Dynamic Collaborative Recurrent Learning

Teng Xiao
School of Data and Computer Science, Sun Yat-sen University, China
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
tengxiao01@gmail.com

Shangsong Liang*
School of Data and Computer Science, Sun Yat-sen University, China
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
liangshangsong@gmail.com

Zaiqiao Meng
School of Computing Science, University of Glasgow, Scotland, UK
zaiqiao.meng@gmail.com

ABSTRACT

In this paper, we provide a unified learning algorithm, dynamic collaborative recurrent learning, DCRL, of two directions of recommendations: temporal recommendations focusing on tracking the evolution of users' long-term preference and sequential recommendations focusing on capturing short-term preferences given a short time window. Our DCRL builds based on RNN and Sate Space Model (SSM), and thus it is not only able to collaboratively capture users' short-term and long-term preferences as in sequential recommendations, but also can dynamically track the evolution of users' long-term preferences as in temporal recommendations in a unified framework. In addition, we introduce two smoothing and filtering scalable inference algorithms for DCRL's offline and online learning, respectively, based on amortized variational inference, allowing us to effectively train the model jointly over all time. Experiments demonstrate DCRL outperforms the temporal and sequential recommender models, and does capture users' short-term preferences and track the evolution of long-term preferences.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Sequential recommendation; Temporal recommendation; Deep generative model

ACM Reference Format:
Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Dynamic Collaborative Recurrent Learning. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357901

1 INTRODUCTION

Recommender systems aim at providing items from rich candidates with users' historical data called user-item feedback matrix. Most matrix completion based recommendation methods [13, 20, 32, 44] are proposed to predict the missing values for user-item feedback matrix. However, in many streaming platforms, e.g., Amazon and eBay, such users' feedbacks are sequential, personalized, time-varying, and more complex than those represented by a matrix. Recommendation methods under the streaming scenario should address the following issues simultaneously:
(1) *Capture users' short-term preferences.* One nature of users' feedbacks is sequential, which represents users' short-term preferences. E.g., users prefer to purchase camera accessories than a shoes after purchasing a camera in a short time [26, 28, 39].
(2) *Capture users' long-term preferences* : It is important to consider users' long preferences which are dependent on users' personalization in recommendation [16]. E.g., some users always prefer listening folk musics but not blue musics.
(3) *Track the evolution of users' long-term preferences (concept drift).* The evolution of users' long-term preferences is also important for online recommendation [19, 42]. E.g., a long-time folk music fan might listen blue musics that she finds inspiring, and then switch to listen more blue musics.

In recent years, many recommendation methods under the streaming scenario have been proposed to address the aforementioned three issues, which can be categorized into either ***temporal*** or ***sequential*** ones. Temporal recommendation methods make use of the explicit timestamps to track the evolution of users' long-term preferences over time. For instance, the temporal methods recommend T-shirt in the summer, instead of winter; and recommend coffee in the morning, instead of evening [39]. They utilize traditional probabilistic models such as Gaussian state space models [3, 4, 12], Gamma-Markov chains [7, 17] or temporal factor models [19, 48] to dynamically track users' long-term preferences over time. In contrast, sequential recommendation methods mainly focus on capturing users' short-term preferences and learning the relevance between items in a short time window [33]. For instance, sequential methods would recommend phone accessories just after a user bought an iPhone, independently of the time [39]. Representative sequential methods include those based on Markov chain [5] or RNN [14, 22, 28] etc.

While the temporal recommendation methods can capture users' long-term preferences (issue (2)) and track their evolution (issue (3)), they still apply matrix completion methods at each time window, which limits them to model the sequential pattern of users and capture short-term preferences (issue (1)). While the sequential recommendation methods do can capture users' short preferences (issue (1)), they neglect users' long-term preferences (issue (2))

*Shangsong Liang is the corresponding author of the paper.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357901

and more importantly, they can not capture their evolutions (issue (3)). More recently, some improved personalized sequential methods [34, 39, 40, 43] have been proposed to incorporate users' long-term preferences (issue (2)) into their models, and have shown promising performance in top-N sequential recommendation task. However they only consider users' long-term preferences as static preferences and can not explicitly model their evolutions over time (issue (3)), which is unrealistic in many real-world scenarios. Obviously, none of the previous work can be able to address all the aforementioned issues.

Accordingly, in this paper, we aim at unifying the two directions of recommendations: temporal and sequential recommendations to address all the three issues simultaneously. Inspired by RNN and State Space Model (SSM) [41] that have been investigated separately in dynamic representation learning, we propose a novel deep generative model, **D**ynamic **C**ollaborative **R**ecurrent **L**earning, DCRL, to unify the aforementioned two directions of recommendations. Specifically, we first introduce a new user latent vector into RNN to collaboratively model the relations between users' short-term and long-term preferences, We then propose a nonlinear latent SSM and utilize a deep Kalman filter [21] as a prior for the nonlinear time evolution of the long-term preferences, allowing it to share information across all time periods and to be robust against data sparsity. In our DCRL, the goals of sequential and temporal recommendations can be achieved by the proposed variant RNN and the proposed nonlinear latent SSM, respectively, which are unified by a proposed learning framework. For the efficient inference purpose, instead of introducing variational parameters for every variational distributions like that in black-box variational inference [35], we propose two scalable efficient inference algorithms based on amortized inference [18, 37] for offline and online learnings, respectively.

Our contributions can be summarized as follows:

(1) We proposed a novel algorithm, DCRL, which is the first attempt to unify two directions of recommendations: temporal and sequential recommendations into a single learning framework. DCRL takes advantages from both directions of sequential and temporal methodologies: collaboratively capturing users' short-term and long-term preferences, and explicitly dynamically tracking the evolution of users' long-term preferences.

(2) Since our model requires inferring huge users' latent vectors across all time, we develop two efficient inference algorithms for DCRL, offline and online, based on amortized variational inference.

(3) Experiments demonstrate DCRL outperforms state-of-the-art methods and does collaboratively capture of users' short-term and long-term preferences and track the evolutions of users' long-term preferences over time.

2 RELATED WORK

In this section, we briefly discuss two lines of related work, temporal and sequential recommendation methods.

2.1 Temporal recommendation methods

To model the dynamics of users' long-term preferences, temporal recommendation methods have been proposed in the literature. TimeSVD++ [19] is the first successful temporal recommendation method which is built on a time-changing factor model. To avoid feature engineering and be able to predict future ratings without advance knowledge of future data, Wu et al. [42] proposed recurrent recommender networks which utilize RNN to model the dynamics of users' long-term preferences. To model time probabilistically, several works, e.g., collaborative Kalman fitering [12], dynamic Poisson factorization [4] and streaming recommender system [3], use linear Gaussian SSM to track the evolution of users' long-term preferences over time. Since the Gamma-Poisson structure enjoys more efficient inference and better handling of sparse data than the Gaussian state space model [3], more recent works, e.g., dynamic collaborative filtering with compound Poisson factorization [17], Gamma-Poisson dynamic matrix factorization with metadata [7], model long-term preferences over time using gamma-Markov chains, and dynamic bayesian metric learning for product search [45]. All of the temporal recommendation methods so far, including the above, still work with matrix completion at each single time window, which limits them to capture short-term preferences.

2.2 Sequential recommendation methods

Another line of research is sequential recommendation [14, 22, 28, 39]. Different from temporal recommendation methods, the main goal of sequential recommendation methods is to capture sequential patterns in a short time window (short-term preferences). To achieve this, Latent Markov Embedding [5, 24, 25] have been proposed via utilizing the first-order Markov chains. Since RNN has shown to have an impressive capability in capturing sequence patterns, a number of RNN based sequential recommendations also called session-based recommendation methods [14, 22, 28] have been proposed. More recently, personalized sequential recommendation methods have been proposed, including hierarchical representation model [40], hierarchical RNN model [34] and convolutional sequence embedding recommendation model [39] to incorporate users' long-term preferences into sequential models. However, all these personalized methods consider users' long-term preferences as general preferences [23, 27, 39] that are static over time. Compared with their work, we propose Dynamic Collaborative Recurrent Learning (DCRL) to unify the two directions: temporal and sequential recommendations. DCRL seamlessly combines RNN and SSM to capture the short-term preferences and explicitly track the evolution of long-term preferences over time. To the best of our knowledge, we are the first to unify temporal and sequential recommendation methods into a single learning framework.

3 DYNAMIC COLLABORATIVE RECURRENT LEARNING

In this section, we first detail our DCRL. Then, we present the two inference algorithms for offline and online learning, respectively.

3.1 Task formalization

Unlike pervious sequential recommendation methods [34, 39, 40], we consider the interacted item sequences of user u over time. Formally, let $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ be sets of users and items, with $|\mathcal{U}|$ and $|\mathcal{V}|$ being the sizes, respectively. Let the observed interacted item sequences for user u up to time T as $\mathbf{s}^u_{1:T} = \{\mathbf{s}^u_1, \mathbf{s}^u_2, ..., \mathbf{s}^u_T\}$, where $\mathbf{s}^u_t = \{v^u_{1,t}, v^u_{2,t}, ..., v^u_{|\mathbf{s}^u_t|,t}\}$ denotes

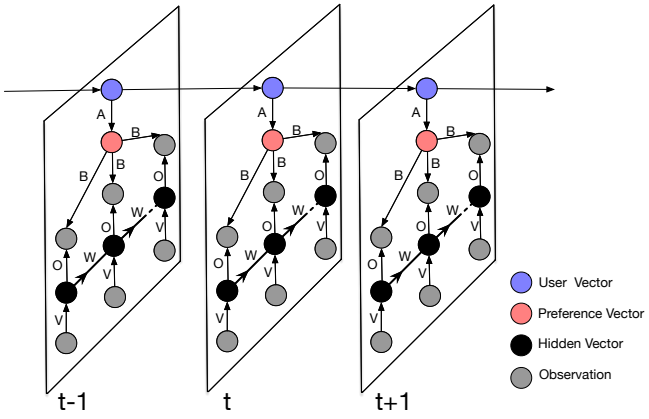

Figure 1: Graphical representation of our collaborative recurrent learning model, DCRL.

the interacted item sequence of user u at time window t. The subscript k and t for $v^u_{k,t}$ denotes the order in which an action occurs in the sequence $\mathbf{s}^u_t$ at time window t. We use $\mathbf{u}_{1:T} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_T\}$ to denote the latent vectors of user u up to time T. We also use $\mathbf{S}_{1:T}$ and $\mathbf{U}_{1:T}$ to denote the all observed sequences and latent vectors of all users across all time windows, respectively. Given interacted item sequences of user u up to time T, the goal of DCRL is to capture the long-term preferences of user u over time $\mathbf{u}_{1:T}$ and predict the clicked probabilities of the items that u will interact over time.

3.2 Collaboratively modeling short-term and long-term preferences

Most past sequential recommendation models [14, 22, 34] are based on traditional RNN structure introduced in [14]. However, it can only capture users' short-term preferences in a short time window and predict the next item based on the previous sequences. This is unrealistic, as the probability of clicking next item should not only depend on the the previous sequences. In other words, the users' personalized long-term preferences should also contribute to the next item clicked probability. In this section, we introduce a novel RNN to collaboratively model the users' short-term and long-term preferences via introduced latent user vectors. Our DCRL is a deep generative model. For a user u, we first draw a user latent vector:

$$\mathbf{u} = \mathcal{N}(\mathbf{0}, \mathbf{I}_{k_u}), \tag{1}$$

where $\mathbf{I}_{k_u}$ is identity matrix of size k_u and $\mathcal{N}$ is the Gaussian distribution. Aiming to let users' latent vectors represent their long-term preferences over difference topics, we further pass $\mathbf{u}$ into a softmax function to parametrize the multinomial topic distributions of users' long-term preferences:

$$\mathbf{p} = f(\mathbf{u}) = \text{softmax}(\mathbf{A}\mathbf{u} + \mathbf{b}_\mathbf{u}), \tag{2}$$

where $\mathbf{A}$ and $\mathbf{b}_\mathbf{u}$ are the trainable parameters. This transformation provides an interpretable way to understand the different topics of long-term preferences. This formulation is inspired from recent neural variational topic modeling [31, 38] and the $\mathbf{p}$ can be seen as user preferences over different topics. Given the clicked sequence $\mathbf{v}^u_{1:k-1} = \{v^u_1, \ldots, v^u_{k-1}\}$, we first recursively compute the hidden vectors $\mathbf{h}_k$ in RNN:

$$\mathbf{h}_{k'} = GRU(v^u_{k'-1}, \mathbf{h}_{k'-1}).\ k' = 1, 2, ..., k \tag{3}$$

where the GRU represents the GRU unit [6]. The likelihood of the observed next clicked item i is defined by a softmax function as:

$$p(v^u_k = i|\mathbf{h}_k, \mathbf{u}) = \frac{\exp(\mathbf{o}_i^\top \mathbf{h}_k + \mathbf{b}_i^\top \mathbf{p})}{\sum_i^{|\mathcal{V}|} \exp(\mathbf{o}_i^\top \mathbf{h}_k + \mathbf{b}_i^\top \mathbf{p})}. \tag{4}$$

Compared with traditional RNN based model [14, 22, 34], we add a bias term computed by the dot product between the user latent vector and the additional weight matrix $\mathbf{B}$ to directly contribute to the next item prediction. The way that we incorporate users' long-term preferences as bias terms makes DCRL be able to clearly distinguish the contribution between users' long-term preferences and short-term preferences on the next item prediction and easily to explicitly model the dynamic of long-term preferences. Based on the generative process introduced above, the likelihood of the observed clicked sequence of user u can be written as follows:

$$p_\theta(\mathbf{s}^u|\mathbf{u}) = \prod_{k=1}^{|\mathbf{s}^u|-1} p_\theta(v^u_k|\mathbf{h}_k, \mathbf{u}), \tag{5}$$

where θ is used to denote the parameters in generative model, i.e., the parameters in GRU unit, the weight matrix $\mathbf{A}_u$ and bias $\mathbf{b}_u$.

3.3 Modeling long-term preferences over time

To dynamically model users' long-term preferences, we should consider a transition probability to model the drift of the long-term preferences between time $t-1$ and t. In other words, we need extend the prior of latent user vector (Eq 2) into a dynamic-style prior. In previous dynamic recommendation methods [3, 4, 29, 46, 49], most of them use the following Kalman filter as a prior as the transition probability defined in Linear Gaussian State Space Model (LGSSM):

$$p(\mathbf{u}_t|\mathbf{u}_{t-1}) = \mathcal{N}(\mathbf{W}_\mu \mathbf{u}_{t-1}, \text{diag}(\sigma^2)). \tag{6}$$

This transition is in fact a linear transformation which is often restrictive for our model since we don't know how the users' long-term preferences drifts in real world. To address this, we define a more generalized non-linear transition probabilities as follows:

$$p(\mathbf{u}_1|\mathbf{u}_0) = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{7}$$

$$p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1}) = \mathcal{N}(\mu_\psi(\mathbf{u}_{t-1}), \text{diag}(\sigma^2))\ \ t > 1, \tag{8}$$

where the mean $\mathbf{u}_t$ is parameterized by a neural network that we called it as transition neural network (TNN) that depends on $\mathbf{u}_{t-1}$: $\mu_\psi(\mathbf{u}_{t-1}) = \mathbf{W}_3\text{TNN}(\mathbf{u}_{t-1}) + \mathbf{b}_3$. ψ denotes the all weights and biases in TNN, $\mathbf{W}_3$ and $\mathbf{b}_3$. The TNN is a one-layer feed-forward network in our implementation. Intuitively, If the TNN is a simple identity function and $\mathbf{b} = \mathbf{0}$, we can exactly recover the mean in Kalman filter prior (Eq. 6). So Eq. 8 defines a large family of linear and non-linear Gaussian state space models. Note that we don't use neural networks to parameterize the variance matrix $\text{diag}(\sigma^2)$. Instead, we consider it as the global parameter. This simplifying assumption follows from that of traditional dynamic recommendation methods [3, 4, 29], but could easily be relaxed. The prior $p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})$ allows the DCRL to smoothly detect the evolution of users' long-term preferences across time via a non-linear way, and

can share information across time slices which makes DCRL able to be effectively learned in every time slices where the feedbacks are very spare, as long as the feedbacks in total are large. Using the generalized transition probability coming with a additional challenge, since inference becomes intractable, we need to resort to approximate methods. See § 3.4 and § 3.5 for more details. For the generative process of observed sequence at time t in our DCRL, we parameterize it using the proposed RNN structure (see Fig. 1) in the § 3.2 (Eq. 2, 3 and 4). Thus the likelihood of the all observed sequences $\mathbf{s}^u_{1:T}$ of user u across all time windows can be written as:

$$p_\theta(\mathbf{s}^u_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T}\prod_{k=1}^{|\mathbf{s}^u_t|-1} p_\theta(v^u_{k,t}|\mathbf{h}_{k,t},\mathbf{u}_t). \tag{9}$$

3.4 Offline Learning

Estimating the parameters of RNN and TNN needs to infer the latent vectors $\mathbf{U}_{1:T}$ at each time. In this section, we first consider offline scenario in which we need to infer the user latent vectors $\mathbf{U}_{1:T}$ given the entire observed item sequences $\mathbf{S}_{1:T}$ for all users up to time T. This is also called (Kalman) smoothing scenario [41]. We can not directly infer the posterior of the latent user matrix $\mathbf{U}_{1:T}$ and estimate the network parameters θ because it is difficult to marginalize out the latent user vector $\mathbf{u}_t$ in Eq. 9, particularly when the number of users is very large. To address this problem, we propose an inference algorithm base on variational inference [1]. We use $q(\mathbf{u}_t;\alpha^u_t)$ to denote the variational distribution of user u at time t with variational parameter α^u_t. For simplicity and effectiveness, We adopt the mean-field [1] approximation inference strategy, and consider the following variational distribution for user u that factorizes across all times:

$$q(\mathbf{u}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{u}_t) = \prod_{t=1}^{T} q(\mathbf{u}_t;\alpha^u_t). \tag{10}$$

In variational inference framework, we could derive the variational **E**vidence **L**ower **BO**und (ELBO) corresponding to the log-marginal likelihood of observed item sequences across all time windows using Jensen's inequality as follows:

$$\begin{aligned}
&\log p_{\theta,\psi}(\mathbf{S}_{1:T}) = \log \int p_{\theta,\psi}(\mathbf{S}_{1:T},\mathbf{U}_{1:T})d\mathbf{U}_1\ldots d\mathbf{U}_T = \\
&\log \sum_{u\in\mathcal{U}} \int \prod_{t=1}^{T} q(\mathbf{u}_t) \prod_{t=1}^{T} \frac{p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})p_\theta(\mathbf{s}^u_t|\mathbf{u}_t)}{q(\mathbf{u}_t)} d\mathbf{u}_1\cdots d\mathbf{u}_T \geqslant \\
&\sum_{u\in\mathcal{U}} \int \prod_{t=1}^{T} q(\mathbf{u}_t) \sum_{t=1}^{T} \log \frac{p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})p_\theta(\mathbf{s}^u_t|\mathbf{u}_t)}{q(\mathbf{u}_t)} d\mathbf{u}_1\cdots d\mathbf{u}_T = \\
&\sum_{u\in\mathcal{U}}\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{u}_t)}\big[\log p_\theta(\mathbf{s}^u_t|\mathbf{u}_t)\big] - \mathbb{KL}(q(\mathbf{u}_1)||p_\psi(\mathbf{u}_1|\mathbf{u}_0)) - \\
&\sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{u}_{t-1})}\big[\mathbb{KL}(q(\mathbf{u}_t)||p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})\big] = \mathcal{L}.
\end{aligned} \tag{11}$$

Maximizing the ELBO $\mathcal{L}$ respect to the variational distribution $q(\mathbf{u}_t;\alpha^u_t)$ is also equal to minimize the **K**ullback-**L**eibler (KL) divergence between the variational distribution $q(\mathbf{u}_t;\alpha^u_t)$ and the true posterior distribution $p(\mathbf{u}_t|\mathbf{s}^u_{1:T})$. Due to the large number of users and time windows, we can not store and iteratively optimize every variational parameters α^u_t. To address this problem, we use amortized inference [11, 18, 37] which can avoid the optimization of the parameter α^u_t for each user variational distribution $q(\mathbf{u}_t;\alpha^u_t)$ at t; instead, it uses a shared structure to calculate each variational parameter. Specifically, similar to pervious works [18, 37], we design a shared inference neural networks (INN) for all users and all time windows denoted as inference network, as follows:

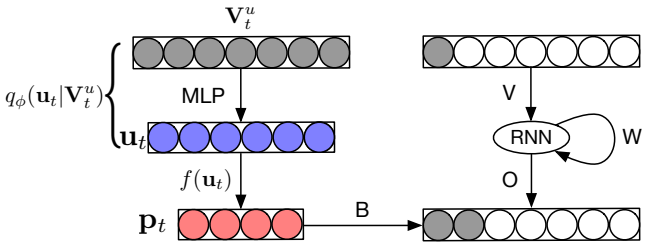

Figure 2: Inference neural network (INN) structure of the proposed DCRL.

$$q_\phi(\mathbf{u}_t|\mathbf{V}^u_t) = \mathcal{N}(\mathbf{u}_t;\mu(\mathbf{V}^u_t),\mathrm{diag}(\sigma^2(\mathbf{V}^u_t))), \tag{12}$$

$$\mu(\mathbf{V}^u_t) = \mathbf{W}_1\mathrm{INN}(\mathbf{V}^u_t) + \mathbf{b}_1, \tag{13}$$

$$\log\sigma(\mathbf{V}^u_t) = \mathbf{W}_2\mathrm{INN}(\mathbf{V}^u_t) + \mathbf{b}_2, \tag{14}$$

where $\mathbf{V}^u_t \in \mathbb{N}^{|\mathcal{V}|}$ denotes the item-frequency representation interacted by user u at time window t. The outputs of the inference network are the parameters of variational distribution on $\mathbf{u}_t$. Fig. 2 shows the INN structure of DCRL. We use ϕ to denote the all parameters of the INN. Another problem of maximizing the ELBO (Eq. 11) is that we can not have closed-form solution due to non-conjugacy in our model. To tackle this, we thus estimate it via Monte Carlo estimates and the reparameterization trick [18, 37]. The KL terms in ELBO between two Gaussians have closed forms. Thus the ELBO (Eq. 11) can be rewritten as follows:

$$\begin{aligned}
\mathcal{L} = \mathcal{L}(\theta,\psi,\phi) = &\sum_{u\in\mathcal{U}}\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{u}_t|\mathbf{V}^u_t)}\big[p_\theta(\mathbf{s}^u_t|\mathbf{u}_t)\big] + \frac{Tk_u}{2} + \\
&\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{u}_{t-1}|\mathbf{V}^u_t)}\big[(\mu_\psi(\mathbf{u}_{t-1}) - \mu(\mathbf{V}^u_t))^\top(\sigma^2)^{-1}(\mu_\psi(\mathbf{u}_{t-1}) - \mu(\mathbf{V}^u_t))\big] \\
&- \frac{1}{2}\Big(\sum_{t=1}^{T}\sum_{i=0}^{k_u-1} \log\sigma^2_i - \log\eta^2_{t,i} + \frac{\eta^2_{t,i}}{\sigma^2_i}\Big),
\end{aligned} \tag{15}$$

where σ^2_i and $\eta^2_{t,i}$ are the i-th component of diagonals σ^2 and $\mathrm{diag}(\sigma^2(\mathbf{V}^u_t))$ respectively. Then we can use stochastic optimization methods such as SGD or Adagrad [8] to iteratively optimize the parameters θ,ϕ,ψ using the gradient $\nabla_{\theta,\phi,\psi}\mathcal{L}(\theta,\phi,\psi)$ based on mini-batch. Moreover, the mean-filed strategy breaks the temporal dependency of the ELBO, which comes with a computational advantages. Instead of using the ancestral sampling [9], we can compute the last expectation KL term in ELBO (Eq. 11) in parallel by sampling mini-batch in the arbitrary two adjacent time windows. Once the iteration process converge, we obtain the optimal network parameters, θ^*, ψ^* and ϕ^*. The posteriors of user u in all

Algorithm 1: The inference algorithm for offline learning.

Input: Observed clicked sequences $\mathbf{S}_{1:T}$, item-frequency representation $\mathbf{V}_{1:T}$.
Output: Dynamic posterior $q(\mathbf{u}_{1:T})$.
Randomly initialize the network parameters θ, ψ and ϕ.
while *not convergent* **do**
  Sample two arbitrary adjacent time windows: t and $t+1$, and a mini-batch user set $\mathcal{U}$.
  for $u \in \mathcal{U}$ **do**
    Draw two samples from the variational distributions of $q_\phi(\mathbf{u}_t|\mathbf{V}_t^u)$ and $q_\phi(\mathbf{u}_{t+1}|\mathbf{V}_{t+1}^u)$.
    Approximate the two expectation terms in Eq. 11 using the reparameterization trick.
  end
  Update the network parameters using the gradient $\nabla_{\theta,\phi,\psi}\mathcal{L}(\theta,\phi,\psi)$ of the mini-batch ELBO.
end
Obtain posterior $q(\mathbf{u}_{1:T})$ for each user u (Eq. 16).

time windows, $p(\mathbf{u}_{1:T}|\mathbf{s}_{1:T}^u)$ can be inferred as follows:

$$p(\mathbf{u}_{1:T}|\mathbf{s}_{1:T}^u) \approx q(\mathbf{u}_{1:T}) = \left\{q_{\phi^*}(\mathbf{u}_t|\mathbf{V}_t^u)\right\}_{t=1}^T. \tag{16}$$

3.5 Online Learning

Unlike offline learning that is conditioned on the entire item sequence of observation $\mathbf{S}_{1:T}$. In many real streaming scenarios where the data arrives sequentially, we can only condition on past and not on future observations called online learning and the huge volume of feedback is hard to be materialized in memory, which necessitates the design of a cost-effective learning and inference algorithm with limited memory. In additional, offline inference algorithm need recompute the latent user posteriors of all the users from scratch at every time stamp. Accordingly, in what follows, we propose an online learning algorithm which can iteratively approximate the posterior $p(\mathbf{u}_t|\mathbf{s}_{1:t}^u)$ as the feedbacks from each time step t become available for DCRL, without the need to revisit past data or have advance knowledge of future data. This is also called the filtering problem [41]. We begin by autoregressively decomposing the log marginal likelihood of observed sequences as follows:

$$\log p_{\theta,\psi}(\mathbf{S}_{1:T}) = \sum_{u\in\mathcal{U}}\sum_{t=1}^{T}\log p_{\theta,\psi}(\mathbf{s}_t^u|\mathbf{s}_{1:t-1}^u). \tag{17}$$

We further factorize the above log-marginal likelihood of user u as follows by considering the conditional independence properties $\mathbf{s}_t^u \perp\!\!\!\perp \mathbf{s}_{1:t-1}^u|\mathbf{u}_t$ of the DCRL (see Fig. 1) using *d-separation* [10]:

$$\log p_{\theta,\psi}(\mathbf{s}_t^u|\mathbf{s}_{1:t-1}^u) = \log\int p_\theta(\mathbf{s}_t^u|\mathbf{u}_t)p(\mathbf{u}_t|\mathbf{s}_{1:t-1}^u)d\mathbf{u}_t. \tag{18}$$

Given the above factorization, the ELBO respect to the log marginal likelihood up to time t for user u can be derived as follows:

$$\log p_{\theta,\psi}(\mathbf{s}_t^u|\mathbf{s}_{1:t-1}^u) \geqslant \tilde{\mathcal{L}}_t^u(\theta,\phi,\psi) = \mathbb{E}_{q_\phi(\mathbf{u}_t)}\big[\log p_\theta(\mathbf{s}_t^u|\mathbf{u}_t)\big] - \mathbb{E}_{q_\phi(\mathbf{u}_t)}\big[\log q_\phi(\mathbf{u}_t)\big] + \mathbb{E}_{q_\phi(\mathbf{u}_t)}\big[\log p(\mathbf{u}_t|\mathbf{s}_{1:t-1}^u)\big], \tag{19}$$

where the $q_\phi(\mathbf{u}_t)$ is the introduced variational distribution. Maximizing the ELBO $\tilde{\mathcal{L}}_t^u$ above is equal to approximate the following KL divergence between the variational distribution $q_\phi(\mathbf{u}_t)$ and the true posterior distribution $p(\mathbf{u}_t|\mathbf{s}_{1:t}^u)$ at time t:

$$q(\mathbf{u}_t) = \arg\min_{q_\phi(\mathbf{u}_t)} \mathbb{KL}(q_\phi(\mathbf{u}_t)||p(\mathbf{u}_t|\mathbf{s}_{1:t}^u)). \tag{20}$$

Since the number of users is large and the time windows may be also long, similarly to the proposed offline inference § 3.4, we thereby define the introduced variational distributions $q_\phi(\mathbf{u}_t)$ as the following distribution which is parameterized by a shared inference network:

$$q_\phi(\mathbf{u}_t) = \mathcal{N}(\tilde{\mu}_t^u, \tilde{\Sigma}_t^u) = \mathcal{N}(\mu(\mathbf{V}_t^u), \mathrm{diag}(\sigma^2(\mathbf{V}_t^u)), \tag{21}$$

where the $\mathbf{V}_t^u \in \mathbb{N}^{|\mathcal{V}|}$ denotes the item-frequency representation interacted by user u at time window t. Specifically, similarly to the proposed inference algorithm in offline learning § 3.4, the inference network parameterized by ϕ takes the $\mathbf{V}_t^u$ as the inputs, and outputs the mean $\tilde{\mu}_t^u$ and log-variance $\tilde{\Sigma}_t^u$ of the variational distribution $q_\phi(\mathbf{u}_t)$, respectively. The first expectation term in ELBO (Eq. 19) can be estimated via Monte Carlo estimates and the reparameterization trick [18, 37]. The second term can be computed analytically. For the third term, we can not directly solve it analytically. We recursively approximate it as follows:

$$\begin{aligned}&\mathbb{E}_{q_\phi(\mathbf{u}_t)}\big[\log p(\mathbf{u}_t|\mathbf{s}_{1:t-1}^u)\big] = \\ &\mathbb{E}_{q_\phi(\mathbf{u}_t)}\Big[\log\int p(\mathbf{u}_{t-1}|\mathbf{s}_{1:t-1}^u)p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})d\mathbf{u}_{t-1}\Big] \approx \\ &\mathbb{E}_{q_\phi(\mathbf{u}_t)}\Big[\log\int q(\mathbf{u}_{t-1})p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})d\mathbf{u}_{t-1}\Big] = \mathbb{E}_{q_\phi(\mathbf{u}_t)}\big[\log p_\psi(\mathbf{u}_t)\big],\end{aligned} \tag{22}$$

where the second line is obtained from the conditional independence properties: $\mathbf{u}_t \perp\!\!\!\perp \mathbf{s}_{1:t-1}^u|\mathbf{u}_{t-1}$ using *d-separation* [10], and the approximation at the third line is obtained from that we already approximate $p(\mathbf{u}_{t-1}|\mathbf{s}_{1:t-1}^u)$ at pervious time window $t-1$: $q(\mathbf{u}_{t-1}) \approx p(\mathbf{u}_{t-1}|\mathbf{s}_{1:t-1}^u)$. Since $p_\psi(\mathbf{u}_t|\mathbf{u}_{t-1})$ is parameterized by the transition neural network, the last integral in Eq. 22 can not be computed analytically and we approximate it using Monte Carlo estimates. The resulting approximate prior $p_\psi(\mathbf{u}_t)$ is approximated as:

$$p_\psi(\mathbf{u}_t) = \mathcal{N}(\mathbf{u}_t;\mu_t^u,\Sigma_t^u) = \mathcal{N}(\frac{1}{M}\sum_{m=1}^{M}\mu_\psi(\mathbf{u}_{t-1}^m), \sigma^2\mathbf{I}), \tag{23}$$

where the $\mathbf{u}_{t-1}^m$ is the m-th sample drawn from $q(\mathbf{u}_{t-1})$ using reparameterization trick [18, 37] and M is the total sampling number. Given the above discussion, the ELBO (Eq. 19) can be approximated as follows:

$$\tilde{\mathcal{L}}_t^u(\theta,\phi,\psi) \approx \log p_\theta(\mathbf{s}_t^u|\hat{\mathbf{u}}_t) - \mathbb{KL}(q_\phi(\mathbf{u}_t)||p_\psi(\mathbf{u}_t)), \tag{24}$$

where the $\hat{\mathbf{u}}_t$ is the sample drawn from $q_\phi(\mathbf{u}_t)$ using reparameterization trick [18, 37]. Then we can use stochastic optimization methods such as SGD or Adagrad [8] to optimize the parameters θ, ϕ, ψ using the gradients $\nabla_{\theta,\phi,\psi}\tilde{\mathcal{L}}_t^u(\theta,\phi,\psi)$. In summary, the inference algorithm for online learning can be described as Fig. 3.

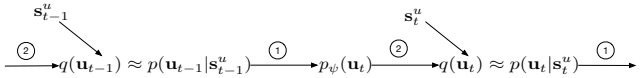

Figure 3: Patterns of our online inference algorithm. ① We first learn the approximated prior $q_\psi(\mathbf{u}_t)$ from the obtained posterior $p(\mathbf{u}_{t-1}|\mathbf{s}^u_{t-1})$ (Eq 23), and ② we then optimize the ELBO at time t (Eq 24) to obtain the variational distribution $q_\phi(\mathbf{u}_t)$ which is the approximate of true $p(\mathbf{u}_t|\mathbf{s}^u_t)$ at time t (Eq 20).

Table 1: The dimensions of the key network parameters.

V	W	O	B	A	$\mathbf{W}_1$	$\mathbf{W}_2$	$\mathbf{W}_3$
$\|\mathcal{V}\|\times$h	$h \times h$	$h \times \|\mathcal{V}\|$	$k_p \times \|\mathcal{V}\|$	$k_u \times k_p$	$d_i \times k_u$	$d_i \times k_u$	$d_t \times k_u$

3.6 Recommendation

In particular, we consider two different recommendation scenarios: personalized within-time window recommendation, and next-time window recommendation:

Within-time window recommendation. We directly utilize the learned posterior at time window t to recommend items at time t as follows:

$$p_{\theta^*}(v^u_{k,t} = i|\mathbf{s}^u_{1:k-1,t}) = p_{\theta^*}(v^u_{k,t} = i|\mathbf{h}_{k,t}, \tilde{\mu}^u_t), \quad (25)$$

where the $\tilde{\mu}^u_t$ is the mean of $q(\mathbf{u}_t)$.

Next-time window recommendation. In this configuration, we need predict the item sequence which user u would like click at time $t+1$ given the pervious observed sequences $\mathbf{s}^u_{1:t}$. We take the latest feedbacks as input at each time, update users' posteriors and make predictions based on the newly updated posteriors. Formally, the predicted probability of item sequence for user u in time window $t+1$ is:

$$p_{\theta^*}(\mathbf{s}^u_{t+1}|\mathbf{s}^u_{1:t}) = \int p_{\theta^*}(\mathbf{s}^u_{t+1}|\mathbf{u}_{t+1})p(\mathbf{u}_{t+1}|\mathbf{s}^u_{1:t})d\mathbf{u}_{t+1} =$$
$$\int p_{\theta^*}(\mathbf{s}^u_{t+1}|\mathbf{u}_{t+1})p_{\psi^*}(\mathbf{u}_{t+1})d\mathbf{u}_{t+1} \approx p_{\theta^*}(\mathbf{s}^u_{t+1}|\mu^u_{t+1}), \quad (26)$$

where μ^u_{t+1} is the mean of $p_{\psi^*}(\mathbf{u}_{t+1})$. For the specific item in sequence $\mathbf{s}^u_{t+1}$, the item clicked probability at each step in time window $t+1$ as follows:

$$p_{\theta^*}(v^u_{k,t+1} = i|\mathbf{s}^u_{1:k-1,t+1}) = p_{\theta^*}(v^u_{k,t+1} = i|\mathbf{h}_{k,t+1}, \mu^u_{t+1}). \quad (27)$$

3.7 Complexity analysis

Our algorithm builts based on neural networks, the time complexity depend on the number of the network parameters. We give the notations of the dimensions of the key network parameters. $|\mathcal{V}|$ is the item size, h is number of the hidden units of the RNN, k_u and k_p are the dimensions of latent user and preference vectors respectively. Table 1 gives the dimensions of key parameters in DCRL model (omitting the biases). Since the sizes of k_u, k_p, d_i and d_t are much smaller than $|\mathcal{V}|$, the model complexity of the propose models is at the same level as the vanilla RNN whose parameter size is (**O+V+W**). The computation cost of offline learning at one iteration is as same as online learning. However the offline learning need recompute all posteriors at every time, it typically needs more iterations to converge.

4 EXPERIMENTAL SETUP

4.1 Research questions

We seek to answer the following research questions that guide the remainder of the paper:

(RQ1) Can our DCRL outperform vs. state-of-the-art methods for difference recommendation scenario such as within-time window (**RQ1-1**) and next-time window recommendations (**RQ1-2**)?

(RQ2) Is the proposed DCRL sensitive to the latent topic numbers?

(RQ3) Is the proposed online learning algorithm effective to recursively infer users' long-term preferences over time (**RQ3-1**)? and Is the deep transition neural network more effective than traditional linear transition probability for online learning (**RQ3-2**)?

(**RQ4**) Can DCRL capture high-quality users' long-term preferences and their evolutions?

4.2 Datasets and data preprocessing

In order to answer our research questions, we work with these publicly available datasets:

(1) **Movielens-20M**[1]: this is a widely used movie datasets which consists of 20m ratings collected between 1995 and 2015. Since the rating values are scale from 0.5 to 5, following pervious work [39], we convert the rating values into implicit feedbacks. We denote this dataset as **Movielens** for brevity.

(2) **LFMTracks**[2]: this is a widely used music dataset which contains the number of times a user listened to a song between Feb.2005 and Jun.2009 [2].

(3) **Tmall**[3]: this comes from IJCAI-15 competition and contains 23k users' shopping logs over six months in a Chinese online shopping website Tmall.

(4) **Gowalla**[4]: This dataset contains 70K Gowalla check-ins made within California and Nevada between Feb. 2009 and Oct. 2010.

We split each dataset into time windows using the time stamps. Specifically, the durations of time windows for the **Movielens** and **LFMTracks** are set to two year and three months, respectively. For **Tmall** and **Gowalla**, the user tastes might evolve over a half month and two months, respectively. Note that the duration can be selected manually to suit different datasets in real world. We use the four dataets to confirm if our model can collaboratively balance the impact of sequential and temporal signals in different type of datasets and achieve better performance. Table 2 shows the statistics of four datasets.

4.3 Baselines and Settings

Baselines. We make comparisons between our proposed models and the following state-of-the-art recommendation methods:

Static methods:

[1]https://grouplens.org/datasets/movielens/20m/
[2]http://ocelma.net/MusicRecommendationDataset/index.html
[3]https://ijcai-15.org/index.php/repeat-buyers-prediction-competition/
[4]https://snap.stanford.edu/data/loc-gowalla.html

NCF: This method [13] is the state-of-the-art static recommendation method which utilizes neural networks to model the interactions between users and items.

Temporal methods:

TSVD++: This method (TimeSVD++) [19] is a temporal recommendation method which is the first work to consider the evolution of users' latent long-term preferences.

SRS: This method [3] is a temporal method which is based on streaming feedbacks. It uses Brownian motion [3] to dynamically model users' long-term preferences over time.

RRN: This method [42] is the state-of-the-art temporal recommendation which utilizes RNN to dynamically track the evolution of users' long-term preferences and incorporate users' general preferences via matrix factorization at each time window.

Sequential methods:

GRU4Rec: This method [14] utilizes the basic GRU [6] unit to the task of session-based recommendation. It only can capture users' short-term preferences in a short time window.

FPMC: This method [36] is a hybrid model which combines the Markov chain and matrix factorization for next-basket recommendation. It can capture both short-term preferences and general preferences of users.

STAMP: This method [28] utilizes a short-term attention/memory mechanism to capture users' long-term and short-term preferences.

To further validate the benefits brought by tracking the evolution of users' long-term preferences, the effectiveness of online and offline learning algorithms. We consider three variant versions of our proposed DCRL model:

DCRL: This is a static version of DCRL which captures both users' short-term preferences and long-term preferences via an introduced latent vectors and uses static prior for the user latent vector (Eq 1).

DCRL-off: This Dynamic Collaborative Recurrent learning model proposed in this paper conduct proposed offline inference.

DCRL-on: This Dynamic Collaborative Recurrent learning model proposed in this paper conduct proposed online inference.

For all the baselines, we use a grid search to find the optimal hyper-parameters using the validation set. For our DCRL, we first randomly initialize model parameters using uniform distribution with variance 0.01 and bases being 0. We use one-layer GRU. The hyper-parameters are found via extensive grid search. We use AdaGrad [8] with momentum to optimize our model. We use a drop regularization on the hidden states of GRU with a fixed drop rate of 0.1. The sampling number M is set to 128.

Scenario Simulation. Similar to that in [15, 28, 42], we perform our model and the baselines in the within-time window and next-time window recommendation tasks. For within-time window recommendation, following pervious works [15, 28], we hold out the last items of all users in the last slice $\mathbf{S}_t$ as the test data and the items before the last are treated as the validation set. For next-time window recommendation, following [15], all baselines and our model are trained using the all time slices $\mathbf{S}_{1:t-2}$, the last slice $\mathbf{S}_t$ as the test data and the slice $\mathbf{S}_{t-1}$ as the validation set. Since the item candidate set is very large, following to previous works [13, 15], we pair each positive feedback with n negative feedbacks. n is 500,100,100,100 for MovieLens, Lastfm, Tmall, and Gowalla, respectively. To reduce biases, a half of the negative feedbacks is randomly sampled and the another half is sampled according to the popularity.

Table 2: Statistics of datasets.

Datasets	Movielens	LFMTracks	Tmall	Gowalla
Users	14k	0.9k	24k	13.1k
Items	27k	1M	13k	14k
Time windows	10	16	12	10
Sparsity	99.5%	99.95%	99.89%	99.71%

Table 3: Within-time window recommendation performance on the Movielens, LFMTracks, Tmall and Gowalla datasets in terms of Recall@10 (Rec) and NDCG@10 (NDCG).

	Movielens		LFMTracks		Tmall		Gowalla	
	Rec	NDCG	Rec	NDCG	Rec	NDCG	Rec	NDCG
NCF	.292	.172	.187	.115	.149	.089	.135	.103
TSVD++	.303	.187	.201	.127	.165	.097	.157	.112
SRS	.324	.227	.213	.136	.173	.103	.159	.121
RRN	.344	.247	.232	.142	.187	.112	.184	.124
FPMC	.335	.219	.221	.137	.217	.104	.175	.109
GRU4Rec	.359	.262	.247	.178	.238	.128	.186	.132
STAMP	.351	.268	.254	.192	.257	.137	.195	.138
DCRL	.363	.277	.256	.199	.264	.142	.203	.140
DCRL-on	.375	.285	.262	.208	.269	.149	.211	.145
DCRL-off	**.384**	**.294**	**.273**	**.219**	**.271**	**.154**	**.217**	**.151**

Table 4: Next-time window recommendation performance on the Movielens, LFMTracks, Tmall and Gowalla datasets in terms of Recall@10 (Rec) and NDCG@10 (NDCG).

	Movielens		LFMTracks		Tmall		Gowalla	
	Rec	NDCG	Rec	NDCG	Rec	NDCG	Rec	NDCG
NCF	.327	.189	.198	.149	.108	.078	.116	.087
TSVD++	.338	.219	.224	.163	.132	.082	.128	.093
SRS	.343	.257	.256	.185	.151	.095	.131	.107
RRN	.367	.273	.272	.198	.176	.105	.145	.119
FPMC	.342	231	.247	.162	.193	.097	.128	.098
GRU4Rec	.349	.254	.252	.171	.205	.119	.141	.121
STAMP	.361	.262	.259	.191	.217	.133	.153	.128
DCRL	.377	.278	.268	.197	.223	.139	.158	.139
DCRL-on	.381	.283	.279	.209	.231	.145	.161	.146
DCRL-off	**.389**	**.291**	**.287**	**.213**	**.235**	**.149**	**.165**	**.151**

Evaluation metrics. To evaluate the recommendation performance, as in [13, 14, 34], we use a number of widely used evaluation metrics, Recall@k and NDCG@k (Normalized Discounted Cumulative Gain at k) [47]. We compute the scores at depth 10, i.e., let k =10 for evaluation. For MRR@10, the reciprocal rank is manually set to zero if the rank is greater than 10. We report the average results of all test users.

5 RESULTS AND ANALYSIS

5.1 Quantitative results (RQ1)

RQ1-1. To evaluate the effectiveness of DCRL in within-time window recommendation task, we compare our methods to the state-of-the-art on four datasets. Table 3 shows the performance. From Table 3, we have the following conclusions: (1) Most methods capturing users' short-term preferences, i.e., STAMP, GRU4Rec and our proposed methods work better than temporal and general recommendation methods, i.e., NCF, SRS, TSVD++ and SRS on Tmall and Gowalla, which illustrates that users' short-term preferences need to be captured for within-time window recommendations on the Tmall and Gowalla datasets. (2) The propose models, i.e., DCRL, DCRL-on and DCRL-off perform better than the best baseline, i.e., STAMP on Tmall and Gowalla. A reason could be that different from STAMP, our DCRL explicitly and collaboratively captures short-term preferences and the evolution of long-term preferences across the time. (3) DCRL-off and DCRL-on significantly outperform DCRL on all datasets, which demonstrates that capturing the evolutions of long-term preferences does help improve the performance. (4) In terms of the inference algorithm, DCRL-off outperforms DCRL-on. This is expected since smoothing outperforms filtering as it shares information from both the past and the future.

RQ1-2. Next we consider the setting where recommending items at next time window. Table 4 shows the results. According to the table: (1) The variants of DCRL still dominate other baselines, which confirms that our models does effectively infer users' preferences for next-time window recommendation. (2) RRN becomes a competitive method and outperforms other baselines on Movielen and LFMTracks datasets, which demonstrates that capturing users' long-term preferences and their evolutions are more important than short-term preferences for next-time window recommendation on Movielens and LFMTracks datasets. It is reasonable since users' tastes on movie and music do not change most in a short time window. (3) Our models DCRL-off and DCRL-on significantly outperform RRN, which demonstrates that the latent variable characteristics make our model to more effectively capture the evolution users' long-term preferences. All of the findings from the two scenarios demonstrate that DCRL can effectively capture users' short-term preferences and the evolution of long-term preferences, and capturing this patterns does help to maintain significant improvements over the state-of-the-arts.

5.2 Impact of number of latent topics (RQ2)

RQ2. Subsequently, we examine the impact of the number of latent topics (the dimension of $\mathbf{p}$) on performance. Figure 4 shows the performance on different number of latent topics varying between 25 and 100 on the Movielen dataset. Performance evaluated on other datasets is not reported here, as it follows the same pattern. For clarity, we also report performance of the most strong baselines, i.e., RRN and STAMP, respectively. It is clear from the figure that the performance increases with larger number of latent topics. The performances reach a plateau when the number > 75. At all different topic numbers, DCRL keeps outperforming the best baselines. All of these findings demonstrate that our DCRL is not sensitive to the number of latent topics when the size is set to be large enough.

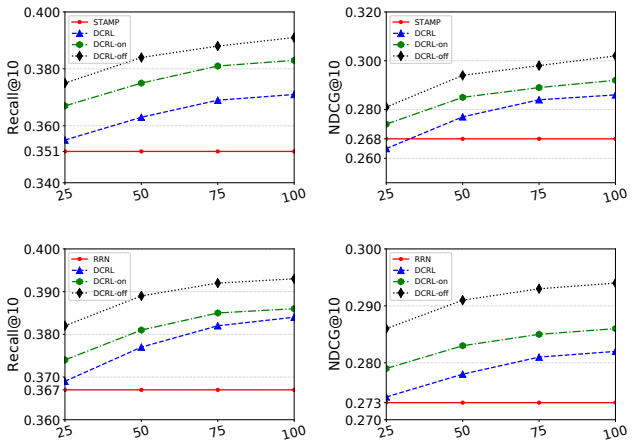

Figure 4: Within-time window (the top two) and next-time window (the bottom two) performances in terms of Recall@10 and NDCG@10 on Movielen dataset with different latent topic numbers.

5.3 Online learning (RQ3)

Next, we turn to **RQ3**, for analyzing the effectiveness of inferring users' long-term preferences over time of the proposed online inference algorithm. Following [3], we consider an online streaming scenario where the feedbacks come over the time. We recursively infer users' latent matrix $\mathbf{U}_t$ using DCRL-on by maximizing Eq. 24 with feedbacks $\mathbf{S}_t$ and evaluate the model at $t+1$ on next-time window recommendation. Next we infer $\mathbf{U}_{t+1}$ based on $\mathbf{S}_{t+1}$, evaluating at $t+2$ and so on. For the first time window, we recommend the most popular items in the training set. We compare the best temporal baselines, SRS and RRN, that infer users' long-term preferences over time with our DCRL-on in this scenario. To further show the effectiveness of non-linear transition neural networks, we also consider another variant of DCRL called DCRL-l whose transition function is replaced by a weight matrix so that where the transition function is Eq. 6. Figure 5 shows Recall@10 performances on Movielen and LFMTracks. The analysis on Tmall and Gowalla returns similar results.

Performance of online inference (RQ3-1). From Figure 5, we can find: (1) The two variants of DCRL, i.e., DCRL-l and DCRL-on significantly and consistently outperform, SRS and RRN, over the time, which demonstrates the effectiveness of proposed online inference for tracking long-term preferences' drifts. (2) DCRL-on's Recall grows over the first few time steps as it sees more data, and then saturates, which demonstrates it can share information across time to improve performance. This is crucial because there is little data at every time window.

Effectiveness of non-linear transition networks (RQ3-2). From Figure 5, we also find DCRL-on consistently outperforms DCRL-l on two datasets, which illustrates that the non-linear transition does capture better relationships across time and lead better performance.

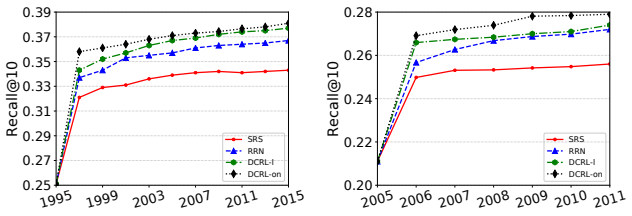

Figure 5: Next-time window recommendation performance in terms of Recall@10 through year with online inference on Movielen (the left) and LFMTracks (the right) datasets.

5.4 Qualitative analysis (RQ4).

Finally, we address **RQ4**. We apply our DCRL-on algorithm on the LFMTracks dataset and conduct case studies to explore the effectiveness of the learned users' long-term preferences and their evolutions. We first seek to understand the learned each user's long-term preferences by DCRL-on. Consider two representative users: 'U437' and 'U270' who have the same metadata ('age', 'gender' etc.) [5]. The left subfigure in Figure 6 shows the number of times that the two users listened to the 'rock' and 'pop' tracks with 16 time windows each of which consists of three months, and they both change their preferences from 'pop' to 'rock' music over time. To qualitatively evaluate the learned long-term preferences by DCRL-on, we select three time windows: the 5-th (A), 10-th (B) and 15-th (C) as shown in the left subfigure and present the inferred probability of each topic of long-term preference $\mathbf{p} = f(\mathbf{u})$ by estimating the posterior probability $q(\mathbf{u}|\mathbf{V}_t^u)$ of 'U437' and 'U270' of three time windows. From Figure 6, we can make the following observations: (1) The inferred topics of long-term preferences of two users have the similar distribution across the three time windows, consistent with the observed numbers of the two users' listen times. This confirms that DCRL-on can effectively infer long-term preferences for users at different time windows. (2) The evolution of inferred topic distributions of the two users' long-term preferences has the same tendencies (see the black boxes in the right subfigure of Figure 6) over time which is also consistent with changed numbers of listen times that users listened to the two 'rock' and 'pop' tracks. These observations indicate that DCRL can effectively capture the users' long-term preferences and their evolutions. We now turn to visualize all users' embeddings of the inferred long-term preferences at three time windows. We first use k-means algorithm to cluster the inferred preferences at time window A into 10 clusters, then use t-SNE [30] and keep its random seed to visualize them in Figure 7. Note that, to understand the evolution of the embeddings, all the points in the same cluster in time window A are marked with the same color and such marked color for all points is consistent in windows B and C whatever the cluster results change. From Figure 7, we find: (1) Compared to the initialization, embeddings of long-term preferences at A, B and C do have their cluster patterns and are different with each other, which demonstrates that our model can capture the long-term preferences over time. (2) The embeddings of the two users, i.e., 'U437' and 'U270', are always in the same cluster in different time windows. This once again confirms that DCRL-on does capture users' long-term preferences over time.

[5] http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html

6 CONCLUSIONS AND FUTURE WORKS

This paper has proposed a dynamic collaborative recurrent learning model, DCRL, that unifies two directions of recommendations: sequential and temporal recommendations, via a dynamic deep generative process. Such a unified framework takes advantages from both directions: (a) collaboratively capture users' short-term preferences and long-term preferences in a short time window as doing so in sequential methods; (b) dynamically track the evolution of users' long-term preferences as in temporal methods. Specifically, we first introduce latent user vectors, which serve as contextual biases to an RNN-based model, and then propose a latent SSM for non-linear evolution of users' long-term preferences. For the efficient inference purpose, we develop two inference algorithms, DCRL-off and DCRL-on, by leveraging amortization and stochastic gradients for offline and online learning, respectively. Experiment results demonstrate that the proposed algorithms can achieve better recommendation performance and capture users' short-term preferences and the evolution of long-term preferences.

Acknowledgments. This research was partially supported by the National Natural Science Foundation of China (Grant No. 61906219).

REFERENCES

[1] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.
[2] Oscar Celma. 2010. Music recommendation. In *Music recommendation and discovery*. Springer, 43–85.
[3] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A. Hasegawa-Johnson, and Thomas S. Huang. 2017. Streaming Recommender Systems. In *WWW*. 381–389.
[4] Laurent Charlin, Rajesh Ranganath, James McInerney, and David M Blei. 2015. Dynamic poisson factorization. In *RecSys*. ACM, 155–162.
[5] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *KDD*. 714–722.
[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[7] Trong Dinh Thac Do and Longbing Cao. 2018. Gamma-Poisson Dynamic Matrix Factorization Embedded with Metadata Influence. In *NIPS*. 5827–5838.
[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12, Jul (2011), 2121–2159.
[9] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*. 2199–2207.
[10] Dan Geiger, Thomas Verma, and Judea Pearl. 1990. Identifying independence in Bayesian networks. *Networks* 20, 5 (1990), 507–534.
[11] Samuel Gershman and Noah D. Goodman. 2014. Amortized Inference in Probabilistic Reasoning. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society, CogSci 2014, Quebec City, Canada, July 23-26, 2014.*
[12] San Gultekin and John Paisley. 2014. A Collaborative Kalman Filter for Time-Evolving Dyadic Processes. In *ICDM*. 140–149.
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. WWW, 173–182.
[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).
[15] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. ACM, 505–514.
[16] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys*. ACM, 211–218.
[17] Ghassen Jerfel, Mehmet Basbug, and Barbara Engelhardt. 2017. Dynamic Collaborative Filtering With Compound Poisson Factorization. In *AISTATS*. 738–747.

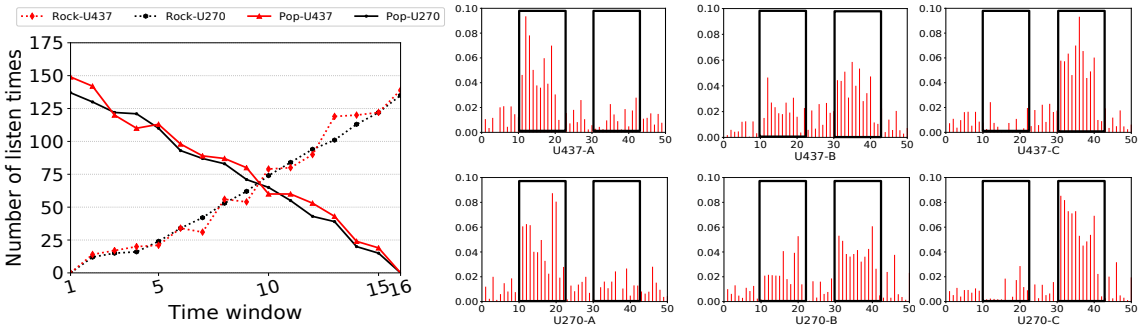

Figure 6: Left: the number of listen times that two users listened to the two 'rock' and 'pop' tracks with 16 time windows. The right: the inferred probability of each topic of long-term preferences $p = f(u)$ at time windows A, B and C, respectively.

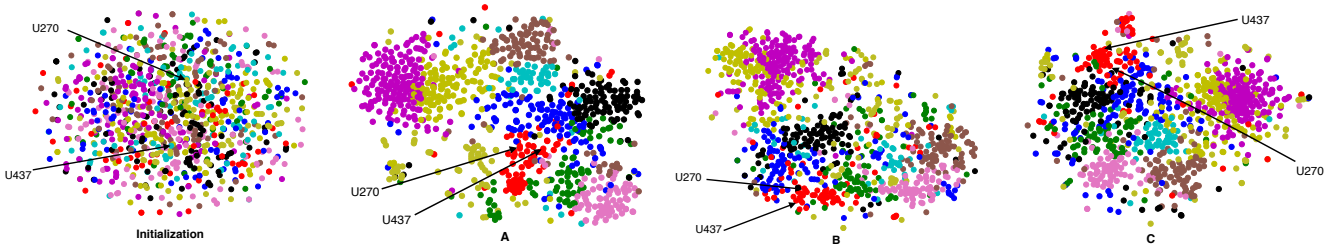

Figure 7: 2d t-SNE visualizations of inferred user long-term preferences at different time windows A , B, and C.

[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR* (2013).

[19] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. ACM, 447–456.

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

[21] Rahul G. Krishnan, Uri Shalit, and David Sontag. 2015. Deep Kalman Filters. (2015).

[22] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. ACM, 1419–1428.

[23] Shangsong Liang. 2018. Dynamic user profiling for streams of short texts. In *AAAI*.

[24] Shangsong Liang. 2019. Collaborative, dynamic and diversified user profiling. In *AAAI*.

[25] Shangsong Liang. 2019. Unsupervised semantic generative adversarial networks for expert retrieval. In *WWW*. 1039–1050.

[26] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. 2019. Collaboratively Tracking Interests for User Clustering in Streams of Short Texts. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 257–272.

[27] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic embeddings for user profiling in twitter. In *KDD*. 1764–1773.

[28] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.

[29] Yong Liu, Lifan Zhao, Guimei Liu, Xinyan Lu, Peng Gao, Xiao-Li Li, and Zhihui Jin. 2018. Dynamic Bayesian Logistic Matrix Factorization for Recommendation with Implicit Feedback. In *IJCAI*. 3463–3469.

[30] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, Nov (2008), 2579–2605.

[31] Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering Discrete Latent Topics with Neural Variational Inference. In *ICML*. 2410–2419.

[32] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*. 1257–1264.

[33] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (July 2018).

[34] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*. ACM, 130–137.

[35] Rajesh Ranganath, Sean Gerrish, and David M. Blei. 2014. Black Box Variational Inference. In *AISTATS*. 814–822.

[36] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.

[37] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*. 1278–1286.

[38] Akash Srivastava and Charles A. Sutton. 2017. Autoencoding Variational Inference For Topic Models. In *ICLR*.

[39] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. ACM, 565–573.

[40] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *SIGIR*. 403–412.

[41] Greg Welch and Gary Bishop. 1995. *An Introduction to the Kalman Filter*. Technical Report. Chapel Hill, NC, USA.

[42] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. ACM, 495–503.

[43] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Hierarchical Neural Variational Model for Personalized Sequential Recommendation. In *WWW*. ACM, 3377–3383.

[44] Teng Xiao, Shangsong Liang, Weizhou Shen, and Zaiqiao Meng. 2019. Bayesian Deep Collaborative Matrix Factorization. In *AAAI*.

[45] Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic Bayesian Metric Learning for Personalized Product Search. In *CIKM*. ACM.

[46] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM*. SIAM, 211–222.

[47] Christopher C Yang. 2010. Search engines information retrieval in practice. *Journal of the American Society for Information Science and Technology* 61, 2 (2010), 430–430.

[48] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS*. 847–855.

[49] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. 2014. Latent Factor Transition for Dynamic Collaborative Filtering. In *SIAM*. 452–460.