Sinkhorn Collaborative Filtering

Xiucheng Li
Nanyang Technological University
xli055@e.ntu.edu.sg

Jin Yao Chin
Nanyang Technological University
S160005@e.ntu.edu.sg

Yile Chen
Nanyang Technological University
yile001@e.ntu.edu.sg

Gao Cong
Nanyang Technological University
gaocong@ntu.edu.sg

ABSTRACT

Recommender systems play a vital role in modern web services. In a typical recommender system, we are given a set of observed user-item interaction records and seek to uncover the hidden behavioral patterns of users from these historical interactions. By exploiting these hidden patterns, we aim to discover users' personalized tastes and recommend them new items. Among various types of recommendation methods, the latent factor collaborative filtering models have dominated the field. In this paper, we develop a unified view for the existing latent factor models from a probabilistic perspective. The unified framework enables us to discern the underlying connections of different latent factor models and deepen our understandings of their advantages and limitations. In particular, we observe that the loss functions adopted by the existing models are oblivious to the geometry induced by the item-similarity. To address this, we propose a novel model—SinkhornCF—based on Sinkhorn divergence. To address the challenge of the expensive computational cost of Sinkhorn divergence, we also propose new techniques to enable the resulting model to be able to scale to large datasets. Its effectiveness is verified on two real-world recommendation datasets.

KEYWORDS

Latent factor models, probabilistic generative models, Sinkhorn divergence

ACM Reference Format:
Xiucheng Li, Jin Yao Chin, Yile Chen, and Gao Cong. 2021. Sinkhorn Collaborative Filtering. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3449869

1 INTRODUCTION

Recommender systems play a vital role in modern web services. They help the websites direct to the users the items–movies, songs, articles, products–that they may like and help the users to more effectively interact with the Web. In a recommender system, we are given a set of observed user-item interaction records and seek to uncover the hidden behavioral patterns of users from these historical interactions. By exploiting these hidden patterns, we aim to discover their personalized tastes and recommend them the unconsumed items that they might like in the future.

Motivated by their great usefulness, various types of recommendation models have been developed in the past decades [5, 14, 17, 19, 30, 32, 33]. In particular, the latent factor collaborative filtering or latent factor models are the most extensively studied and widely adopted ones [10, 14, 30, 33]. The latent factor models associate each user a latent vector of preferences and each item a latent vector of attributes, which represent a user's personalized preference and an item's unique attribute in a low dimensional space. These latent representations are learned from partial observed user-item interactions (*e.g.*, rating products, clicking pages or reading posts) and then they are used to predict the users' preferences on those unseen items. Matrix factorization (MF), arguably the most popular latent factor CF model, associates a user u and an item i with latent vectors $\boldsymbol{\xi}_u$ and $\boldsymbol{\lambda}_i$, respectively. MF looks for the best parameters by minimizing the discrepancy between the inner product $\boldsymbol{\xi}_u^\top \boldsymbol{\lambda}_i$ and corresponding observation. MF (and its variants) were the de facto models in real-world deployment and hit the top ranks in the Netflix prize [33]. With the progress of deep learning algorithms, many other latent factor models were developed to harvest the nonlinear representation power of deep neural nets [8, 13, 18, 34, 36]. For example, NeuMF replaces the linear interaction $\boldsymbol{\xi}_u^\top \boldsymbol{\lambda}_i$ with a nonlinear function (MLP) [8, 13], and MultVAE learns the user representations directly from their past interaction records using a deep neural net [18].

In this paper, we provide a unified view for the various collaborative filtering models proposed in the literature from a probabilistic perspective. More specifically, we unify different collaborative filtering models by interpreting the latent representations $\boldsymbol{\xi}_u$ and $\boldsymbol{\lambda}_i$ being drawn from their underlying representation distributions; the representations are then fed into a transformation function $f(\cdot, \cdot)$ to obtain the parameters of the observation distribution, conditional on which we generate the observations. The benefit is two-fold: 1) it enables us to draw connections between different models to deepen our understandings; 2) it enables us to analyze their advantages and limitations from a new perspective, based on which we propose a novel solution to address the limitations of the existing models. In particular, we observe that almost all of the existing latent factor collaborative filtering models resort to minimize the Euclidean distance or Kullback-Leibler divergence between the empirical distribution and model-defined distribution, which is oblivious to the geometry induced by the item-similarity and thus might yield undesirable penalty towards different model outputs. To address this, we propose a novel model—SinkhornCF—based on Sinkhorn divergence, which induces fine topology by considering

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449869

the item-similarity, and develop it an efficient learning algorithm. The resulting model is both item-similarity-aware and scales to large datasets. In summary, our contributions are as follows.

- We provide a unified view for the various latent factor collaborative filtering models proposed in the literature, which helps us to discern their underlying connections and deepen our understandings on them.
- The proposed unified framework enables us to analyze the advantages and limitations of the existing latent factor models from a new perspective; in particular, we note that almost all of them attempt to minimize the loss functions that are oblivious to the geometry induced by item similarity, which tend to produce undesirable models.
- To address this, we propose a novel model—SinkhornCF—based on Sinkhorn divergence. The resulting model is able to not only be aware of the item-similarity in its loss and but also scale to large datasets. Its effectiveness is verified on two real-world recommendation datasets.

2 METHODOLOGY

2.1 Notations and definitions

We use $u \in \{1, 2, \ldots, U\}$ to index users and $i \in \{1, 2, \ldots, I\}$ to index items, where U, I are the number of users and items, respectively, and use $\mathbb{R}_+ = \{x | x \geq 0, x \in \mathbb{R}\}$ to denote the non-negative real numbers. $\mathbf{X} \in \{0,1\}^{U \times I}$ represents the user-item interaction matrix (*e.g.,* click, watch, check-in, etc.) and $\mathbf{x}_u = [x_{u1}, x_{u2}, \ldots, x_{uI}]$ is a one-dimensional binary vector (u-th row of $\mathbf{X}$) whereby $x_{ui} = 1$ implies that user u has interacted with item i. In addition, we use $\langle \mathbf{a}, \mathbf{b} \rangle$ to represent the inner product of two vectors $\mathbf{a}, \mathbf{b}$ or Frobenius inner product for two matrices $\mathbf{A}, \mathbf{B}$, *i.e.,* $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}^\top \mathbf{B})$. $\mathbb{1}(\cdot)$ denotes the indicator function, $[I]$ indicates the natural number set $\{1, 2, \ldots, I\}$ and $\mathrm{nz}(\mathbf{x})$ denotes the number of nonzero entries in a vector $\mathbf{x}$.

$$\mathbf{X}^\top = \underbrace{\begin{pmatrix} \cdots & x_{u1} & \cdots \\ \cdots & x_{u2} & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & x_{uI} & \cdots \end{pmatrix}}_{u=1,2,\ldots,U}, \quad \mathbf{x}_u^\top = \begin{bmatrix} x_{u1} \\ x_{u2} \\ \vdots \\ x_{uI} \end{bmatrix}.$$

- **Positive Items.** The positive items for a given user are the ones that the user provides a positive feedback.
- **True Negative Items.** The true negative items for a given user are the ones that the user provides no feedback and indeed dislikes.
- **False Negative Items.** The false negative items for a given user are the ones that the user provides no feedback but actually likes.

The above definitions are all specific to a given user and an item could be positive to a user u_1 but negative to another user u_2.

Problem Setting. In this paper, we consider top-n recommendation without using side information under the implicit feedback setting, in which only positive items (one entries) are observed and the unobserved entries (zero entries) are either true negative items or false negative items.

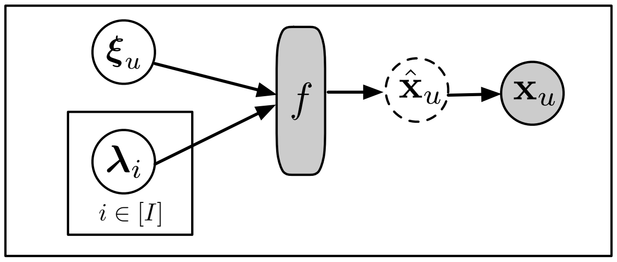

Figure 1: The graphical model of the proposed generative process, which describes the generation of observed interaction $\mathbf{x}_u$ for a particular user u. f is the transformation function of user and item representations, $\hat{\mathbf{x}}_u$ is the parameter of observation sampling distribution $p_x(\mathbf{x}_u|\hat{\mathbf{x}}_u)$, and thus is indicated in dash line.

2.2 A unified probabilistic view of CF models

The latent factor models attempt to find a distributed representation for every user u and item i, denoting as $\boldsymbol{\xi}_u \in \mathbb{R}^k$ and $\boldsymbol{\lambda}_i \in \mathbb{R}^k$, which represent user u's preference and item i's attribute in a low dimensional space, respectively. This is achieved via minimizing a particular loss function, $\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u)$, which takes as inputs the observation $\mathbf{x}_u \in \mathbb{R}^I$ and model output $\hat{\mathbf{x}}_u \in \mathbb{R}^I$, by using the variants of gradient descent method [3]. In this paper, we propose a probabilistic generative process to unify such latent factor models as follows.

- Draw user representation $\boldsymbol{\xi}_u \sim p_u(\boldsymbol{\xi})$;
- For $i \in \{1, 2, \ldots, I\}$,
  - Draw item representation $\boldsymbol{\lambda}_i \sim p_i(\boldsymbol{\lambda})$;
  - Transform user-item pair $(\boldsymbol{\xi}_u, \boldsymbol{\lambda}_i)$ via $f(\cdot, \cdot)$;
- Draw observation $\mathbf{x}_u \sim p_x(\mathbf{x}_u|\hat{\mathbf{x}}_u)$.

Here $p_u(\boldsymbol{\xi})$ and $p_i(\boldsymbol{\lambda})$ denotes the user and item underlying representation distribution, respectively, $f(\cdot, \cdot)$ is a transformation function that could be either a linear function (*e.g.,* inner product) or a nonlinear function (*e.g.,* deep neural network), and $p_x(\mathbf{x}_u|\hat{\mathbf{x}}_u)$ is the sampling distribution for the observation $\mathbf{x}_u$, whose parameters are specified by $\hat{\mathbf{x}}_u \overset{\text{def.}}{=} [f(\boldsymbol{\xi}_u, \boldsymbol{\lambda}_1), \ldots, f(\boldsymbol{\xi}_u, \boldsymbol{\lambda}_I)] \in \mathbb{R}^I$. The observation sampling distribution $p_x(\mathbf{x}_u|\hat{\mathbf{x}}_u)$ has a close relationship with the loss function $\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u)$ and, as we will show, different observation sampling distributions induce distinct loss functions. The graphical model for the generative process is shown in Figure 1.

We are now ready to recover the existing latent factor collaborative filtering models developed in the recommendation community under our proposed probabilistic framework. The key insight is that the existing latent factor models differ from each other only by the underlying representation distributions $p_u(\boldsymbol{\xi})$, $p_i(\boldsymbol{\lambda})$, transformation function $f(\cdot, \cdot)$ and observation sampling distribution $p_x(\mathbf{x}_u|\hat{\mathbf{x}}_u)$. In what follows, we examine several representatives of latent factor collaborative filtering models under this unified probabilistic view. For ease of reference, the examined models are summarized in Table 1.

Table 1: Summary of representational CF models

Name	$p_u(\xi)$	$p_i(\boldsymbol{\lambda})$	f	$p(\mathbf{x}_u\|\hat{\mathbf{x}}_u)$	Inductive over users	Minimized loss
MF	$\frac{1}{U}\sum_{u=1}^{U}\delta(\xi-\xi_u)$	$\frac{1}{I}\sum_{i=1}^{I}\delta(\boldsymbol{\lambda}-\boldsymbol{\lambda}_i)$	inner product	Gaussian	✗	Euclidean
WMF	$\frac{1}{U}\sum_{u=1}^{U}\delta(\xi-\xi_u)$	$\frac{1}{I}\sum_{i=1}^{I}\delta(\boldsymbol{\lambda}-\boldsymbol{\lambda}_i)$	inner product	Gaussian	✗	Euclidean
NeuMF	$\frac{1}{U}\sum_{u=1}^{U}\delta(\xi-\xi_u)$	$\frac{1}{I}\sum_{i=1}^{I}\delta(\boldsymbol{\lambda}-\boldsymbol{\lambda}_i)$	MLP	Bernoulli	✗	KL-Divergence
BPF	Gamma(a,η_u)	Gamma(c,η_i)	inner product	Poisson	✗	KL-Divergence
MultVAE	Gaussian(ξ)	$\delta(\boldsymbol{\lambda}-\mathbf{0})$	MLP	Multinomial	✓	KL-Divergence

Classical Matrix Factorization Model. The classical matrix factorization model—MF—can be derived from the proposed probabilistic framework by specifying the underlying representation distributions as $p_u(\xi)=\frac{1}{U}\sum_{u=1}^{U}\delta(\xi-\xi_u)$, and $p_i(\boldsymbol{\lambda})=\frac{1}{I}\sum_{i=1}^{I}\delta(\boldsymbol{\lambda}-\boldsymbol{\lambda}_i)$ where $\delta(\cdot)$ denotes the Dirac delta function (*i.e.,* the probability density is only concentrated at a collection of discrete points), the transformation function as inner product $f(\mathbf{a},\mathbf{b})=\langle\mathbf{a},\mathbf{b}\rangle$ and the observation sampling distribution to be the Gaussian distribution with fixed variance σ^2,

$$p\left(\mathbf{x}|\boldsymbol{\mu},\sigma^2\right)=\frac{1}{(2\pi)^{I/2}\sigma^I}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{I}(x_i-\mu_i)^2\right\}, \quad (1)$$

in which the parameter $\boldsymbol{\mu}$ corresponds to $\hat{\mathbf{x}}_u$. To see this, substituting $\mathbf{x}$ with $\mathbf{x}_u$ and $\boldsymbol{\mu}$ with $\hat{\mathbf{x}}_u$ in Equation 1, and calculating the negative log-likelihood of $p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u,\sigma^2\right)$ yields the loss function

$$\ell(\mathbf{x}_u,\hat{\mathbf{x}}_u)=\sum_{i=1}^{I}\left(x_{ui}-f(\xi_u,\boldsymbol{\lambda}_i)\right)^2. \quad (2)$$

The weighted matrix factorization model (WMF) [14] generalizes MF by associating a weight w_i to item i to down-weight the negative items, *i.e.,*

$$\ell(\mathbf{x}_u,\hat{\mathbf{x}}_u)=\sum_{i=1}^{I}w_i\left(x_{ui}-f(\boldsymbol{\xi}_u,\boldsymbol{\lambda}_i)\right)^2. \quad (3)$$

This can be obtained by introducing item-dependent variance σ_i^2 rather than a fixed σ^2 in Equation 1.

Neural Matrix Factorization. Neural matrix factorization model (NeuMF) [8, 13] generalizes the classical matrix factorization by substituting the inner product with a nonlinear deep neural network. NeuMF can be easily recovered from our probabilistic framework by setting the transformation function f to be a Multilayer Perceptron with a sigmoid activation output layer, *i.e.,* $f(\mathbf{a},\mathbf{b})=\mathrm{MLP}(\mathbf{a}||\mathbf{b})$, and choosing the Bernoulli distribution to be the observation sampling distribution

$$p(\mathbf{x}|\boldsymbol{\theta})=\prod_{i=1}^{I}\theta_i^{x_i}(1-\theta_i)^{1-x_i}, \quad (4)$$

in which the parameter $\boldsymbol{\theta}$ corresponds to the model output $\hat{\mathbf{x}}_u$. By replacing $\mathbf{x}$ with $\mathbf{x}_u$ and $\boldsymbol{\theta}$ with $\hat{\mathbf{x}}_u$ in Equation 4, and calculating its negative log-likelihood we have

$$\ell(\mathbf{x}_u,\hat{\mathbf{x}}_u)=-\sum_{i=1}^{I}x_{ui}\log f(\boldsymbol{\xi}_u,\boldsymbol{\lambda}_i)-\sum_{i=1}^{I}(1-x_{ui})\log(1-f(\boldsymbol{\xi}_u,\boldsymbol{\lambda}_i)). \quad (5)$$

NeuMF shares the same underlying representation distributions $p_u(\xi)$ and $p_i(\boldsymbol{\lambda})$ with MF.

Bayesian Poisson Factorization. Bayesian Poisson factorization (BPF) developed in [10, 11] is a hierarchical collaborative filtering model. It models the observation with a Poisson distribution, and unlike MF and NeuMF, BPF is a true Bayesian probabilistic model because it draws its user and item representations from two Gamma prior distributions rather than the degraded point distributions specified by Dirac functions. Putting it in another way, the user and item representations are random variables in BPF and are learned from Bayesian inference. BPF has several nice properties such as it could introduce inductive bias to explicitly model user activities and item popularities, and it automatically down-weights the impact of negative items [10]. We could recover BPF under our probabilistic framework by setting the underlying representation distributions to be Gamma(a,η_u) and Gamma(c,η_i) for each user and item, respectively, letting the transformation function $f(\mathbf{a},\mathbf{b})=\langle\mathbf{a},\mathbf{b}\rangle$, and using the Poisson distribution as the observation sampling distribution,

$$p(\mathbf{x}|\boldsymbol{\theta})=\prod_{i=1}^{I}\frac{1}{x_i!}\exp\{x_i\log\theta_i-\theta_i\}, \quad (6)$$

in which the parameter $\boldsymbol{\theta}$ corresponds to the model output $\hat{\mathbf{x}}_u$. Substituting $\mathbf{x}$ with $\mathbf{x}_u$ and $\boldsymbol{\theta}$ with $\hat{\mathbf{x}}_u$ in Equation 6, and calculating its negative log-likelihood produces,

$$\ell(\mathbf{x}_u,\hat{\mathbf{x}}_u)=-\sum_{i=1}^{I}x_{ui}\log f(\xi_u,\boldsymbol{\lambda}_i)+\sum_{i=1}^{I}f(\xi_u,\boldsymbol{\lambda}_i). \quad (7)$$

The second term $\sum_{i=1}^{I}f(\xi_u,\boldsymbol{\lambda}_i)$ corresponds to the summation of parameter $\boldsymbol{\theta}$ of the Poisson distribution.

Multinomial Variational Auto-encoders. Multinomial variational auto-encoders, MultVAE, explore the usage of deep generative models for collaborative filtering [18]. MultVAE comprises of an encoder that maps a user's partial interactions into a latent representation distribution and a decoder that attempts to reconstruct the completed user interactions using the latent representation drawn from the latent distribution. MultVAE also falls into the Bayesian probabilistic family and approximates the posterior distribution with a neural network, and it is optimized by maximizing the evidence lower bound (ELBO) of log-likelihood, which corresponds to loss function consisting of a reconstruction loss and a KL-divergence regularization term [18]. We could derive MultVAE by setting the user underlying representation distribution to be the encoder-defined latent distribution, setting the transformation function f to be deep neural networks and choosing the multinomial distribution as the observation sampling distribution,

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{M!}{x_1!x_2!\dots x_I!} \exp\left\{\sum_{i=1}^{I} x_i \log \theta_i\right\}, \tag{8}$$

where the parameter $\boldsymbol{\theta}$ corresponds to the model output $\hat{\mathbf{x}}_u$. Its reconstruction loss function can be obtained by substituting $\mathbf{x}$ with $\mathbf{x}_u$ and $\boldsymbol{\theta}$ with $\hat{\mathbf{x}}_u$ in Equation 8 as

$$\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u) = -\sum_{i=1}^{I} x_{ui} \log f(\xi_u, \mathbf{0}). \tag{9}$$

An interesting point here is that MultVAE computes $\hat{\mathbf{x}}_u$ solely by taking ξ_u as input and without requiring the item representation $\boldsymbol{\lambda}_i$, and thus we use $\mathbf{0}$ as a dummy argument for f in Equation 9. The details of log-likelihood derivation for Equation 2, 5, 7, 9 can be found in Appendix A.

2.3 The analysis of latent factor CF models

Despite the existing latent factor collaborative filtering models seemingly disparate, the proposed probabilistic view enables us to draw their connections to deepen our understanding on them. We list several points here. 1) WMF differs from MF only by changing the variance of observation sampling distribution from fixed σ^2 to item-dependent σ_i^2 with $i \in [I]$. 2) In comparison to MF, NeuMF makes two changes, *i.e.,* replacing the inner product transformation function f with a MLP and substituting the Gaussian observation sampling distribution with Bernoulli. 3) The loss function adopted by BPF is a regularized version of the one used by MultVAE. This is revealed by Equation 7 and Equation 9 as well as the fact that $f(\xi_u, \boldsymbol{\lambda}_i)$ in Equation 7 is the parameter of Poisson distribution, which is nonnegative and can be considered as a regularization term.

Next, we analyze the advantages and limitations of the existing CF models. First, the existing latent factor CF models are all transductive over items in the sense that if we wish to recommend an item to users we must own its relevant observations in the training stage and consider it explicitly in the model design, which implies that the number of parameters of the models will grow with the number of items. Second, most of CF models (MF, WMF, NeuMF, BPF) are transductive over users whereas MultVAE is inductive. Being similar to item transductive, the parameters of models with user transductive are proportional to the number of users, however, for the MultVAE architecture as it adopts an encoder directly mapping users' activities into their representations, its model parameters are thus independent of the number of users and we are not required to observe a user's activities during the training phase. This is a very appealing property because it allows the models to scale to very large number of users in real-world applications. Lastly, the loss functions of the existing CF models are all induced by the empirical distribution ($\mathbf{x}_u$) and the model defined distribution ($\hat{\mathbf{x}}_u$) using the negative log-likelihood, $-\log p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u\right)$, we illustrate this in more details with the following proposition.

Proposition 2.1. *Minimizing the negative log-likelihood, that is, $-\log p(\mathbf{x}_u|\hat{\mathbf{x}}_u)$, leads to either minimizing the square of Euclidean distance between $\mathbf{x}_u$ and $\hat{\mathbf{x}}_u$ or minimizing the Kullback-Leibler divergence from the empirical distribution $\mathbf{x}_u$ (with appropriate normalization) to the model defined distribution $\hat{\mathbf{x}}_u$.*

More specifically, MF and WMF minimize the Euclidean distance while NeuMF, BPF and MultVAE attempt to minimize the KL-divergence. The argument can be found in Appendix A.

We now give an analysis of the problem of training the latent factor models using Euclidean distance and KL divergence from empirical distribution to model-defined observation sampling distribution, with an example.

$$\mathbf{X}^\top = \begin{bmatrix} 1 & \underline{0} & 0 & 0 \\ \underline{0} & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathrm{S} = \begin{bmatrix} 1.0 & 0.8 & 0.1 & 0.1 \\ 0.8 & 1.0 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1.0 & 0.8 \\ 0.1 & 0.1 & 0.8 & 1.0 \end{bmatrix}.$$

Assuming there are four users and four items, the observation matrix $\mathbf{X}$ and item similarity matrix S are given as above. S_{ij} indicates the similarity coefficient between item i and j, and the larger the value the more similar of two items. We make the following assumptions.

- i_1 and i_2 (i_3 and i_4) are similar items;
- i_2 and i_1 are false negative item for u_1 and u_2, respectively;
- whereas i_1, i_2 are true negative items for both u_3, u_4.

In particular, we assume that i_1 and i_2 are false negative items indicated in underline text. To see the problem of KL divergence, consider $\mathbf{x}_1 = [1,0,0,0]^\top$ and two model outputs $\hat{\mathbf{x}}_1^{(1)} = [1,1,0,0]^\top$, $\hat{\mathbf{x}}_1^{(2)} = [1,0,1,0]^\top$. For both outputs the KL divergence yields the same loss $\mathrm{KL}(\boldsymbol{\rho}_1||\hat{\mathbf{x}}_1^{(1)}) = \mathrm{KL}(\boldsymbol{\rho}_1||\hat{\mathbf{x}}_1^{(2)}) = \log 2$, where $\boldsymbol{\rho}_1$ is the normalized $\mathbf{x}_1$. Similarly, the Euclidean distance produces $\|\mathbf{x}_1 - \hat{\mathbf{x}}_1^{(1)}\|_2^2 = \|\mathbf{x}_1 - \hat{\mathbf{x}}_1^{(2)}\|_2^2 = 1$.

Clearly, in both cases we much prefer the model output $\hat{\mathbf{x}}_1^{(1)}$ as i_1 and i_2 are more similar. But neither KL divergence nor Euclidean distance is able to distinguish that. The essential problem lies in that both KL divergence and Euclidean distance are oblivious to the geometry induced by the item similarities when measuring the discrepancy between observation $\mathbf{x}_u$ and model output $\hat{\mathbf{x}}_u$, which might yield undesirable penalty towards different model outputs. Actually, this also explains why Hu *et al.* [14] advocates to put more weights on the positive items by adding weights w_{ui}, so that we do not penalize too much for the false negative items in the implicit feedback setting.

2.4 Sinkhorn collaborative filtering

To address this, it is highly desirable to introduce a loss function that explicitly takes the item similarities into consideration. The Earth Mover distance or more generally 1-Wasserstein distance $\mathcal{W}_1(\mu, \nu)$ measures the distance between two probability distributions μ and ν over a metric space $\mathcal{X}$ as follows:

$$\mathcal{W}_1(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} c\left(\mathbf{x}_1, \mathbf{x}_2\right) d\pi\left(\mathbf{x}_1, \mathbf{x}_2\right),$$

where $\Pi(\mu, \nu)$ denotes all joint distributions $\pi\left(\mathbf{x}_1, \mathbf{x}_2\right)$ whose marginals are μ and ν, respectively, and $c\left(\mathbf{x}_1, \mathbf{x}_2\right)$ represents the cost of moving a unit of mass from support point $\mathbf{x}_1$ to point $\mathbf{x}_2$ in the metric space $\mathcal{X}$. That is to say, $\mathcal{W}_1(\mu, \nu)$ explicitly considers the geometry induced by the metric in the space $\mathcal{X}$. In particular, when the distributions $\mu \in \mathbb{R}_+^m$ and $\nu \in \mathbb{R}_+^n$ are discrete, $\mathcal{W}_1(\mu, \nu)$ has the following form:

$$\mathcal{W}_1(\mu,\nu) = \min_{\mathbf{P}\geq 0} \langle \mathbf{P}, \mathbf{C}\rangle \quad \text{s.t.} \quad \mathbf{P}\mathbf{1} = \mu, \quad \mathbf{P}^\top \mathbf{1} = \nu. \tag{10}$$

in which $\mathbf{P} \in \mathbb{R}_+^{m\times n}$ is transport matrix, and $\mathbf{C} \in \mathbb{R}_+^{m\times n}$ is the cost matrix with $C_{ij} = c(\mathbf{x}_i, \mathbf{x}_j)$. In other words, the cost matrix $\mathbf{C}$ characterizes the geometry over the discrete metric space $\mathcal{X}$.

Wasserstein distance could naturally address the issues that KL divergence and Euclidean distance suffer if we could define a sensible transport cost matrix $\mathbf{C}$ such that C_{ij} is smaller if item i and j are more similar. Let us review the aforementioned example and define

$$\mathbf{C} = 1 - \mathbf{S} = \begin{bmatrix} 0.0 & 0.2 & 0.9 & 0.9 \\ 0.2 & 0.0 & 0.9 & 0.9 \\ 0.9 & 0.9 & 0.0 & 0.2 \\ 0.9 & 0.9 & 0.2 & 0.0 \end{bmatrix},$$

then under the cost matrix $\mathbf{C}$ we have

$$\mathcal{W}_1(\boldsymbol{\rho}_1, \hat{\mathbf{x}}_1^{(1)}) = 0.1, \mathcal{W}_1(\boldsymbol{\rho}_1, \hat{\mathbf{x}}_1^{(2)}) = 0.45.$$

by solving the linear programming problem in Equation 10. The results are consistent with our intuition that $\hat{\mathbf{x}}_1^{(1)}$ ought to produce a smaller distance than $\hat{\mathbf{x}}_1^{(2)}$.

However, to apply Wasserstein distance into training collaborative filtering models, two problems still remain: how to define the ground cost matrix and how to address the non-differentiable computation of Wasserstein distance. For the first problem, one possible solution is to learn the cost matrix $\mathbf{C}$ in an end-to-end fashion by employing min-max scheme [9, 31], however, this requires us to accurately estimate the value of $\mathcal{W}_1(\mu,\nu)$ which, as we will show later, is very difficult. Instead, in this paper, we consider a simple solution by defining $\mathbf{C} = 1 - \mathbf{S}$ where $\mathbf{S} \in \mathbb{R}^{I\times I}$ is the item-similarity matrix and can be estimated from user-item interaction matrix $\mathbf{X}$, *e.g.,* the normalized item-item cosine value. Another challenge is the computation of Wasserstein distance in Equation 10 involves solving a linear programming problem, which is not only very expensive but also is non-differentiable and cannot be used to propagate gradient backwards to train the model.

To sidestep this, one observation is that it is not necessary to compute the accurate value of $\mathcal{W}_1(\mu,\nu)$ as long as we could acquire a loss function that explicitly takes the item-similarity into account. Fortunately, if we regularize the distance in Equation 10 with an entropy term, we obtain the Sinkhorn loss which admits an iterative computation algorithm and is differentiable [6, 27], as follows.

$$\mathcal{W}_\epsilon(\mu,\nu) = \mathcal{W}_1(\mu,\nu) - \epsilon H(\mathbf{P}), \tag{11}$$

where $H(\mathbf{P})$ denotes the entropy of transport matrix $\mathbf{P}$. Sinkhorn algorithm starts with $b_0 = \mathbf{1}_m, \ell = 0$, and iterates as

$$\begin{aligned} \mathbf{a}_{\ell+1} &\overset{\text{def.}}{=} \frac{\mu}{\mathbf{K}\mathbf{b}_\ell} \\ \mathbf{b}_{\ell+1} &\overset{\text{def.}}{=} \frac{\nu}{\mathbf{K}^\top \mathbf{a}_{\ell+1}}, \end{aligned} \tag{12}$$

where $\mathbf{K} \in \mathbb{R}^{m\times n}$ is the kernel matrix with $K_{ij} \overset{\text{def.}}{=} \exp(-C_{ij}/\epsilon)$ and ϵ is a hyperparameter that controls the smooth of optimization objective of Equation 11. The iteration ends up with $\mathbf{P}_L = \text{diag}(\mathbf{a}_L)\mathbf{K}\text{diag}(\mathbf{b}_L)$ and Sinkhorn loss $\mathcal{W}_\epsilon^{(L)}(\mu,\nu) = \langle \mathbf{C}, \mathbf{P}_L\rangle$.

The Sinkhorn algorithm requires us to iterate L steps for Equation 12 and a larger L will lead to a more accurate loss value. However, each step involves a matrix multiplication with size $\mathbf{K} \in \mathbb{R}^{m\times n}$ or $\mathbb{R}^{\text{nz}(\mathbf{x}_u)\times \text{nz}(\hat{\mathbf{x}}_u)}$ in our case since only the nonzero entries in the probability measures contribute to the Sinkhorn loss [27]. This implies the computation cost and memory usage are both proportional to $O(L \cdot \text{nz}(\mathbf{x}_u) \cdot \text{nz}(\hat{\mathbf{x}}_u))$ for one user. In practice, $\text{nz}(\mathbf{x}_u)$ represents the number of items a user consumed which is usually small (less than 100 on average), however, the model output $\hat{\mathbf{x}}_u$ is a dense vector with size that equals to the number of items. The computation becomes infeasible when the number of items is large (*e.g.,* $I > 2000$), which makes it impractical to be adopted in real-world recommendation tasks. For example, the computation would be extremely slow and exhaust a 16GB GPU even when $L = 5$, $\text{nz}(\mathbf{x}_u) = 70$ and $I = 2000$.

To address this, we propose an accelerated version of Sinkhorn loss to scale the model to datasets with large number of items. The key observation is that when evaluating the Sinkhorn loss $\mathcal{W}_\epsilon^{(L)}(\mathbf{x}_u, \hat{\mathbf{x}}_u)$ with a sparse $\mathbf{x}_u$ and dense vector $\hat{\mathbf{x}}_u$, we can approximate it by selecting the top-k largest entries of $\hat{\mathbf{x}}_u$ in which only a matrix of size $\text{nz}(\mathbf{x}_u)\times k$ is required. The problem here is that only the selected k entries in $\hat{\mathbf{x}}_u$ could obtain their corresponding derivatives when back-propagating gradients from loss whereas the derivatives of all other $I - k$ entries are left to be zeros. In other words, we will end up with a partial gradient for $\hat{\mathbf{x}}$.

Note that as the sum of probability mass $\hat{\mathbf{x}}$ is a constant 1, the probability mass will flow from the entries with negative derivatives to the remaining entries (with non-negative derivatives) when updating $\hat{\mathbf{x}}$ using the gradient descent relevant methods. This implies that if we optimize $\hat{\mathbf{x}}_u$ using the partial gradient, the probability mass will flow from the k entries to the other $I - k$ positions (since they have non-negative derivatives) and consequently, the probability mass of the largest k-entries of $\hat{\mathbf{x}}_u$ will leak. Moreover, if we repeat updating $\hat{\mathbf{x}}$ by selecting its k-largest elements it will collapse into a uniform distribution. To remedy this, we propose to force the probability mass to concentrate around the most likely positions in $\hat{\mathbf{x}}_u$ by maximizing the sum of their corresponding probabilities. In the extreme case, these most likely positions correspond to the items consumed by user u; however, to avoid the over concentration we also allow a small fraction of probability mass to flow into the remaining positions, which is controlled by a hyperparameter $0 < \alpha < 1$. We equip this accelerated Sinkhorn loss with a MultVAE-style encoder-decoder architecture to obtain our final model—SinkhornCF—such that it is also inductive over users.

The details of accelerated Sinkhorn loss are described in Algorithm 1. In the first line of Algorithm 1, we partition the items into two sets, namely, the positive items I_1 and the remaining unobserved items I_2. In lines 3-8 we calculate the Sinkhorn loss between positive items $\mathbf{x}_u[I_1]$ and the top-k largest elements of model output $\hat{\mathbf{x}}_u[I_k]$, which is a lower bound approximation of $\mathcal{W}_\epsilon^{(L)}(\mathbf{x}_u, \hat{\mathbf{x}}_u)$. The Sinkhorn algorithm requires us to specify the number of iteration and convex smooth coefficient, and we find that $L = 5, \epsilon = 1.0$ gives rise to good performance in all our experiments. To mitigate the probability mass collapse due to the partial gradient evaluation, we add a concentration loss $\mathcal{L}_2$ in line 9. The first term of $\mathcal{L}_2$ encourages the probability mass to concentrate on the most

Algorithm 1: Accelerated Sinkhorn Loss

```
Input: $\mathbf{x}_u \in \mathbb{R}^I$, $\hat{\mathbf{x}}_u \in \mathbb{R}^I$, $\mathrm{C} \in \mathbb{R}_+^{I\times I}$, $\alpha, k, L = 5, \epsilon = 0.1$
Output: Loss $\mathcal{L}$
1  $I_1 \longleftarrow \{i \mid \mathbf{x}_{ui} > 0, i \in [I]\}$, $I_2 \longleftarrow [I] \setminus I_1$;
2  $I_k \longleftarrow$ The index of top-k elements of $\hat{\mathbf{x}}_u$;
3  $\mathbf{K} \longleftarrow \exp(-\mathrm{C}[I_1, I_k]/\epsilon)$, $\mathbf{b}_0 \longleftarrow \mathbf{1}_{|I_1|}$;
4  for $\ell \in 0, 1 \ldots L-1$ do
5      $\mathbf{a}_{\ell+1} \longleftarrow \frac{\mathbf{x}_u[I_1]}{\mathbf{K}\mathbf{b}_\ell}$;
6      $\mathbf{b}_{\ell+1} \longleftarrow \frac{\hat{\mathbf{x}}_u[I_k]}{\mathbf{K}^\top \mathbf{a}_{\ell+1}}$;
7  $\mathbf{P}_L \longleftarrow \mathrm{diag}(\mathbf{a}_L)\mathbf{K}\mathrm{diag}(\mathbf{b}_L)$;
8  $\mathcal{L}_1 \longleftarrow \langle \mathrm{C}, \mathbf{P}_L \rangle$;
9  $\mathcal{L}_2 \longleftarrow -\sum_{i\in I_1} \log \hat{\mathbf{x}}_{ui} - \sum_{i\in I_2} \alpha \log \hat{\mathbf{x}}_{ui}$;
10 $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$;
11 return $\mathcal{L}$;
```

probable positions whereas the second term also permits a small fraction of mass to flow into the remaining positions. For the purpose of numerical stability, we compute both $\mathcal{L}_1$ and $\mathcal{L}_2$ in the log-space. The storage of cost matrix $\mathrm{C} \in \mathbb{R}_+^{I\times I}$ would be expensive if I is very large; however, we can choose to store the similarity matrix S which is sparse and acquire the involved cost matrix as $\mathrm{C}[I_1, I_k] = 1 - \mathrm{S}[I_1, I_k]$. The computational cost of $\mathcal{L}_1$ and $\mathcal{L}_2$ is around $L * k$ and L, respectively, and we can alternately evaluate them with a higher frequency for $\mathcal{L}_2$ to accelerate the algorithm further in practice.

3 EXPERIMENTS

In this section, we aim to verify our analysis and evaluate the performance of our proposed model on two real-world recommendation datasets. First, we would like to verify our analysis that if the item-similarity-aware loss function—Sinkhorn loss—could improve the performance of the existing latent factor collaborative filtering models by simply replacing their original losses with Sinkhorn loss. Second, we aim to study whether our proposed model—SinkhornCF—is competitive with the state-of-art collaborative filtering models on the datasets with large number of items when Sinkhorn loss fails to fit in. Our source code is available at the public repository [1].

3.1 Datasets

- **ML-20M**. ML-20M is the MovieLens-20M dataset consisting of 136,677 users and 20,108 movies. Each record contains a user id, movie id as well as a score (from 1 to 5) indicating the preference of the user on the movie. The scores are binarized by the threshold score four, *i.e.,* the items with four or higher scores are interpreted as positive items. We only keep users who have watched at least five movies.
- **Netflix**. Netflix is movie rating dataset released by the Netflix prize competition. Similar to ML-20M, we process it by keeping the movies with four or higher scores as positive items and users having watched at least five movies.
- **ML-20M-S**. To verify our analysis that whether Sinkhorn loss could improve the performance of the existing latent factor collaborative filtering models when it could fit, we randomly select a subset of records from ML-20M, which results in 6,847 users and 845 items. We refer to this small dataset as ML-20-S.

[1]https://github.com/boathit/sinkhorncf

Table 2: Dataset statistics.

	ML-20M	**Netflix**	**ML-20M-S**
#users	136,677	463.435	6,847
#items	20,108	17,769	845
#iteractions	10.0M	56.9M	0.76M
#validation users	10,000	40,000	500
#test users	10,000	40,000	500

The basic statistics of datasets are summarized in Table 2.

3.2 Experimental setup

3.2.1 Evaluation metrics. We evaluate the performance of different methods using two ranking metrics, namely, Recall@N and nDCG@N (normalized discounted cumulative gain) under strong generalization [21]. To this end, we split the users into training, validation and test sets. The entire interaction historical records of training users are used to train the model. For a user on validation and test sets, we take a fraction of his/her clicking history (80%) to obtain his/her representation ξ_u and use it to calculate the model output $\hat{\mathbf{x}}_u$, then we compute the ranking metrics using the remaining clicking history of the user and $\hat{\mathbf{x}}_u$.

Recall@N compares the predicted rank of the items with their ground truth rank within the first N predictions, and it considers these ranks to be equally important. On the other hand, nDCG@N adopts a monotonically increasing weight to favor the higher ranks. More formally, for a given user, they are defined as follows.

$$\text{Recall@N} \overset{\text{def.}}{=} \sum_{n=1}^{N} \frac{\mathbb{1}(r(n) \in I_1)}{\min(N, |I_1|)},$$

$$\text{DCG@N} \overset{\text{def.}}{=} \sum_{n=1}^{N} \frac{2^{\mathbb{1}(r(n)\in I_1)} - 1}{\log(r+1)},$$

where $r(n)$ indicates the item at rank n and I_1 denotes the positive items set for the user. nDCG@N is obtained by normalizing DCG@N using the best possible DCG@N value, in which all positive items are ranked on the top.

3.2.2 Parameter settings. As aforementioned, we aim to conduct two types of experiments in this section. First, we replace the original loss functions of several existing latent factor models with Sinkhorn loss while keeping all the remaining parts to be identical, so as to check their performance changes on the small dataset ML-20M-S. More specifically, we choose three representative models, MF, NeuMF and MultVAE. For MF and NeuMF, we set the user and item representation dimension to be 16 and adopt a one hidden-layer MLP with hidden layer size 50 for NeuMF. The architecture $[I \to 600 \to 200 \to 600 \to I]$ is used for MultVAE. We set $L = 5, \epsilon = 1.0$ for the Sinkhorn loss and all models are optimized by Adam [15] with a batch size 200.

Second, we intend to study the performance of our proposed SinkhornCF on the two entire datasets ML-20M and Netflix and test whether it is competitive with the state-of-art collaborative filtering models. To this end, we compare SinkhornCF with the following latent factor collaborative filtering models WMF, SLIM, NeuMF and MultVAE. WMF and SLIM are two classical linear latent factor models whereas NeuMF and MultVAE are deep learning-based nonlinear models. In addition, a recent evaluation paper [7] points out that the traditional simple models usually could achieve much better results than the deep learning-based methods when tuned carefully. For this reason, we also include ItemKNNCF, P3alpha and RP3beta in our experiments. We tune the hyperparameters of all baseline methods on the validation data sets and the details are explained in Section 3.2.3. For the parameters of SinkhornCF, we adopt the auto-encoder architecture $[I \longrightarrow 600 \longrightarrow 200 \longrightarrow 600 \longrightarrow I]$ and $L = 5, \epsilon = 1.0, k = 200$ for both two datasets. The α is tuned using binary search on the validation datasets and set to 0.03 and 0.01 for ML-20M and Netflix dataset, respectively. It is optimized by Adam [15] with a batch size 500 for 100 epochs and the best model is selected using nDCG@100 on validation datasets.

In all experiments, the cost matrix for the Sinkhorn and accelerated Sinkhorn loss is obtained by $\mathbf{C} = 1 - \mathbf{S}$ and the entry S_{ij} of $\mathbf{S}$ is estimated by using the cosine value of the angle between vector $\mathbf{x}_i$ and $\mathbf{x}_j$ on the training datasets, *i.e.*, $S_{ij} = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}$.

3.2.3 *Baseline methods.*

- **ItemKNNCF**. ItemKNNCF [19, 32] is a classic item-based nearest-neighbor method using the Cosine Similarity. Following [7], we choose the neighborhood size from [5, 1000] and the shrinkage term from [0, 1000].
- **P3alpha**. P3alpha [5] is a graph-based method which performs a random walk between users and items based on the observed interaction matrix. We select the number of neighbors from [5, 1000] and set the damping factor, α, to a real value between 0 to 2.
- **RP3beta**. RP3beta [26] is an improved version of P3alpha [5], whereby the similarity between two items is further divided by each item's popularity raised to the the power of a coefficient β. Similarly, we select the number of neighbors from [5, 1000] and set both the damping factor, α, and coefficient, β, to a real value between 0 to 2.
- **SLIM**. SLIM [25] is a linear latent factor model which learns an asymmetric item-similarity matrix by imposing sparse constraint. It is learned by solving a constrained linear optimization problem with a mixed ℓ_1 and ℓ_2 regularization. Following [7], we use the more scalable variant from [17]. The ratio of ℓ_1 and ℓ_2 regularization is selected from $[10^{-3}, 10^0]$ and the regularization magnitude coefficient is set to a value in $[10^{-3}, 10^0]$.
- **WMF**. As we showed in Section 2.2, Weighted Matrix Factorization (**WMF**) [14] can be considered as a generalized matrix factorization by adopting a Gaussian observation sampling distribution with item-dependent diagonal variance matrix. The weight for unobserved entries is set to 1, while the weight for observed entries is selected from $\{2, 5, 10, 30, 50, 100\}$. The number of latent dimensions is chosen from $\{25, 50, 75, 100, 150, 200\}$, and the model is trained using Alternating Least Squares (**ALS**).
- **NeuMF**. NeuMF [8, 13] is also a generalized matrix factorization by exploring the nonlinear interaction between user and item representations (Section 2.2). We use the public available source code released by the authors. As suggested by [13], we adopt a 3-layer MLP with a tower structure, whereby the number of predictive factors for both GMF and MLP is selected from $\{8, 16, 32, 64\}$. NeuMF is optimized using Adam [15] with a default learning rate of 0.001, and we use between 2 to 8 negative samples for each observed entry.
- **MultVAE**. MultVAE [18] is a deep generative model (Section 2.2) which has been shown to be a strong deep-learning-based model over a variety of baseline methods [7]. We use the public available code provided by the authors and tune the necessary hyperparameters. Specifically, the autoencoder-based architecture for MultVAE with 1 hidden layer is $[I \longrightarrow 600 \longrightarrow 200 \longrightarrow 600 \longrightarrow I]$, whereby I is the number of items and the dimension of any hidden layer is set to 600. For MultVAE, we tried between 0 to 2 hidden layers, while the β hyperparameter (for KL annealing) is selected from $\{0.1, 0.2, 0.3, 0.5, 1.0\}$. MultVAE is optimized by Adam with a learning rate 0.001 for 200 epochs.

Table 3: nDCG changes of MF on *ML-20M-S*.

Methods	nDCG@20	nDCG@50	nDCG@100
Original	0.258	0.297	0.315
Replaced	**0.283**	**0.326**	**0.352**
Improvement (%)	9.690	9.764	11.746

Table 4: nDCG changes of NeuMF on *ML-20M-S*.

Methods	nDCG@20	nDCG@50	nDCG@100
Original	0.255	0.279	0.312
Replaced	**0.324**	**0.363**	**0.390**
Improvement (%)	27.059	30.108	25.000

Table 5: nDCG changes of MultVAE on *ML-20M-S*.

Methods	nDCG@20	nDCG@50	nDCG@100
Original	0.400	0.432	0.461
Replaced	**0.415**	**0.457**	**0.476**
Improvement (%)	3.750	5.787	3.254

3.3 The effectiveness of Sinkhorn loss

In this section, we aim to verify our hypothesis that an item-similarity-aware loss function could benefit the existing latent factor models by designing the following experiments. We choose

Table 6: Recall & nDCG for *MovieLens-20M*. Best result is made *bold*, and the runner-up is underlined. Statistically significant improvements with *p-value < 0.01* using the paired sample t-test are indicated with **.

Model	Recall				nDCG			
	@ 20	@ 50	@ 75	@ 100	@ 20	@ 50	@ 75	@ 100
ItemKNNCF	0.311	0.435	0.504	0.555	0.266	0.306	0.329	0.346
P3alpha	0.291	0.397	0.446	0.475	0.244	0.280	0.298	0.308
RP3beta	0.340	0.463	0.528	0.571	0.290	0.331	0.353	0.368
SLIM	0.375	0.498	0.566	0.615	0.328	0.367	0.391	0.407
WMF	0.361	0.493	0.561	0.611	0.307	0.351	0.376	0.392
NeuMF	0.161	0.321	0.419	0.494	0.110	0.168	0.195	0.221
MultVAE	0.400	0.538	0.610	0.661	0.339	0.387	0.412	0.429
SinkhornCF	**0.405**	**0.542**	**0.613**	**0.665**	**0.349**	**0.395**	**0.420**	**0.438**
Improvement (%)	1.223 **	0.723 **	0.493 **	0.490 **	2.987 **	2.180 **	1.932 **	1.872 **

Table 7: Recall & nDCG for *Netflix*. Best result is made *bold*, and the runner-up is underlined. Statistically significant improvements with *p-value < 0.01* using the paired sample t-test are indicated with **.

Model	Recall				nDCG			
	@ 20	@ 50	@ 75	@ 100	@ 20	@ 50	@ 75	@ 100
ItemKNNCF	0.271	0.348	0.406	0.454	0.251	0.269	0.288	0.305
P3alpha	0.259	0.298	0.321	0.337	0.242	0.250	0.259	0.266
RP3beta	0.307	0.378	0.426	0.463	0.288	0.302	0.319	0.334
SLIM	0.347	0.426	0.486	0.534	0.325	0.341	0.361	0.379
WMF	0.314	0.393	0.450	0.496	0.293	0.310	0.330	0.347
NeuMF	0.107	0.213	0.299	0.373	0.089	0.130	0.162	0.190
MultVAE	0.350	0.441	0.504	0.555	0.321	0.344	0.366	0.385
SinkhornCF	**0.356**	**0.445**	**0.507**	**0.558**	**0.327**	**0.349**	**0.371**	**0.389**
Improvement (%)	1.641 **	0.775 **	0.609 **	0.418 **	0.513 **	1.470 **	1.279 **	1.154 **

three representative latent factor models—MF, NeuMF and Mult-VAE—and replace their original loss functions with the Sinkhorn loss while keeping all the remaining components and training details to be identical, using the small dataset ML-20M-S on which the Sinkhorn loss could fit. Then we study the performance changes of them in terms of nDCG.

Table 3-5 show the nDCG changes of different methods when replacing their original loss functions with Sinkhorn loss. First, all methods have quite significant improvements when equipped with Sinkhorn loss over their original loss functions. In particular, the improvements of MF and NeuMF are significant, which are around 10% and 27%, respectively. Another point we would like to emphasize is that MF gives rise to a better performance than NeuMF even when they use the same latent representation dimension; however, when we substitute their original losses with Sinkhorn loss, NeuMF could outperform MF by very large margins, around 10%. This may suggest that the Sinkhorn loss is more suitable for models with high learning capabilities.

3.4 The performance of SinkhornCF

The Sinkhorn loss shows its effectiveness in enhancing the performance of the existing latent factor models. However, it becomes extremely slow and GPU-expensive when the number of items grow. For this reason, we would like to test the performance of our proposed SinkhornCF which adopts the accelerated Sinkhorn loss described in Algorithm 1 on the two complete datasets, ML-20M and Netflix.

Table 6 and Table 7 present the performance of different models in terms of nDCG and Recall on the two datasets, respectively. The best results are highlighted in bold and the runner-ups are indicated in underline, the p-values to test the statistical significance of improvements are also calculated and the improvements with p-value smaller than 0.01 are labeled with double stars. First, the proposed SinkhornCF is able to achieve the best results across all metrics on both two datasets. Its improvements over the best baseline models are reported on the bottom of the tables, and the improvements are more evident in terms of nDCGs. On average, the improvements are around 2% to 3% in terms of nDCG on ML-20M dataset while they are roughly 1% on Netflix dataset. Among all baseline methods MultVAE and SLIM competes the best positions, and MultVAE can acquire better performance than SLIM over most of metrics except nDCG@20 on Netflix dataset, this is consistent with the results found in [7] that SLIM is more competitive in terms of small cutoff lengths. It is worthwhile pointing out that the deep learning-based model NeuMF is not as competitive as all other

Table 8: nDCG@20 varies against various parameters on *ML-20M* dataset.

k	50	100	200	300	500
nDCG@20	0.347	0.348	0.349	0.349	0.349
L	1	2	3	5	10
nDCG@20	0.346	0.342	0.345	0.349	0.339
ϵ	0.01	0.1	1.0	2.0	5.0
nDCG@20	0.342	0.343	0.349	0.347	0.345
α	0.01	0.02	0.03	0.05	0.1
nDCG@20	0.347	0.348	0.349	0.345	0.342

Table 9: Running time (minutes) on *ML-20M* and *Netflix*.

Datasets	NeuMF	MultiVAE	SinkhornCF
ML-20M	53	16	41
Netflix	1,740	223	450

baseline methods on the two datasets even though we have tuned its parameters over a wide range of suggested values.

In the end, we should admit that although our proposed SinkhornCF shows competitive results over a wide range of collaborative filtering models on the two studied datasets, the improvements are not as large as those achieved by the Sinkhorn loss on the smaller dataset. We suspect this is due to the gradient estimation of our accelerated Sinkhorn loss is not as accurate as that of the Sinkhorn loss. But this also implies it is a good direction to adopt item-similarity-aware loss function for the latent factor models, and we leave it as our future work to explore more effective loss functions that could consider the geometry induced by the item similarity.

3.5 Parameter-sensitivity analysis

In this experiment, we study the impact of parameters in Algorithm 1 on the performance of SinkhornCF. Table 8 reports the nDCG@20 of SinkhornCF changes against k, L, ϵ and α, respectively, on ML-20M dataset. A larger k implies a better approximation of the Sinkhorn loss and raising k from 50 to 200 indeed could yield better nDCG; however, no further performance gain is observed when continuing increasing it. We suspect this is caused by the probability mass leakage and seek to improve it in the future. Next, technically, a larger L and a smaller ϵ could produce a more accurate loss value but it also brings challenges to the model optimization. As the table shows, the performance drops a lot when increasing L from 5 to 10. We observe that $L = 5$ and $\epsilon = 1.0$ could often yield very good results in our experiments. Lastly, α that controls the concentration degree also has a notable impact on the performance. A smaller α means a higher probability mass concentration and as table shown, nDCG@20 achieves its best value at $\alpha = 0.03$ and it drops as α continues growing. We empirically find that the best α could be selected using binary search from range $(0, 0.1)$ on validation datasets.

3.6 Empirical running time

The empirical running time of different methods (we only include methods running on GPU) on *ML-20M* and *Netflix* are presented in Table 9. The running time of SinkhornCF is roughly twice of that of MultiVAE but is less than that of NeuMF.

4 RELATED WORK

Latent factor collaborative filtering. Latent factor collaborative filtering models attempt to make recommendations by learning each user a latent representation of preferences and each item a latent representation of attributes. They have been the dominating models in the recommendation community [10, 14, 30, 33] during the past decades. MF, as the most basic latent factor model, has yielded various types of advanced methods. WMF [14] generalizes MF by introducing weighted penalty for unobserved items. PMF [30] approaches user and item representation learning from a hierarchical probabilistic perspective. BPF [10, 11] proposes to explain the generation of user interactions with a Poisson distribution. Very recently, inspired by the success of deep learning techniques in computer vision and natural language processing, many deep neural nets-based latent factor models were developed in the recommendation community. CDAE [35] learns the latent user representations using a denoising auto-encoder. MultVAE [18], powered by Variational Auto-Encoders [16, 28], maps a user historical records into a latent representation distribution directly with a neural net and reconstructs the observation using Multinomial distribution. ANR [4] attempts to model the different aspects of users and items with attention mechanism. HyperML [34] characterizes the embedding proximity relationship with the hyperbolic distance to generalize the latent space from Euclidean space to Hyperbolic space.

Sinkhorn divergence. Sinkhorn divergence arises from the research of Optimal Transport [6, 27]. Optimal Transport (OT) has been a long standing research topic in machine learning [2, 20, 23, 24, 29] and gained its popularity recently due to its ability in successfully training implicit generative models [1, 9, 12, 31]. The success lies in the fact that OT is able to measure the discrepancy between two probability distributions with non-overlapping supports, which is achieved by explicitly considering the geometry of the metric space on which the probability distributions are defined [27]. Our usage of OT differs from that in implicit generative model training in two aspects. First, OT is used to minimize the discrepancy between only two distributions, *i.e.*,, data distribution $\mathbb{P}_r$ and model distribution $\mathbb{P}_m$; however, in our case, we need to minimize U-pairs of distributions $(\mathbf{x}_u, \hat{\mathbf{x}}_u)$, $u \in [U]$. Second, the dimensions of the metric spaces for implicit generative models are medium (around 200) whereas it equals to the number of items in our case, which implies the direct application of Sinkhorn loss will fail. We also note that OT is applied into cold-start recommendation problem in [22], where it is used to measure the similarity between two users from their historical records, *i.e.*, $(\mathbf{x}_{u_1}, \mathbf{x}_{u_2})$. Since $\mathbf{x}_u$ is sparse, a straightforward application of Sinkhorn loss works very well. This is quite different from our case where Sinkhorn loss is used to train the model whose output $\hat{\mathbf{x}}_u$ is a high-dimensional dense vector.

5 CONCLUSION

In this paper we provide a unified view to study various types of latent factor collaborative filtering models proposed in recommendation community from a probabilistic perspective. This is accomplished by interpreting different models using one probabilistic generative process, and specifying them with their unique underlying representation distributions, transformation function and observation sampling distribution. We examine a collection of representative latent factor models under this unified framework, which not only helps us discern their underlying connections and deepen our understanding on them but also enables us to investigate their advantages and limitations. In particular, we note that the loss functions adopted either attempt to minimize the Euclidean distance or KL-divergence, which are oblivious to the geometry induced by the item similarities and might lead to undesirable trained models. Motivated by this, we propose to adopt Sinkhorn loss function that explicitly considers item similarities in its computation. Despite its success in training implicit generative models, we note that Sinkhorn loss struggles in handling probability measures defined in high dimensional spaces whose dimension equals to the number of items in our cases. To sidestep this, we develop an accelerated Sinkhorn loss and equip it with the user-inductive auto-encoder architecture to obtain our proposed SinkhornCF, which is not only item-similarity aware but also scales to large datasets. We conduct experiments on two real-world recommendation datasets. We first verify the effectiveness of Sinkhorn loss on a small dataset by replacing with it the original loss functions of several representative latent factor models. Then we compare our proposed SinkhornCF with a collection of strong baseline methods on the two entire datasets. The experimental results show that the proposed model is competitive with the state-of-art collaborative filtering models, which inspires us to explore other more effective item-similarity-aware loss functions beyond Sinkhorn in the future.

We would like to highlight that our work proves it is a good start of designing item-similarity-aware loss functions for latent factor collaborative filtering models. In the future, we would like to explore the following problems: 1) how to enhance the cost matrix design when side information is available; 2) how to generalize the work to the explicit feedback settings; 3) how to design more effective and efficient item-similarity-aware loss functions.

ACKNOWLEDGEMENTS

This work was supported in part by a MOE Tier-2 grant MOE2019-T2-2-181, a MOE Tier-1 grant RG114/19, and an NTU ACE grant. The authors would like to thank the anonymous reviewers who give the thoughtful comments and helpful suggestions.

REFERENCES

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* (2017).

[2] Federico Bassetti, Antonella Bodini, and Eugenio Regazzini. 2006. On minimum Kantorovich distance estimators. *Statistics & probability letters* 76, 12 (2006), 1298–1302.

[3] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* 60, 2 (2018), 223–311.

[4] Jin Yao Chin, Kaiqi Zhao, Shafiq R. Joty, and Gao Cong. 2018. ANR: Aspect-based Neural Recommender. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 147–156.

[5] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random Walks in Recommender Systems: Exact Computation and Simulations. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea) *(WWW '14 Companion)*. Association for Computing Machinery, New York, NY, USA, 811–816. https://doi.org/10.1145/2567948.2579244

[6] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2292–2300.

[7] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 101–109. https://doi.org/10.1145/3298689.3347058

[8] Gintare Karolina Dziugaite and Daniel M. Roy. 2015. Neural Network Matrix Factorization. *CoRR* abs/1511.06443 (2015).

[9] Aude Genevay, Gabriel Peyré, and Marco Cuturi. 2018. Learning Generative Models with Sinkhorn Divergences. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain (Proceedings of Machine Learning Research, Vol. 84)*, Amos J. Storkey and Fernando Pérez-Cruz (Eds.). PMLR, 1608–1617.

[10] Prem Gopalan, Jake M. Hofman, and David M. Blei. 2015. Scalable Recommendation with Hierarchical Poisson Factorization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, Marina Meila and Tom Heskes (Eds.). AUAI Press, 326–335. http://auai.org/uai2015/proceedings/papers/208.pdf

[11] Prem K Gopalan, Laurent Charlin, and David Blei. 2014. Content-based recommendations with poisson factorization. In *NIPS 2014*. 3176–3184.

[12] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 5767–5777.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 173–182.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets *(ICDM '08)*. IEEE Computer Society, USA, 263–272. https://doi.org/10.1109/ICDM.2008.22

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

[16] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[17] Mark Levy and Kris Jack. 2013. Efficient Top-N Recommendation by Linear Regression. (2013).

[18] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[19] G. Linden, B. Smith, and J. York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[20] James Robert Lloyd and Zoubin Ghahramani. 2015. Statistical Model Criticism using Kernel Two Sample Tests. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 829–837.

[21] Benjamin Marlin. 2004. *Collaborative filtering: A machine learning perspective*. University of Toronto Toronto.

[22] Yitong Meng, Xiao Yan, Weiwen Liu, Huanhuan Wu, and James Cheng. 2020. Wasserstein Collaborative Filtering for Item Cold-start Recommendation. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 318–322.

[23] Grégoire Montavon, Klaus-Robert Müller, and Marco Cuturi. 2016. Wasserstein Training of Restricted Boltzmann Machines. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 3711–3719.

[24] Jonas Mueller and Tommi S. Jaakkola. 2015. Principal Differences Analysis: Interpretable Characterization of Differences between Distributions. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 1702–1710.

[25] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proceedings of the 2011 IEEE 11th International*

Conference on Data Mining (ICDM '11). IEEE Computer Society, USA, 497–506. https://doi.org/10.1109/ICDM.2011.134

[26] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2016. Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications. 7, 1, Article 1 (Dec. 2016), 34 pages. https://doi.org/10.1145/2955101

[27] Gabriel Peyré and Marco Cuturi. 2019. Computational Optimal Transport. *Found. Trends Mach. Learn.* 11, 5-6 (2019), 355–607.

[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML 2014, Beijing, China, 21-26 June 2014.* 1278–1286.

[29] Antoine Rolet, Marco Cuturi, and Gabriel Peyré. 2016. Fast Dictionary Learning with a Smoothed Wasserstein Loss. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016 (JMLR Workshop and Conference Proceedings, Vol. 51)*, Arthur Gretton and Christian C. Robert (Eds.). JMLR.org, 630–638.

[30] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (Eds.). 1257–1264.

[31] Tim Salimans, Han Zhang, Alec Radford, and Dimitris N. Metaxas. 2018. Improving GANs Using Optimal Transport. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. 285–295.

[33] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008*, Pearl Pu, Derek G. Bridge, Bamshad Mobasher, and Francesco Ricci (Eds.). ACM, 267–274. https://doi.org/10.1145/1454008.1454049

[34] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. HyperML: A Boosting Metric Learning Approach in Hyperbolic Space for Recommender Systems. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020.* ACM, 609–617. https://doi.org/10.1145/3336191.3371850

[35] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, Paul N. Bennett, Vanja Josifovski, Jennifer Neville, and Filip Radlinski (Eds.). ACM, 153–162.

[36] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.

A APPENDIX

Classical Matrix Factorization Model.

$$\begin{aligned}\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u) &= -\log p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u, \sigma^2\right)\\ &= \sum_{i=1}^{I}(x_{ui}-\hat{x}_{ui})^2 = \sum_{i=1}^{I}\left(x_{ui} - f(\xi_u, \boldsymbol{\lambda}_i)\right)^2.\end{aligned} \tag{13}$$

Neural Matrix Factorization.

$$\begin{aligned}\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u) &= -\log p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u\right)\\ &= -\sum_{i=1}^{I} x_{ui}\log \hat{x}_{ui} - \sum_{i=1}^{I}(1-x_{ui})\log(1-\hat{x}_{ui})\\ &= -\sum_{i=1}^{I} x_{ui}\log f(\boldsymbol{\xi_u}, \boldsymbol{\lambda}_i) - \sum_{i=1}^{I}(1-x_{ui})\log(1-f(\boldsymbol{\xi_u}, \boldsymbol{\lambda}_i)).\end{aligned} \tag{14}$$

Bayesian Poisson Factorization.

$$\begin{aligned}\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u) &= -\log p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u\right)\\ &= -\sum_{i=1}^{I} x_{ui}\log \hat{x}_{ui} + \sum_{i=1}^{I}\hat{x}_{ui}\\ &= -\sum_{i=1}^{I} x_{ui}\log f(\xi_u, \boldsymbol{\lambda}_i) + \sum_{i=1}^{I} f(\xi_u, \boldsymbol{\lambda}_i).\end{aligned} \tag{15}$$

Multinomial Variational Auto-encoders.

$$\begin{aligned}\ell(\mathbf{x}_u, \hat{\mathbf{x}}_u) &= -\log p\left(\mathbf{x}_u|\hat{\mathbf{x}}_u\right)\\ &= -\sum_{i=1}^{I} x_{ui}\log \hat{x}_{ui}\\ &= -\sum_{i=1}^{I} x_{ui}\log f(\xi_u, \mathbf{0}).\end{aligned} \tag{16}$$

Proof of Proposition 2.1.

Proof. First, it is easy to see that MF (WMF) minimize the Euclidean distance between $\mathbf{x}_u$ and $\hat{\mathbf{x}}_u$ from their definition. For NeuMF, as it adopts the Bernoulli distribution for $p(\mathbf{x}_u|\hat{\mathbf{x}}_u)$ which is defined over each dimension i, hence

$$\begin{aligned}\mathrm{KL}(\mathbf{x}_u||\hat{\mathbf{x}}_u) &= \sum_{i=1}^{I}\mathrm{KL}(x_{ui}||\hat{x}_{ui})\\ &= \sum_{i=1}^{I} x_{ui}\log\frac{x_{ui}}{f(\xi_u, \boldsymbol{\lambda}_i)} + (1-x_{ui})\log\frac{1-x_{ui}}{1-f(\xi_u, \boldsymbol{\lambda}_i)}\\ &= Z - \sum_{i=1}^{I} x_{ui}\log f(\xi_u, \boldsymbol{\lambda}_i) + (1-x_{ui})\log(1-f(\xi_u, \boldsymbol{\lambda}_i))\\ &\propto -\log p(\mathbf{x}_u|\hat{\mathbf{x}}_u),\end{aligned} \tag{17}$$

where $Z = \sum_{i=1}^{I} x_{ui}\log x_{ui} + (1-x_{ui})\log(1-x_{ui})$ is a constant and $-\log p(\mathbf{x}_u|\hat{\mathbf{x}}_u)$ corresponds to the one in Equation 14.

To see the last two, let $\boldsymbol{\rho}_u = \mathbf{x}_u/\sum_{i=1}^{I} x_{ui}$, *i.e.*, we normalize $\mathbf{x}_u$ to be an appropriate distribution, then

$$\begin{aligned}\mathrm{KL}(\mathbf{x}_u||\hat{\mathbf{x}}_u) &= \sum_{i=1}^{I}\rho_{ui}\log\frac{\rho_{ui}}{f(\xi_u, \boldsymbol{\lambda}_i)}\\ &= -\sum_{i=1}^{I}\rho_{ui}\log f(\xi_u, \boldsymbol{\lambda}_i) + \sum_{i=1}^{I}\rho_{ui}\log\rho_{ui}\\ &= -\sum_{i=1}^{I}\frac{x_{ui}}{\sum_{i=1}^{I} x_{ui}}\log f(\xi_u, \boldsymbol{\lambda}_i) + \sum_{i=1}^{I}\rho_{ui}\log\rho_{ui}\\ &= Z_2 - Z_1\sum_{i=1}^{I} x_{ui}\log f(\xi_u, \boldsymbol{\lambda}_i)\\ &\propto -\log p(\mathbf{x}_u|\hat{\mathbf{x}}_u),\end{aligned} \tag{18}$$

where $Z_1 = 1/\sum_{i=1}^{I} x_{ui} > 0$ and $Z_2 = \sum_{i=1}^{I}\rho_{ui}\log\rho_{ui}$ are constants and $-\log p(\mathbf{x}_u|\hat{\mathbf{x}}_u)$ corresponds to the ones in Equation 15 (up to a regularization term) and Equation 16. □