluate. The only issue left is the selection of s_{ui}, which will be introduced in the next section.

3.5 Selection of Approximation Points

In the previous section, we use Taylor's expansion for approximation. Since Taylor's expansion only can approximate the objective function $\mathcal{L}$ around an approximation point locally, the closer to the optimum an approximation point we find, the more precise the estimated variational variables are. Next, we will show how to select approximation point s_{ui} which should be as close to the optimum of the posteriori $\mathcal{L}(\hat{x}_{ui})$ as possible.

In view of the complexity of concave function f_{ui}, we approximate f_{ui} by quadratic approximation to make finding the optimum of $\mathcal{L}(\tilde{x}_{ui})$ easier. The 2nd-degree Taylor polynomial of f_{ui} at approximation point s_{ui}, denoted by g_{ui}, only can approximate the part of f_{ui} near s_{ui}. If f_{ui} was approximated at a wrong point, the difference between $\mathcal{L}(\tilde{x}_{ui})$ and its approximation would be too much to find a feasible value of $\tilde{x}_{ui}$. Hence, the problem can be described as how to find approximate point s_{ui} such that $\mathcal{L}(\tilde{x}_{ui})$ and its approximation share the same optimum. To simplify this, we focus the approximation of $\mathcal{F}_{ui}(\hat{x}_{ui})$ and want to find s_{ui} such that the concave function $\mathcal{F}_{ui}(\hat{x}_{ui})$ and its concave approximation $g_{ui}(\hat{x}_{ui}) + \log p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$ have their global maxima at the same point. However, this task is still hard to tackle. The relaxation is to find s_{ui} such that $\mathcal{F}_{ui}(\hat{x}_{ui})$ and its approximation, i.e., $g_{ui}(\hat{x}_{ui}) + \log p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$, have their maxima in the same interval.

Assume that $f_{ui}(\hat{x}_{ui})$ and $p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$ have their global maxima at s and t. From THEOREM 1 derived in the Appendix, given s_{ui} such that $f'_{ui}(s_{ui}) = 0$, the approximation $g_{ui}(\hat{x}_{ui}) + \log p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$ has a global maxima in the interval $[\min(s,t), \max(s,t)]$ and so does $\mathcal{F}_{ui}$. With the progress of training, s and t are close gradually so that the approximation can be more precise. However, $f'_{ui}(x) = 0$ is a transcendental equation and there is no closed-form solution. From the prediction set with respect to user u (i.e., $\{\hat{x}_{uj}|j \in \mathcal{I}_u\}$), we set s_{ui} to $\hat{x}_{uj}$ as the approximation of the optimal selection, where

$$j = \arg\min_{l \in \mathcal{I}_u} |f'(\hat{x}_{ul})|. \tag{15}$$

The selection procedure and the updating of variational parameters are shown in Algorithm 1.

3.6 User-wise Stochastic Mini-Batch VI

To improve the efficiency of training and be capable to analyze large collections of data, the classical stochastic mini-batch variational inference [19] selects observations randomly and updates corresponding variables. In the proposed framework, since the objective function is based on user preference order, it is necessary to consider the item permutation for a user during the training phase. In the light of this, we propose a modified method, called *user-wise stochastic mini-batch variational inference*, which will be described as follows.

Algorithm 1 Variational Inference of the Framework

Require: Utility matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$, number of latent factors K, tuning parameter α and δ
Ensure: User preference $\boldsymbol{\theta} \in \mathbb{R}^{M \times K}$, item attribute $\boldsymbol{\beta} \in \mathbb{R}^{N \times K}$

For each user u and item i, initialize the user parameters $\tilde{\boldsymbol{\theta}}_u$, $\tilde{\epsilon}_u^{rte}$ and item parameters $\tilde{\boldsymbol{\beta}}_i$, $\tilde{\eta}_i^{rte}$ to the prior with a small random offset.
while *not converge* **do**
  for each user u **do**
    For each item $i \in \mathcal{I}_u$, find s_{ui} according to Eq. (15) and obtain $f'_{ui}(s_{ui})$;
    For each item $i \in \mathcal{I}_u$, compute $f''_{ui}(s_{ui})$;
    For each item $i \in \mathcal{I}_u$, compute $\tilde{x}_{ui}$ by Eq. (14);
  end for
  For each entry $x_{ui} > 0$, update the multinomials $\boldsymbol{\varphi}_{ui}$;
  For each u, k, update $\tilde{\theta}_{uk}^{shp}$, $\tilde{\theta}_{uk}^{rte}$ and θ_{uk};
  For each i, k, update $\tilde{\beta}_{ik}^{shp}$, $\tilde{\beta}_{ik}^{rte}$ and β_{ik};
  For each u, update $\tilde{\epsilon}_u^{shp}$, $\tilde{\epsilon}_u^{rte}$ and ϵ_u;
  For each i, update $\tilde{\eta}_i^{shp}$, $\tilde{\eta}_i^{rte}$ and η_i;
end while

For each iteration, we sample a subset of users $\overline{\mathcal{U}} \in \mathcal{U}$. As such, the corresponding item set $\overline{\mathcal{I}} \in \cup_{u \in \overline{\mathcal{U}}} \mathcal{I}_u$, which consists of items that users in $\overline{\mathcal{U}}$ have ever consumed, and the corresponding subset of observed entries $\overline{\mathcal{X}} = \{x_{ui} | u \in \overline{\mathcal{U}}, i \in \mathcal{I}_u\}$ are obtained. Since only the predictions $\hat{x}_{uj}$ for all $j \in \mathcal{I}_u \subseteq \overline{\mathcal{I}}$ are associated with prediction $\hat{x}_{ui}$ according to Eq. (7), the required user preference orders can be obtained from the entries set $\overline{\mathcal{X}}$. Hence, there is no need to modify the equation in Eq. (14) for updating $\hat{x}_{ui}$.

Recall that $\overline{\mathcal{I}}$ consists of items that have been consumed by users in $\overline{\mathcal{U}}$. Since all the observed entries associated with θ_{uk} are in $\mathcal{I}_u \in \overline{\mathcal{I}}$, the updating of θ_{uk} is regarded as the same with the updating in classical variational inference. Specifically, the update equations of the variational parameters of θ_{uk} can be obtained as

$$\tilde{\theta}_{uk}^{shp} = a + \sum_{i \in \overline{\mathcal{I}}} \hat{x}_{ui} \varphi_{uik}, \quad \tilde{\theta}_{uk}^{rte} = \epsilon_u + \sum_{i \in \mathcal{I}} \beta_{ik}.$$

Since our random selection procedure does not select all the users that are associated with β_{ik}, it is necessary to rescale the objective function by $|\mathcal{U}|/|\overline{\mathcal{U}}|$. To update the variational parameters of β_{ik}, we have

$$\tilde{\beta}_{ik}^{shp} = d + \frac{|\mathcal{U}|}{|\overline{\mathcal{U}}|} \sum_{u \in \overline{\mathcal{U}}} \hat{x}_{ui} \varphi_{uik}, \quad \tilde{\beta}_{ik}^{rte} = \eta_i + \frac{|\mathcal{U}|}{|\overline{\mathcal{U}}|} \sum_{u \in \overline{\mathcal{U}}} \theta_{uk}.$$

By doing so, given user u, the user preference order can be well preserved by the proposed selection procedure that considers all the items consumed by u.

4 Examples to Apply LTR Models

In this section, we will select three multiple models (one for pair-wise, two for list-wise) as examples and demonstrate how to apply them to our framework.

4.1 Point-wise Learning to Rank

In order to inherit the fast inference of HPF, the objective function of point-wise LTR should be regression-based. Thus, the ranking problem is reduced to a regression problem. By reassigning a positive value to latent variable $\hat{x}_{ui}$ of each nonzero entry according to the ground-truth preference order, probabilities $p(\pi_u|\{\hat{x}_{ui}|i \in \mathcal{I}_u\})$ and $p(\hat{x}_{ui} > 0)$ in Eq. (7) are 1 consistently. As such, the model is reduced to HPF.

To reduce the influence of outliers of entries in a utility matrix, for each observed entry, we use a trivial method to assign a new value to its prediction according to its ranking position in the user preference order. Thus, the prediction of x_{ui} is defined as

$$\hat{x}_{ui} = |\{j|i \succ_u j\}| + 1. \tag{16}$$

By doing so, we use HPF to optimize the likelihood of the predictions rather than the real values of observed entries.

4.2 Pair-wise Learning to Rank

Model definition. From the posteriori in Eq. (7), we define the ranking probability $p(\pi_u|\{\hat{x}_{ui}|i \in \mathcal{I}_u\})$ with respect to user u as

$$p(\pi_u|\{\hat{x}_{ui}|i \in \mathcal{I}_u\}) \propto \prod_{i,j \in \mathcal{I}_u} p(i \succ_u j|\hat{x}_{ui}, \hat{x}_{uj})^{r_{uij} C/|\mathcal{I}_u|},$$

where the preference for user u can be defined as

$$p(i \succ_u j|\hat{x}_{ui}, \hat{x}_{uj}) = \sigma(\delta(\hat{x}_{ui} - \hat{x}_{uj})) = \frac{1}{1 + e^{-\delta(\hat{x}_{ui} - \hat{x}_{uj})}}, \tag{17}$$

where σ denotes the logistic function and δ is a scale factor. Since the model is based on pair-wise learning to rank, the model should compute the ranking criterion defined in Eq. (17) for each pair of items. The number of ranking criterion terms increases quadratically with the increasing of $|\mathcal{I}_u|$. Accordingly, the posteriori will be seriously dominated by the users who consumes a large number of items. To avoid the skewness of the distributions of the count of item pairs per user, we introduce a user-level normalization $C/|\mathcal{I}_u|$ to the posteriori by following the work in [7].

According to Eq. (9), function f_{ui} can be written as

$$\begin{aligned} f_{ui}(x) =& \frac{-C}{|\mathcal{I}_u|} \sum_{i \succ_u j} \log(1 + e^{-d_{uj}(x)}) + \\ & \frac{-C}{|\mathcal{I}_u|} \sum_{h \succ_u i} \log(1 + e^{d_{uh}(x)}) - \alpha \log(1 + e^{-x}), \end{aligned} \tag{18}$$

where $d_{uj}(x) = \delta(x - \hat{x}_{uj})$. While $x_{ui} = x_{uj}$ (i.e., $i \succ_u j \wedge j \succ_u i$), item j will not affect the value of $f(x)$ since $-\log(1 + e^{-d_{uj}(x)}) - \log(1 + e^{d_{uj}(x)}) = 0$. In this case, the posteriori in terms of $\hat{x}_{ui}$ is dominated merely by the probability of observation $p(\hat{x}_{ui} > 0)$ and the conditional $p(\hat{x}_{ui}|\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i)$. As a result, when the ratings/consumings of items i and j are the same for user u, our framework will not need to impose the consideration, $x_{ui} = x_{uj}$, in the objective function.

Let R_{ui} be the number of items ranked higher than item i with respect to user u, i.e., $R_{ui} = \sum_{h \succ_u i} 1$. The first derivative of $f_{ui}(x)$ can be

$$\begin{aligned} f'_{ui}(x) =& \frac{C\delta}{|\mathcal{I}_u|} \sum_{i \succ_u j} \frac{1}{1 + e^{d_{uj}(x)}} - \frac{C\delta}{|\mathcal{I}_u|} \sum_{h \succ_u i} \frac{e^{d_{uh}(x)}}{1 + e^{d_{uh}(x)}} + \\ & \frac{\alpha e^{-x}}{1 + e^{-x}} \\ =& \frac{C\delta}{|\mathcal{I}_u|} \sum_{i \succ_u j} \sigma(-d_{uj}(x)) + \frac{C\delta}{|\mathcal{I}_u|} \sum_{h \succ_u i} \sigma(-d_{uh}(x)) - \\ & \frac{C\delta}{|\mathcal{I}_u|} R_{ui} + \frac{\alpha}{1 + e^{x}}, \end{aligned} \tag{19}$$

We have the second derivative of $f_{ui}(x)$ as

$$\begin{aligned} f''_{ui}(x) =& -\frac{C\delta^2}{|\mathcal{I}_u|} \sum_{i \succ_u j} \frac{e^{d_{uj}(x)}}{(1 + e^{d_{uj}(x)})^2} \\ & - \frac{C\delta^2}{|\mathcal{I}_u|} \sum_{h \succ_u i} \frac{e^{d_{uh}(x)}}{(1 + e^{d_{uh}(x)})^2} - \frac{\alpha e^{x}}{(1 + e^{x})^2}. \end{aligned} \tag{20}$$

Time complexity analysis. From Algorithm 1, personalized ranking is learned in Lines 3-7. Assume that a user consumes D items in average. It takes $O(D^3)$ in Line 4 since computing f_{ui} takes $O(D)$ according to Eq. (18) for each (u, i) pair and there are D items each of which considers D candidates according to Eq. (15). In Line 6, it takes $O(DE)$ where E denotes the number of iterations for solving Lambert W function. Latent parameters updating in Lines 8-12 takes $O(X)$, where $X = |\mathcal{X}|$ denotes the number of nonzero entries in $\mathbf{X}$. The time complexity per iteration (Lines 3-12) is thus $O(MD(D^2+E)+X)$. The algorithm is efficient since D is much smaller than X usually. When some users consume many items, we can only consider top-F items rather than D ones per user to reduce the training complexity.

4.3 List-wise Learning to Rank

Model definition. Considering the preference order of user u, given the relevance scores of the items outputted by the underlying model, i.e., $\{\hat{x}_{ui}|i \in \mathcal{I}_u\}$, Plackett-Luce model defines a probability for each possible permutation π_u of the items, based on the chain rule, which is

$$p(\pi_u|\{\hat{x}_{ui}|i \in \mathcal{I}_u\}) = \prod_{i=1}^{|\mathcal{I}_u|} \frac{\lambda(\hat{x}_{u\pi_u^{-1}(i)})}{\sum_{j=i}^{|\mathcal{I}_u|} \lambda(\hat{x}_{u\pi_u^{-1}(j)})}, \tag{21}$$

where function $\lambda(\cdot)$ denotes a transformation function which should be increasing and strictly positive and $\pi_u^{-1}(i)$ denotes the item ranked at the i-th position of permutation π_u. Thus, when item i is ranked at position j in permutation π_u, we have the relation $\hat{x}_{ui} = \hat{x}_{u\pi_u^{-1}(j)}$ since $i = \pi_u^{-1}(j)$. For instance, suppose we have in total three items A, B, and C associated with user u. The probability of permutation $\pi_u = (A \succ B \succ C)$ is equal to the product of the following three probabilities (i.e., $P_{\pi_u} = P_1 P_2 P_3$), where

$$P_1 = \frac{\lambda(S_A)}{\lambda(S_A) + \lambda(S_B) + \lambda(S_C)}, \tag{22}$$

$$P_2 = \frac{\lambda(S_B)}{\lambda(S_B) + \lambda(S_C)}, \tag{23}$$

$$P_3 = \frac{\lambda(S_C)}{\lambda(S_C)} = 1. \tag{24}$$

According to Eq.(21), function f_u, which denotes the combination between the log probability of LTR and observation for user u, can be defined as

$$f_u = \sum_{i=1}^{|\mathcal{I}_u|} \left[\log \lambda(\hat{x}_{u\pi_u^{-1}(i)}) - \log \left(\sum_{j=i}^{|\mathcal{I}_u|} \lambda(\hat{x}_{u\pi_u^{-1}(j)}) \right) \right] - \alpha \sum_{i \in \mathcal{I}_u} \log(1 + e^{-\hat{x}_{ui}}).$$

Considering the item i ranked at position π_{ui} in permutation π_u, the sub-function f_{ui} that only relates item i in f_u can be defined as

$$f_{ui} = \log \lambda(\hat{x}_{ui}) - \sum_{j=1}^{\pi_{ui}} \log s_j^{|\mathcal{I}_u|} - \alpha \log(1 + e^{-\hat{x}_{ui}}),$$

where $s_j^{|\mathcal{I}_u|} = \sum_{p=j}^{|\mathcal{I}_u|} \lambda(\hat{x}_{u\pi_u^{-1}(p)})$. Considering f_{ui} as a function of $\hat{x}_{ui}$, we replace $\hat{x}_{ui}$ by x and rewrite f_{ui} as

$$f_{ui}(x) = \log \lambda(x) - \sum_{j=1}^{\pi_{ui}} \log \left(s_j^{|\mathcal{I}_u|} - \lambda(\hat{x}_{ui}) + \lambda(x) \right) - \alpha \log(1 + e^{-x}), \tag{25}$$

since we have

$$s_j^{|\mathcal{I}_u|} - \lambda(\hat{x}_{ui}) + \lambda(x) = s_j^{\pi_{ui}-1} + \lambda(x) + s_{\pi_{ui}+1}^{|\mathcal{I}_u|},$$

where $s_j^{\pi_{ui}-1} = \sum_{p=j}^{\pi_{ui}-1} \lambda(\hat{x}_{u\pi_u^{-1}(p)})$ and we define the conditions $s_{\pi_{ui}}^{\pi_{ui}-1} = 0$ and $s_{|\mathcal{I}_u|+1}^{|\mathcal{I}_u|} = 0$. The transformation function $\lambda(\cdot)$ can be a linear function or an exponential one, both of which will be derived in the following.

Time complexity analysis. Assume that a user consumes D items in average. Before computing the first order derivative of f_{ui}, to compute $h_{ui}(x)$, we compute $s_j^{|\mathcal{I}_u|}$ for each $j \in [1, |\mathcal{I}_u|]$, which takes $O(D)$ by calculating the cumulative sum of array $[\lambda(\hat{x}_{u\pi_u^{-1}(1)}), ..., \lambda(\hat{x}_{u\pi_u^{-1}(|\mathcal{I}_u|)})]$ with the direction from right to left. As such, for each (u, i) pair, computing f'_{ui} and f''_{ui} costs $O(D)$ respectively. When computing f'_{ui}, we obtain s_{ui} by selecting the item from $\mathcal{I}_u$ to minimize $f'_{ui}(\hat{x}_{ul})$ according to Eq. (15). Hence, from Algorithm 1, it takes $O(D^3)$ in Line 4 to find s_{ui} from candidate set $\mathcal{I}_u$ for item $i \in \mathcal{I}_u$. In other words, for each item, it takes $O(D^2)$ to find s_{ui} to obtain $f'_{ui}(s_{ui})$ and there are $O(D)$ items to be computed for each user approximately. It also takes $O(D^3)$ to compute f''_{ui} in Line 5. Letting E be the number of iterations for solving Lambert W function, it costs $O(DE)$ to solve Lambert W function for all item consumed by user u approximately. The time complexity of the rest of the algorithm is the same with the pair-wise version, that is, $O(X)$. The time complexity per iteration (Lines 3-12) is thus $O(MD(D^2 + E) + X)$. The algorithm is efficient since D is much smaller than X usually. Again, we can only consider top-F items per user rather than D ones per user to reduce the training complexity.

4.3.1 Linear transformation

When exploiting the linear transformation, the transformation function can be defined as

$$\lambda(x) = \delta x. \tag{26}$$

Function $f_{ui}(x)$ can be written as

$$f_{ui}(x) = \log \delta x - \sum_{j=1}^{\pi_{ui}} \log \left(s_j^{|\mathcal{I}_u|} - \delta \hat{x}_{ui} + \delta x \right) - \alpha \log(1 + e^{-x}). \tag{27}$$

The first order partial derivative of $f_{ui}(x)$ in terms of x is

$$f'_{ui}(x) = \frac{1}{x} - \delta \sum_{j=1}^{\pi_{ui}} h_{ui}(x) + \frac{\alpha}{1 + e^x}, \tag{28}$$

where $h_{ui}(x) = \frac{1}{s_j^{|\mathcal{I}_u|} - \delta \hat{x}_{ui} + \delta x}$. The second order partial derivative of $f_{ui}(x)$ in terms of x is

$$f''_{ui}(x) = -\frac{1}{x^2} + \delta^2 \sum_{j=1}^{\pi_{ui}} h_{ui}(x)^2 - \frac{\alpha e^x}{(1 + e^x)^2}. \tag{29}$$

4.3.2 Exponential transformation

When exploiting the exponential transformation, we have the transformation function

$$\lambda(x) = e^{\delta x}. \tag{30}$$

Function $f_{ui}(x)$ can be written as

$$f_{ui}(x) = \delta x - \sum_{j=1}^{\pi_{ui}} \log \left(s_j^{|\mathcal{I}_u|} - e^{\delta \hat{x}_{ui}} + e^{\delta x} \right) - \alpha \log(1 + e^{-x}). \tag{31}$$

The first order derivative of $f_{ui}(x)$ is

$$f'_{ui}(x) = \delta - \delta \sum_{j=1}^{\pi_{ui}} h_{ui}(x) + \frac{\alpha}{1 + e^x}, \tag{32}$$

where $h_{ui}(x) = \frac{e^{\delta x}}{s_j^{|\mathcal{I}_u|} - e^{\delta \hat{x}_{ui}} + e^{\delta x}}$. The second order partial derivative of $f_{ui}(x)$ in terms of $\hat{x}_{ui}$ is

$$f''_{ui}(x) = -\delta^2 \sum_{j=1}^{\pi_{ui}} h_{ui}(x)(1 - h_{ui}(x)) - \frac{\alpha e^x}{(1 + e^x)^2}. \tag{33}$$

5 Experiments

In this section, we evaluate the effectiveness of several learning to rank models based on the proposed framework empirically. We select two important tasks for recommender systems covering implicit count and explicit rating on real world datasets.

5.1 Datasets

We experiment on five datasets which are described as follows. The statistics of the datasets are shown in Table 3, where density denotes the matrix density $\frac{|\mathcal{X}^+|}{MN}$ and scale denotes the value range of an entry.

Last.fm1K. The dataset[3] is collected from Last.fm[4], an online music listening service, and contains 898,073 listening logs (user-artist-count triplet) from Feb. 2005 to May 2009 for nearly 992 users and 174,091 artists. In this dataset, the distributions of the number of artists listened per user is

3. http://ocelma.net/
4. http://www.last.fm

TABLE 3
The Statistics of the Datasets

dataset	Users	Items	Density	Scale
Last.fm1K	992	174,091	0.52%	1-26,496
Last.fm2K	1,892	17,632	0.278%	1-352,698
Last.fm360K	359,347	292,589	0.017%	1-419,157
MovieLens100K	943	1,682	6.304%	1-5
MovieLens1M	6,040	3,900	4.245%	1-5

very skewed. The users with large consumed items will cause heavy computational cost of the training.

Last.fm2K. The dataset[5] is released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). We use the dataset which is extracted from Last.fm. The dataset contains 92,834 listening logs made by 1,892 users and 17,632 artists. The range of listening logs covers [1, 352,698]. The number of artists listened by a user is limited in the range [40, 50] and the average is 49.067. In other words, there is no users with large consumed artists.

Last.fm360K. To test the performance of these models on a large-scale dataset, we conduct the experiment on Last.fm360K[6] which contains 17,559,530 listening logs made by 359,347 users and 292,589 items. Owning to the fact that a user consumes 166 items at most in the dataset, i.e., D is small, the three proposed model exhibit the learning efficiency.

MovieLens100K. GroupLens Research[7] collected the rating datasets through the MovieLens[8] web site during the seven-month period from September 19th, 1997 through April 22nd, 1998. The dataset contains 100,000 anonymous ratings of approximately 1,682 movies made by 943 MovieLens users who joined MovieLens in 2000. Users who had less than 20 ratings or did not have complete demographic information were removed from this dataset. Ratings are made on a 5-star scale.

MovieLens1M. The source is the same as the above whereas the dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users.

5.2 Competing Methods

We utilize four learning to rank models based on the proposed framework to connect with Poisson factorization.

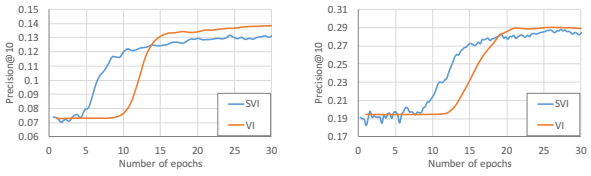
Point-wise PRPF (pointPRPF). To reduce the effect of outlier entries, for each observed entry, we assign a new value by Eq. (16) to its predicted score according to its ranking position in the user preference order to replace the corresponding ground-truth value.

Pair-wise PRPF (pairPRPF). We define the posteriori based on pair-wise learning to rank. During the training phase, we compute $f'_{ui}(\hat{x}_{ui})$ by Eq. (19), i.e., Line 4, and compute $f''_{ui}(\hat{x}_{ui})$ by Eq. (20), i.e., Line 6, in Algorithm 1.

List-wise PRPF with linear tranformation (listPRPF-linear). Recall that we have shown the two transformations of the Plackett-Luce model, which comprises exponential and linear transformations. We use the list-wise LTR model based on Plackett-Luce model with linear transformation. As such, during the training phase, $f'_{ui}(\hat{x}_{ui})$ and $f''_{ui}(\hat{x}_{ui})$ can be computed by Eq. (28) and Eq. (29) respectively.

List-wise PRPF with exponential tranformation (listPRPF-exp). Compared with listPRPF-linear, listPRPF-exp puts more emphasis on high-ranked items due to its exponential transformation. During the training phase, $f'_{ui}(\hat{x}_{ui})$ and $f''_{ui}(\hat{x}_{ui})$ can be computed by Eq. (32) and Eq. (33) respectively.

In addition, we use four methods for comparison: Non-negative Matrix Factorization (NMF) [17], Bayesian personalized ranking (BPR) [9], Hierarchical Poisson factorization (HPF) [4] and List-wise probabilistic matrix factorization (ListPMF) [18].



(a) Last.Fm2K (b) MovieLens100K

Fig. 5. The convergence of listPRPF-exp on various datasets with $K = 20$ and mini-batch size 50 users.

5.3 Experimental Settings

Data partition. Since the datasets cover a long period from months to years, the datasets are not suitable to be split by time-stamps. It is likely that a user did not keep active through a dataset. We follow [4] to select 20% of consumings/ratings in each dataset randomly to be used as a test set comprised of items that the user has consumed. In addition, we randomly select 1% of the consumings/ratings in each dataset as a validation set to determine whether the learning task should be terminated and to tune the prior hyperparameters.

Hyperparameter selection. To enforce sparsity, we set the hyperparameters of our framework (i.e., $[a, b, c, d, e, f]$) to 0.3. By following the experiemntal setting in [4], we set the shape hyperparameters of HPF (i.e., a, a', c and c') to provide exponentially shaped prior Gamma distributions and fix each of these hyperparameters at 0.3. Additionally, we set the rate hyperparameters of HPF b' and d' to 1. The hyperparameters, i.e., $[a, b, c, d, e, f]$ for our framework and $[a, a', b', c, c', d']$ for HPF, fix the prior mean at 1.

Initialization. The variational parameters are initialized to the prior on the corresponding latent variables and include small uniform noises as random offsets. We observe that the convergence speed is inversely proportional to the magnitude of the offset approximately. Empirically, we use a uniform noise covers $[0, m/100]$ where m denotes the initial value of a variable.

Stopping criterion. Since pair-wise learning to rank model compares the ranking positions of two items, some entries in $\boldsymbol{\beta}$ may be over-boosted during the training phase if some items are ranked high for most of users. In addition, a model may not fit testing dataset as well as the

5. http://grouplens.org/datasets/hetrec-2011/
6. https://archive.org/details/lastfm-dataset-360K
7. http://grouplens.org/datasets/movielens/
8. https://movielens.org/

training one. In order to alleviate the two situations, we use *validation-based early stopping*, which is commonly employed in the training of neural networks. We train the proposed methods on the training set and evaluate precision@10 on the validation set once in a while, which is determined by the size of dataset.

Learning rate for SVI. While employing stochastic variational inference, we refer to the works in [2], [19] and set the learning rate to $lr(t) = (\tau_0 + t)^{-\kappa}$, where t is the current iteration, τ_0 is set to 1 to down-weight early iterations and forgetting rate κ is set to 0.5.

5.4 Evaluation Criteria

Precision and recall. For each method, we select Z items with the highest predictive score $\boldsymbol{\theta}\boldsymbol{\beta}^\top$ for each user as the top Z recommendations during validation and testing. We compute precision@Z and recall@Z for user u, which can be defined respectively as

$$\text{Precision}_u@Z = \frac{|\mathcal{TP}_u|}{Z}, \quad \text{Recall}_u@Z = \frac{|\mathcal{TP}_u|}{|\mathcal{P}_u|},$$

where $\mathcal{TP}_u$ denotes the retrieved items that user has consumed and $\mathcal{P}_u$ denotes the items that user u has consumed. Thus, we can compute the mean precision@Z and recall@Z for all users respectively. Let $\mathcal{U}_t$ be the user set in the testing dataset. The mean precision@Z can be defined as $1/|\mathcal{U}_t| \sum_{u \in \mathcal{U}_t} \text{Precision}_u@Z$ and the recall@Z can be defined as $1/|\mathcal{U}_t| \sum_{u \in \mathcal{U}_t} \text{Recall}_u@Z$.

F_1 **score.** We use F_1 score to demonstrate the performance of these methods at various Z. F_1 score, the harmonic mean of precision and recall, is widely used to consider both precision and recall simultaneously. The F_1 score at Z top ranked items with respect to user u, denoted as $F_1\ \text{score}_u@Z$, can be defined as

$$F_1\ \text{score}_u@Z = 2\frac{\text{Precision}_u@Z \cdot \text{Recall}_u@Z}{\text{Precision}_u@Z + \text{Recall}_u@Z}.$$

Similarly, the mean $F_1\ \text{score}@Z$ can be defined as $1/|\mathcal{U}_t| \sum_{u \in \mathcal{U}_t} F_1\ \text{score}_u@Z$. To emphasize the difference between methods in terms of F_1 score, for each method, we subtract the F_1 score@Z of pairPRPF by the F_1 score@Z of the method to obtain *mean difference in* F_1 *score*@Z, that is

$$F_1\ \text{score}@Z - F_1\ \text{score}@Z\ (\text{pairPRPF}).$$

Since implicit count cannot represent graded relevance value, nDCG is not used for evaluation.

5.5 Results and Discussion

Convergence. We use precision@10 to demonstrate the convergence of learning. Figure 5 shows the convergence of precision@10 on LastFm2K and MovieLens100K datasets respectively. The orange lines show the convergence of listPRPF-exp with latent variables updated by VI. The scores on precision@10 decay slightly after the model achieves the highest score on precision@10. This situation can be alleviated by using validation-based early stopping.

The blue lines in Figure 5 demonstrate the convergence of user-wise stochastic mini-batch listPRPF-exp where batch size is set to 50 users. Compared with the batch version, the stochastic mini-batch version reaches lower scores on precision@10. Even so, the stochastic mini-batch version can analyze large size data owing to the efficiency [19].

TABLE 4
The Results on Last.fm360K with 100 Latent Factors

	P@5	P@10	P@15	P@20
HPF	10.86%	8.90%	7.81%	7.07%
pairPRPF	12.28%	10.03%	8.68%	7.79%
listPRPF-linear	12.52%	10.07%	8.82%	7.90%
listPRPF-exp	**12.81%**	**10.31%**	**8.83%**	**7.94%**
	R@5	R@10	R@15	R@20
HPF	5.29%	8.66%	11.44%	13.80%
pairPRPF	6.04%	9.86%	12.76%	15.29%
listPRPF-linear	6.15%	9.88%	12.89%	15.52%
listPRPF-exp	**6.30%**	**10.14%**	**13.11%**	**15.59%**

5.5.1 The results on Last.fm360K dataset

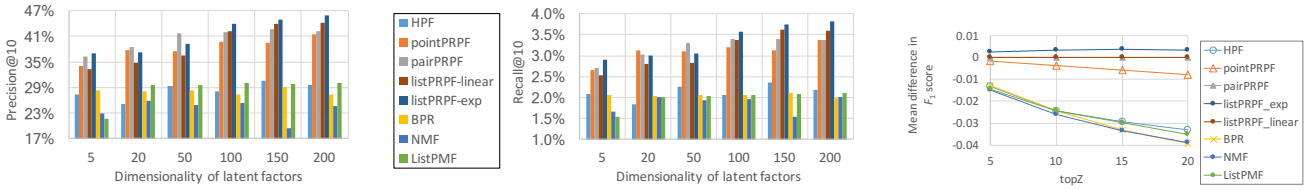
One of the advantages of the Poisson factorization is that the optimizer in the learning phase can only focus on explicit evaluations, i.e., nonzero entries, thus making the approach linear to the number of the data. By contrast, the other state-of-the-arts for comparison in the paper are linear in the size of the whole matrix. As a result, owning to the computational cost, we only compare HPF-based methods which are feasible for scalable recommendation.

The performance of the methods based on hierarchical Poisson factorization (HPF) on Last.fm360K dataset is shown in Table 4. The three proposed methods outperform HPF in terms of both precision and recall. In addition, the list-wise methods can reach higher scores on both precision and recall, especially on precision@5 (improved by 1.98%) and recall@5 (improved by 1.88%). For pairPRPF, the positional information is not considered in its posteriori, and comparatively, the list-wise approaches i.e., listPRPF-linear and listPRPF-exp, consider the ranked list made up of the entire items consumed by the target user. By doing so, positional information is introduced in their posteriori so that they pay more attention on the ranking probabilities with respect to top-ranked items in the Plackett-Luce model. As a result, list-wise approaches exhibit better performance in top-ranked recommendation.

5.5.2 The results on Last.fm1K dataset

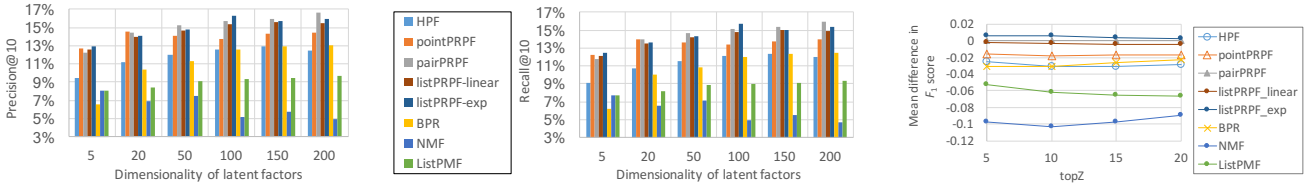
The performance of the competing methods on Last.fm1K is shown in Figure 6. The methods based on the proposed framework, i.e., pairPRPF, listPRPF-linear and listPRPF-exp, reach the highest scores in terms of both precision@10 and recall@10 for all K and also attain the highest F_1 score@Z for all Z. Except for pointPRPF, the models based on our framework outperform the other methods by at least 28% on precision@10 and 30% on recall@10. Since the distribution of the values of observed entries follows a power-law distribution approximately, the ranking-based methods outperform the regression-based ones.

In addition, listPRPF-exp outperforms pairPRPF when the dimensionality of latent factors is large, i.e., $K \geq 100$, and moreover, reaches 45.8% scores on precision@10 and 3.8% scores on recall@10 with $K = 200$. Since users' favorite items are diverse and need more latent factors to model, listPRPF-exp, which considers the permutation of the consumed items, can sufficiently utilize these latent factors. By contrast, the factorization of pairPRPF, which only focuses



(a) Precision@10 for various K. (b) Recall@10 for various K. (c) Difference in F_1 score with $K = 100$.

Fig. 6. The results on Last.fm1K.



(a) Precision@10 for various K. (b) Recall@10 for various K. (c) Difference in F_1 score with $K = 100$.

Fig. 7. The results on Last.fm2K.

on the ranking relationship between any consumed item pair, may be disturbed during the training phase. In this dataset, the average of the number of consumed items per user is 706.51 and hence the effect of permutation of consumed items is obvious. In contrast, in the next dataset, Last.fm2K, the effect is slight since a user consumes 49.067 items in average.

In Figure 6 (c), compared with pairPRPF, the performance of the PRPF-framework-based models maintains with the increasing of Z, while the others are getting worse. On an implicit count dataset with outliers, regression-based methods would fit those high-ranked items with very large ground-truth values for each user merely and neglect the user preference order of consumed items. In contrast, our methods leverage the user preference order to raise the ranking positions of favorable items rather than focusing on top-ranked ones merely. Hence, the proposed framework can factorize the utility more reasonably.

5.5.3 The results on Last.fm2K dataset

Figure 7 shows the performance of all methods on Last.fm2K. Except for pointPRPF, the PRPF-framework-based models outperform the other methods in terms of both precision@10 and recall@10 by at least 19% for all K. BPR outperforms HPF only when $K > 150$. BPR, which merely separates observed entries from unobserved ones and regards observed entries as equally important, will not easily represent a utility matrix by fewer latent factors since the information of preference order of observed entries is neglected. By contrast, HPF considers the variety of the values of observed entries but cannot mitigate the problem of outliers of entries so that it does not outperform BPR.

Not like the performance on Last.fm1K, ListPMF does not conduct satisfied results on Last.fm2K dataset. Since the dataset is sparser than Last.fm1K, as a derived model from PMF, ListPMF cannot tackle the sparsity of utility matrix with count data well even if ListPMF overcomes the outliers of entries by employing the list-wise LTR-based objective function. The PRPF-framework-based models that utilize list-wise learning to rank, comprising listPRPF-linear and listPRPF-exp, do not outperform pairPRPF. The major reason is the fewness of the number of items consumed by a user in average. Since the number of items consumed by a user is few, the advantage of list-wise ranking does not reveal obviously.

In Figure 7 (c), the three PRPF-framework-based models can keep the performance with the increasing of Z. Notice that the performance of NMF seems to be getting better when $Z \geq 10$. We explain that poor performance and too few testing data cause this phenomenon. In this dataset, there are only 10.4 items for testing per user. Since precision of NMF is about 7%, only 0.7 items in average will be retrieved per user. Thus, when a relevant item is retrieved by increasing Z from 15 to 20, the decreasing of precision@Z increases slightly, i.e., from $0.7/15$ to $1.7/20$, and recall@Z increases conspicuously, i.e., from $0.7/10.4$ to $1.7/10.4$.

The three experiments on implicit count datasets, i.e., Last.fm360K, Last.fm1K and Last.fm2K, demonstrate two important aspects stated as follows. First, outlier entries will impair the performance of classical regression-based MF methods. When outlier entries exist, the likelihood will be intensely depressed, thus forcing the optimizer to fit them preferentially. Second, a method that can avoid outliers will perform well even if the method utilizes a trivial approach, e.g., pointPRPF, to reduce the effect of outliers in a utility matrix. Regression-based methods such as HPF and NMF perform worse than PRPF because they have to fit the value for each entry. Moreover, the pair-wise and list-wise PRPF-based models perform better than pointPRPF since the relative order between documents cannot be naturally considered by pointPRPF during the training phase.

5.5.4 The results on MovieLens100K dataset

The performance on MovieLens100K dataset is shown in Figure 8. The three PRPF-framework-based models reach the highest scores on precision@10, recall@10 and F_1 score in most cases, followed by HPF and pointPRPF. The performance of HPF is slightly better than the performance

of pointPRPF. PointPRPF, a trivial learning-to-rank-based method, may not preserve the user preference orders and the separation between observed and unobserved entries simultaneously on a dataset with limited scale.

NMF conducts satisfied results when K is small. Since the dataset is rating (i.e., with limited scale in $[1, 5]$) rather than consuming logs, to model the utility matrix, it is more reasonable by assuming Gaussian entries than by assuming Poisson ones. Moreover, compared with other datasets, MovieLens100K is denser so that the drawback of NMF that sparse matrix cannot be well-modeled does not reveal apparently. Hence, NMF can outperform other methods on this dataset. BPR achieves lower scores than PF-based methods in terms of precision@10 and recall@10. Since BPR lacks the mechanic to model user preference orders on observed entries, the performance of BPR is better than ListPMF but worse than HPF.

On the contrary, ListPMF only considers user preference orders among the consumed items. Due to the lack of terms for unobserved entries in its objective function, all unobserved entries can be arbitrary values. Although it is suggested not to put more total emphasis on unobserved entries due to the sparsity of a utility matrix from the prior discussion in [4], a MF model should not put nothing on unobserved entries absolutely. As a result, in addition to observed entries, a LTR-based MF model should also consider unobserved ones so as to deal with sparsity of a utility matrix. In PRPF-framework-based models, the posteriori integrates the LTR and Poisson factorization, thus dealing with sparse matrix effectively.

5.5.5 The results on MovieLens1M dataset

The performance on MovieLens1M dataset is shown in Figure 9. Similarly, listPRPF-exp outperforms the other methods followed by HPF with a slight distance with larger K. NMF outperforms BPR in most cases. We explain that modeling user preference orders is more important than separating observed entries from unobserved ones on the dataset. Since the scale of values in the dataset is limited, ListPMF cannot exploit the LTR-based objective function to be insensitive to the outliers. Compared with PRPF-framework-based models, the F_1 scores of all other methods decay with the increasing of Z in Figure 9(c).

Recall that pointPRPF outperforms HPF due to the capability of insensitivity to the outliers in a utility matrix. Hence, pointPRPF does not outperform HPF since there is no outliers in the two rating datasets. Consequently, PRPF-framework-based models outperform other methods obviously on consuming logs due to the insensitivity to outliers. On ratings data, PRPF-framework-based models has the performance similar to HPF and outperform pointPRPF due to the LTR-based posteriori.

6 Conclusion

In this paper, we propose the framework of personalized ranking on Poisson factorization, which can be adaptive to LTR problem as well as the traditional matrix factorization one. We point out three critical objectives that a matrix factorization method should achieve: the separation between observed and unobserved entries, user preference order and insensitivity to outliers. We define the Bayesian formulation by maximizing the posterior probability to model personalized pair-wise ranking relationship so as to *preserve user preference order* and be *insensitive to outliers*. Moreover, we also introduce *probability of observation* to separate observed entries from unobserved ones in a utility matrix.

Consequently, our framework have three advantages. First, the LTR-based objective function can be viewed as a relaxation of the regression-based objective function, and hence our model can achieve promising results by neglecting unnecessary objectives. Second, compared with regression-based one, our framework is more insensitive to outlier entries in a utility matrix. Third, our framework can be adaptive to any LTR model which has 1st and 2nd order partial derivatives. From the experimental results, the three objectives play critical roles. Hence, the proposed framework, the only method to achieve the three objectives, outperforms state-of-the-art methods and outputs convincing results.

Appendix

LEMMA 1. *$f_{ui}(x)$ defined in* Eq. (9) *is concave for all user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$.*

Proof. Since $-\log(1+e^{-(b-x)})$ is a concave function of x for any $b \in \mathbb{R}$, functions $-\log(1 + e^{-d_{uj}(x)})$ for all user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ and $-\log(1 + e^{-x})$ are concave. Moreover, since $C/|\mathcal{I}_u| > 0$ and $\alpha > 0$, $f_{ui}(x)$, the positive weighted sum of concave functions, is also concave. □

THEOREM 1. *Function $\mathcal{F}_{ui}(x)$ in* Eq. (8) *and its approximation $g_{ui}(x)+\log p(x|\lambda)$* have their global maxima somewhere in the interval $[\min(s,t), \max(s,t)]$.

Proof. Since continuous Poisson distribution is log-concave, $\log p(x|\lambda)$ and $f_{ui}(x)$ are concave. Thus, $\mathcal{F}_{ui}(x)$ has a global maximum somewhere in the interval $[\min(s,t), \max(s,t)]$. Moreover, since f_{ui} and g_{ui} share the 1st and 2nd derivatives, g_{ui} also has a global maximum at s. Owing to the concavity of g_{ui}, $g_{ui} + \log p(x|\lambda)$ also has a global maximum somewhere in the interval $[\min(s,t), \max(s,t)]$. □

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] P. Gopalan, L. Charlin, and D. M. Blei. Content-based recommendations with Poisson factorization. NIPS 2014: 3176-3184

[3] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. M. Blei. Bayesian Nonparametric Poisson Factorization for Recommendation Systems. AISTATS 2014: 275-283

[4] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable Recommendation with Hierarchical Poisson Factorization. UAI 2015.

[5] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. Introduction to variational methods for graphical models. Machine Learning, 37: 183-233, 1999.

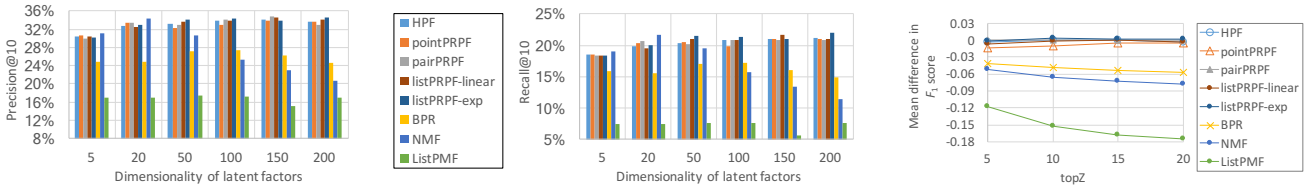
[6] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2008.

[7] Y. Cao, J. Xu, T. Y. Liu, H. Li, Y. Huang, and H. W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR*, 2006.

[8] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

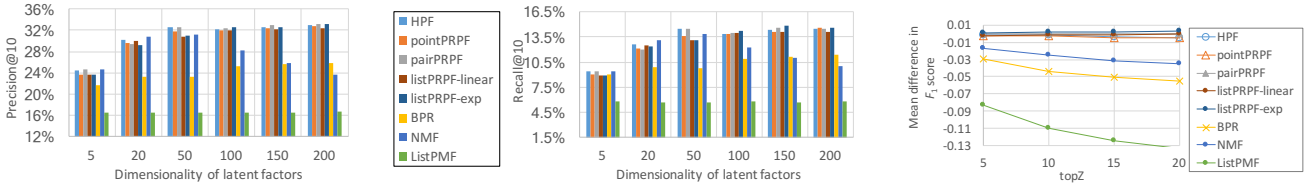
[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[10] C C. Johnson. Logistic Matrix Factorization for Implicit Feedback Data. In *NIPS*, 2014.



(a) Precision@10 for various K. (b) Recall@10 for various K. (c) Difference in F_1 score with $K = 100$.

Fig. 8. The results on MovieLens100K.



(a) Precision@10 for various K. (b) Recall@10 for various K. (c) Difference in F_1 score with $K = 100$.

Fig. 9. The results on MovieLens1M.

[11] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank using Gradient Descent. In *ICML*, 2005.

[12] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. JMLR, 2003.

[13] C. J. Burges, R. Ragno, and Q. V. Le. Learning to Rank with Non-Smooth Cost Functions. In *NIPS*, 2006.

[14] J. H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, 2001.

[15] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting Boosting for Information Retrieval Measures. Journal of Information Retrieval, 2007.

[16] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, Learning to Rank by Optimizing NDCG Measure. In *NIPS*, 2010.

[17] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 1999.

[18] J. Liu, C. Wu, Y. Xiong, and W. Liu. List-wise probabilistic matrix factorization for recommendation. Information Sciences, 2014.

[19] M. D. Hoffman, D. M. Blei, C. Wang, and J. W. Paisley. Stochastic variational inference. JMLR, 2013.

[20] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys*, 2010.

[21] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *International Conference on Trust Management*, 2005.

[22] Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

[23] Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., and Li, H. (2007, June). Learning to rank: from pairwise approach to listwise approach. In Proceedings of the 24th international conference on Machine learning (pp. 129-136). ACM.

[24] Xia, F., Liu, T. Y., Wang, J., Zhang, W., and Li, H. (2008, July). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning, ICML* (pp. 1192-1199). ACM.

[25] L.-Y. Kuo, C.-K. Chou and M.-S. Chen. Personalized Ranking on Poisson Factorization. In *SDM*, 2018.

[26] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua. Neural collaborative filtering. In *WWW*, 2017.

[27] S. Zhang, L. Yao, and A. Sun, Deep learning based recommender system: A survey and new perspectives. *arXiv preprint*, 2017.

[28] W. Zhang, T. Chen, J. Wang, and Y. Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In SIGIR, 2013.

[29] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In SIGIR, 2017.

[30] D. Park and Y. Chang. Adversarial Sampling and Training for Semi-Supervised Information Retrieval. arXiv preprint, 2018.

Li-Yen Kuo received the master's degree from the Department of Department of Computer Science at National Chiao Tung University, Hsinchu, Taiwan, in 2012. He is now a PhD student in the Department of Electrical Engineering at National Taiwan University, Taiwan. His main research interests include recommender systems and Bayesian graphical models.

Chung-Kuang Chou received the BS degree from the Department of Communication Engineering at National Central University, Taoyuan, Taiwan, in 2010 and the PhD degree from the Department of Electrical Engineering at National Taiwan University, Taiwan, in 2017. His main research interests include data mining and social influence analysis.

Ming-Syan Chen received the PhD degree in computer, information, and control engineering from the University of Michigan, Ann Arbor, Michigan. He is now the executive vice president and also a distinguished professor in the EE Department, National Taiwan University. He was a research staff member with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996, the director of the Graduate Institute in Communication Engineering, NTU, from 2003 to 2006, the president/CEO of the Institute for Information Industry (III), from 2007 to 2008, and also the director and a distinguished research fellow of the Research Center of Information Technology Innovation (CITI) in the Academia Sinica from 2008 to 2015. His research interests include databases, data mining, social networks, and multimedia networking, and he has published more than 350 papers in his research areas. He is a national chair professor of the Ministry of Education in Taiwan. He served as the editor-in-chief and editors for many major journals and received numerous awards for his research, teaching, inventions, and patent applications. He is a fellow of the ACM and the IEEE.