Coupled Variational Recurrent Collaborative Filtering

Qingquan Song\(^1\), Shiyu Chang\(^2\), Xia Hu\(^1\)
\(^1\)Department of Computer Science and Engineering, Texas A&M University
\(^2\)MIT-IBM Watson AI Lab, IBM Research
{song_3134,xiahu}@tamu.edu, shiyu.chang@ibm.com

ABSTRACT

We focus on the problem of streaming recommender system and explore novel collaborative filtering algorithms to handle the data dynamicity and complexity in a streaming manner. Although deep neural networks have demonstrated the effectiveness of recommendation tasks, it is lack of explorations on integrating probabilistic models and deep architectures under streaming recommendation settings. Conjoining the complementary advantages of probabilistic models and deep neural networks could enhance both model effectiveness and the understanding of inference uncertainties. To bridge the gap, in this paper, we propose a Coupled Variational Recurrent Collaborative Filtering (CVRCF) framework based on the idea of Deep Bayesian Learning to handle the streaming recommendation problem. The framework jointly combines stochastic processes and deep factorization models under a Bayesian paradigm to model the generation and evolution of users’ preferences and items’ popularities. To ensure efficient optimization and streaming update, we further propose a sequential variational inference algorithm based on a cross variational recurrent neural network structure. Experimental results on three benchmark datasets demonstrate that the proposed framework performs favorably against the state-of-the-art methods in terms of both temporal dependency modeling and predictive accuracy. The learned latent variables also provide visualized interpretations for the evolution of temporal dynamics.

KEYWORDS

collaborative filtering, streaming recommender system, matrix factorization, deep Bayesian learning

ACM Reference Format:

1 INTRODUCTION

With the explosive growth of online information, recommender systems have been pervasively used in real-world business services and widely studied in literature [31, 32]. Upon classical static settings, in real-world applications, data are often grown in a streaming fashion and evolving with time. For example, Snapchat users share over 400 million snaps [40] and Facebook users upload 300 million photos per day [39]. The ever-growing data volume along with rapidly evolved data properties puts the demand of time aware and online recommender systems, which could incorporate the temporal information to handle the data temporality and update in a streaming manner to alleviate the burden of data complexity.

Deep learning techniques have been widely conducted in exploiting temporal dynamics to improve the recommendation performance [2, 18, 50, 52]. Despite the prominence shown recently in deep recommender systems [13, 18, 43], deep frameworks also have their own limitations. One of the well-known facts is that deep recommender systems are usually deterministic approaches, which only output point estimations without taking the uncertainty into account. It significantly limits their power in modeling the randomness of the measurement noises [35] and providing predictions of the missing or unobserved interactions in recommender systems. As probabilistic approaches, especially Bayesian methods, provide solid mathematical tools for coping with the randomness and uncertainty, it motivates us to conduct streaming recommendations from the view of Deep Bayesian Learning (DBL) to conjoin the advantages of probabilistic models and deep learning models. Though some recent attempts have been made on integrating probabilistic approaches with deep autoencoder architecture for recommendation tasks [16, 24, 33], they are still underpinned the static recommendation setting, which allows them to be retrospective to all the historical data during the updates.

Simply applying DBL to streaming recommendations is a non-trivial task due to the following challenges. First, coordinating the temporal dynamics is difficult given the continuous-time discrete-event recommendation process along with the protean patterns on both user and item modes. A user’s preference on certain items may evolve rapidly, while on others maintaining a long-term fix. Second, the high velocity of streaming data requires an updatable model, which could expediently extract the prior knowledge from former time steps and effectively digest it for current predictions. Also, since the data occurrence is, in fact, continuous-valued, taking the continues time information into consideration could be potentially helpful for the knowledge distillation [2]. Third, the DBL frameworks are usually expensive in terms of both time and space complexities. Existing optimization algorithms often require a huge amount of computation to infer and update especially under streaming setting such as Sequential Monte Carlo, which is usually infeasible for large-scale recommendations.

To tackle the aforementioned challenges, in this paper, we propose to investigate the ways to conduct streaming recommendation by leveraging the advantages of both deep models and probabilistic processes. We stick to the factorization-based approaches due
to their popularity and superiority among all collaborative filtering techniques [17]. Specifically, we study: (1) How to model the streaming recommender system with an updatable probabilistic process? (2) How to incorporate deep architectures into the probabilistic framework? (3) How to efficiently learn and update the joint framework with streaming Bayesian inference? Through answering these three questions, we propose a Coupled Variational Recurrent Collaborative Filtering (CVRCF) framework. CVRCF incorporates deep architectures into the traditional factorization-based model and encodes temporal relationships with a coupled variational gated recurrent network, which is optimized through sequential variational inference. The main contributions are summarized as follows:

- Propose a novel streaming recommender system CVRCF, which incorporates deep models and the general probabilistic framework for streaming recommendations;
- Build up a linkage between probabilistic process and deep factorization-based model under a streaming setting with sequential variational inference leveraging a continues-time discrete-event cross RNN model;
- Empirically validate the effectiveness of CVRCF on different real-world datasets comparing with the state-of-the-art, explore the temporal drifting patterns learned from CVRCF, and analyze the model sensitivities.

2 PRELIMINARIES

Notations: Before discussing the proposed framework CVRCF for streaming recommendations, we first introduce the mathematical notations. We consider the streaming interactions as a continuous-time discrete-event process. Equipped with this viewpoint, we denote $T \in \mathbb{N}$ as the discrete time step and the inputs of a streaming recommender system can be denoted as a list of user-item interactions $\{x_{i,j}^T\}$ with their occurrence time $\{t_{ij}^T\}$, where $x_{i,j}^T$ denotes the interaction event of the $i^{th}$ user and the $j^{th}$ item occurred between time step $T - 1$ and $T$, $t_{ij}^T$ denotes the concrete time that $x_{i,j}^T$ occurs. The time interval between two consecutive time steps is called granularity, which does not need to be fixed in practice. All interactions arrived before the $T^{th}$ time step are denoted as $\{x_{i,j}^{\leq T}\}$ (or $\{x_{i,j}^{\leq T}\}$). Without loss of generality, interactions are regarded as ratings throughout this paper.

Problem Statement: Based on these notations, the streaming recommendation problem we studied in this paper is defined as: for any $T = 1, 2, \ldots$, given the sequence of historical user-item interactions $\{x_{i,j}^{\leq T-1}\}$, with the actual time information $\{t_{ij}^{\leq T-1}\}$, we aim at predicting the upcoming interactions $\{x_{i,j}^T\}$ in a streaming manner. The streaming manner here means that the model should be streamingly updatable. In another word, if we assume a model is achieved at $T = k - 1$, then at time $T = k$, the model should be able to update based only on the data acquired between time $T = k - 1$ and $T = k$, i.e., $\{x_{i,j}^T\}$.

3 COUPLED VARIATIONAL RECURRENT COLLABORATIVE FILTERING

The core of CVRCF is a dynamic probabilistic factor-based model that consists of four components. The first two formulate the user-item interactions and temporal dynamics, respectively. Each of them incorporates a probabilistic skeleton induced by deep architectures. The third component is a sequential variational inference algorithm, which provides an efficient optimization scheme for streaming updates. The last component allows us to generate rating predictions based on the up-to-date model.

3.1 Interaction Network

Factor-based models are widely adopted in recommendation modeling. They have shown a great success in multiple recommendation tasks [25]. Most of them follow the traditional matrix factorization setting, in which users and items are modeled as latent factors; and their interactions are defined as the linear combinations of these factors. However, such simple linear combinations are often insufficient to model complex user-item interactions [17]. Thus, we consider a deep probabilistic matrix factorization setting as follow

$$x_{i,j}^T | u_i^T, v_j^T, \sigma_{i,j}^2 \sim \mathcal{N}(f_1(u_i^T, v_j^T), f_2(u_i^T, v_j^T, \sigma_{i,j}^2)), \quad (i)$$

where both $f_1(\cdot)$ and $f_2(\cdot)$ are represented by deep neural networks. We represent the latent vectors of user $i$ and item $j$ at time step $T$ as $\{u_i^T\}$ and $\{v_j^T\}$, respectively. The rating $x_{i,j}^T$ is modeled as a Gaussian random variable whose location and scale values are the output of the deep networks. The environmental noise $\sigma_{i,j}^2$ could either be predefined as a hyperparameter [25] or jointly learned. It is worth pointing out that we assume the variance of $x_{i,j}^T$ depends on both the latent vectors and the environmental noise, which is slightly different from the conventional probabilistic setting [25].

3.2 Temporal Drifting Process

The temporal dynamics of a recommender system depend on the drifting of users’ preferences and item popularities [9, 30]. A user’s tastes for a certain type of items may change over time while the popularity of an item may also vary with time goes by. To capture the inherent dynamics, we intend to encode the drifting processes into user and item latent factors based on three hypotheses:

- We assume the latent factors of both user and item can be decomposed as the combination of a stationary term $(u_i^T)$ and a dynamic term $(\Delta u_i^T)$ [50]. The stationary factor captures the long-term preference, which varies slowly over
time. The dynamic factor encodes the short-term changes, which evolves rapidly. An illustrative example is shown in Figure 1, where a user’s dynamic factor evolves between two consecutive time steps, causing his preference drifted from $u^T_{i-1}$ to $u^T_i$. We assume the two factors are independent of each other for simplicity.

- The dynamic factors of a user or an item follows a Markov process [5]. The intuition of using a Markov process comes from the observation that the changing of a user’s current preference could be highly affected by his former preference.
- The changing of latent factors of a particular user $i$ (or item $j$) between two consecutive time steps $T - 1$ and $T$ depends on the time interval between the last events before these two time steps, which involves this user (or item), i.e., $\Delta u^T_{i,i} = r^T_{u,i} - r^{-1}_{u,i}$, where $r^T_{u,i}$ and $r^{-1}_{u,i}$ denote the actual time of the last two interactions of user $i$ before time step $T - 1$ and $T$, respectively. Intuitively, the longer the interval is, the larger the drifting may happen. $r^T_{u,i}$ is defined to be equal to $r^{-1}_{u,i}$ if no interactions happen between time step $T - 1$ and $T$.

Upon these hypotheses, we model the evolution of hidden topics of a user (or an item), via spatiotemporal Gaussian priors, which is mathematically formulated as follows:

$$
\begin{align*}
\begin{cases}
    u^T_i = u^T_i + \Delta u^T_i, \\
    u^T_j \sim N(0, \sigma^2_u I), \\
    \Delta u^T_i | \Delta u^{T - 1}_i \sim N(\mu_{u,i,T}, \Sigma_{u,i,T}).
\end{cases}
\end{align*}
$$

(2)

It is worth pointing out that only the users, which have interactions between time $T$ and $T - 1$, need to be considered here while factors of users who do not have interactions are assumed to be unchanged till their next interaction happens. We place the zero-mean spherical Gaussian prior on the stationary factors [25], where $\sigma_u$ denotes the scale hyperparameter. For dynamic factors, the kernel matrix $\Sigma_{u,i,T}$ is defined as a diagonal matrix here for simplicity, i.e., $\Sigma_{u,i,T} \triangleq \text{diag}(\sigma_u^2, I)$. Motivated by the recent advances in deep kernel learning, which combines the non-parametric flexibility of kernel approaches with the structural properties of deep architectures [49], we further define the kernel as an output of a deep neural network $f_k(\cdot)$ to enhance its generality, i.e., $\sigma^2_{u,i,T} = f_k(\Delta u^{T - 1}_i, \Delta u^T_{i,j})$.

Coping with the last two hypotheses, this spatiotemporal kernel takes $\Delta u^T_{i}$, which represents the user’s dynamic preference at last time step, as a spatial effect to decide the drifting uncertainty and it is stationary for temporal effect, which means $\Sigma_{u,i,T}$ depends on the time internal $\Delta u^T_{u,i}$ rather than the concrete time $r^T_{u,i}$ and $r^{-1}_{u,i}$. For a more unified representation, we can further define $p_{u,i} = f_k(\Delta u^{T - 1}_i, \Delta u^T_{i,j})$, where $f_k(\cdot)$ denotes a predefined deep neural network. The definition of the whole drifting prior obeys the Markov property for the discrete events on the continues timeline, which implies that the current state depends only on the former state. It is also applicable to employ other state dependency correlations and network structures. Similar prior with corresponding notations is defined for items.

### 3.3 Deep Sequential Variational Inference

The third component of the CVR model is the inference model. It composites the two former components with a sequential Bayesian skeleton and associates them with the last prediction component for streaming recommendations.

#### 3.3.1 Joint Distribution

The joint distribution of all observations up to time $T$ and the latent factors is defined as follows:

$$
\begin{align*}
p(x^{\leq T}, u^{\leq T}, v^{\leq T}) &= p(x^{\leq T}, u^{T}, v^{T}, \Delta u^{\leq T}, \Delta v^{\leq T}) \\
&= p(x^{\leq T}, \Delta u^{\leq T}, \Delta v^{\leq T}|u^{T}, v^{T})p(u^{T})p(v^{T}) \\
&= p(u^{T})p(v^{T}) \prod_{t \leq T} p(x^{t}|x^{<t}, \Delta u^{\leq t}, \Delta v^{<t}, u^{T}, v^{T}) \\
& \times p(\Delta u^{T}, \Delta v^{T}|x^{<t}, \Delta U^{\leq t}, \Delta V^{<t}, u^{T}, v^{T})
\end{align*}
$$

(3)

where $U$ and $V$ are the matrices of the latent factors for existing users and items.

Our goal is to infer the posterior distribution of latent factors for every $t$, i.e., $p(U^{t}, V^{t}|x^{\leq t}), \forall t \leq T$. However, it is intractable for direct inferences based on the current model assumptions. To overcome this challenge, existing works usually focus on two types of approaches - Sequential Monte Carlo methods (SMC) [11] and Variational Inference methods (VI) [3]. The traditional sequential Bayesian updating usually uses SMC methods (a.k.a., particle filtering) to deal with intractable target posterior distributions. Although this approach is very accurate when suitable proposal distributions and enough particle samples are presented, the sampling process is often too slow to apply to high dimensional and large-scale data [34]. On the other hands, the variational inference is much faster compared to SMC. However, the accuracy highly depends on the approximation distribution, especially in streaming settings [42]. Although there are hybrid models combine both algorithms together [15, 27], the computational complexity makes it prohibited for large-scale recommender systems. To trade-off the model scalability and accuracy, we consider the streaming variational inference framework [3] by leveraging deep neural networks as the variational approximator to obtain more flexible posteriors.

#### 3.3.2 Sequential Variational Inference Network

Before introducing the deep architectures, we first assume the latent factors can be partitioned into independent units followed by the traditional mean-field approximation:

$$
q(\Delta U^{\leq T}, \Delta V^{\leq T}|x^{\leq T}, u^{T}, v^{T}) = q(\Delta U^{\leq T}|x^{\leq T})q(\Delta V^{\leq T}|x^{\leq T})
$$

(4)

where $q$ denotes the approximated variational posterior. Further, each user (or item) is placed by a Gaussian variational posterior as follows:

$$
q(u^{T}_{i}|u^{T-1}_{i}, x^{<t}_{i}) = N(\mu_{u,i,t}, \Sigma_{u,i,t}), \forall 1 \leq t \leq T
$$

(5)

where $\Sigma_{u,i,t}$ is diagonal with the similar definition as the priors defined in Equ. (2). $x^{<t}_{i}$ denotes all the interactions related to user $i$ before time step $t$.

To infer the variational posterior, we propose a Coupled Variational Gated Recurrent Network structure (CVGRN) leveraging two variational Gated Recurrent Units (GRUs) for users and items, respectively. Figure 2(a) demonstrates the key idea of the proposed inference network. Blocks represent the inputs of two GRUs at
different time steps. \( q_{u,i}^t \) and \( q_{v,j}^t \) represent the approximated posterior distribution \( q(\Delta u_i^T | \Delta u_i^{t-1}, x_i^t) \) and \( q(\Delta v_j^T | \Delta v_j^{t-1}, x_j^t) \), which are inferred based on the GRUs output states \( h_{u,i}^{t-1} \) and \( h_{v,j}^{t-1} \) and the interactions elated to user \( i \) and item \( j \) between time step \( t - 1 \) and \( t \), i.e., \( \{ x_i^t \} \) and \( \{ x_j^t \} \). Specifically, assume a user and a movie interact with each other at time \( t \). The red and blue blocks denote the inputs of the user chain and item chain at time step \( t \), respectively, which are denoted as \( y_{u,i}^t \) and \( y_{v,j}^t \). These two inputs are constructed based on user \( i \)'s or item \( j \)'s interactions between time steps \( t - 1 \) and \( t \), respectively. For example, \( y_{u,i}^t \) is defined as \( y_{u,i}^t = [W_u \cdot x_{u,i}^t, \log(\Delta r_{u,i}^t), 1_{u,\text{new}}] \), where \( x_{u,i}^t \) denotes a sparse vector consisting of the ratings \( \{ x_i^t \} \) given by user \( i \) in time interval \( \Delta r_{u,i} \). \( W_u \) is an embedding matrix, which is employed to reduce the length of GRUs inputs for alleviating intermediate data explosion. \( 1_{u,\text{new}} \) indicates whether a user is a new user or not [50]. The log interval \( \log(\Delta r_{u,i}^t) \) is concatenated into the inputs to encode continues-time information [2]. Inferring \( q_{u,i}^t \) is equivalent to inferring \( \mu_{u,i}^t, \Sigma_{u,i}^t \) in Eq. (5), which are calculated as:

\[
[\mu_{u,i}^t, \Sigma_{u,i}^t] = [f_3(h_{u,i}^{t-1}, \Delta r_{u,i}^t), f_3(h_{u,i}^{t-1}, \Delta r_{u,i}^t)]
\]

To further encode the temporal information, we exponentially decay the latent state variables at each time step [26] as \( h_{u,i}^t \leftarrow h_{u,i}^{t-1} \cdot e^{-\frac{\lambda T}{T}} \), where \( \lambda \) is a predefined decay rate.

3.3.3 Objective Function. Considering RNN as a graphical model, we leverage the conditionally independency between current latent state and future inputs, and have \( h^T \perp \! \! \! \! \perp x^T | h^{T-1}, x^t \). Then Eq. (4) could be written as:

\[
q(\Delta U^T, \Delta V^T | x^T, U^t, V^t) = \prod_{t \leq T} q(\Delta U_i^T, \Delta U_j^T | x^T, U^t, V^t)\]

To obtain the objective function, we try to follow the traditional variational autoencoder to derive a variant variational lower bound. We start from the joint log likelihood and drive the objective function as follows:

\[
\log p(x^T, U^t, V^t) = \log p(x^T | U^t, V^t) + \log p(U^t) + \log p(V^t)
\]

\[
= \int \log p(x^T, \Delta U^T, \Delta V^T | U^t, V^t) d\Delta U^T d\Delta V^T + \log p(U^t) + \log p(V^t)
\]

\[
\geq \int q(\Delta U^T, \Delta V^T | x^T) \log p(x^T, \Delta U^T, \Delta V^T | U^t, V^t) d\Delta U^T d\Delta V^T
\]

\[
+ \log p(U^t) + \log p(V^t)
\]

\[
= \sum_{t \leq T} \left\{ E_q[\log p(x^T, U^t, V^t) | q(\Delta U_i^T, \Delta V_j^T)] + \log p(x^T | x^T, U^t, V^t) \right\}
\]

\[
- KL(q(\Delta U_i^T | x^T, U^t, V^t) || p(\Delta U_i^T | U^t, V^t))
\]

\[
- KL(q(\Delta V_j^T | x^T, U^t, V^t) || p(\Delta V_j^T | V^t))
\]

\[
+ \log p(U^t) + \log p(V^t)
\]

To further simply the expression, we denote the probabilities \( q(\Delta U_i^T | x^T, U^t, V^t) \), \( q(\Delta V_j^T | x^T, U^t, V^t) \), \( p(\Delta U_i^T | U^t, V^t) \), \( p(\Delta V_j^T | V^t) \), and \( p(x^T | x^T, U^t, V^t) \), as \( q_U^t, q_V^t, p_U^t, p_V^t \), and \( p_x^t \), respectively. Based on the former definitions, the objective function is defined as a timestep-wise variational lower bound as follows:

\[
\mathcal{L} = \sum_{t \leq T} \left\{ E_q[\log p_x^t] - KL(q_U^t || p_U^t) - KL(q_V^t || p_V^t) \right\}
\]

\[
+ \log p(U^t) + \log p(V^t)
\]

It is worth pointing out that the expectation term is calculated based on sampling, i.e., \( E_q[q] \approx \frac{1}{L} \sum_{l=1}^{L} \log p(x^T | x^T, U^{T-l}, V^{T-l}) \), where \( L \) is the number of samples we wish to use to estimate the
quantity. We specifically set $L = 1$ for every iteration in the imple-
mentation following the setting in conventional Variational Auto-
Encoder [21] and adopt the reparameterization trick for feasible
optimization.

As the rating sequence of each user or item could be infinite long
under the streaming setting, which makes it infeasible to feed the
whole sequences into the RNNs, this step-wise objective function
allows us to truncate the sequences into multiple segmentations for
a streaming inference. In another words, assume $q_{u,1}^T, q_{v,1}^T, p_{u,1}^T, p_{v,1}^T, U^s$ and $V^s$ are achieved at time step $T$, they could be treated as the prior
distribution of the latent variables at time step $T + 1$ and updated based
on the new interactions $[x_{ij}^s]$, the CVRCF framework, and the
following step-wise objective function:

$$
\mathcal{L} = \mathbb{E}_{q_{u,1}^T, q_{v,1}^T}[\log p_{x}(x_{ij}^T+1) - KL(q_{u,1}^T||p_{u,1}^T) + KL(q_{v,1}^T||p_{v,1}^T)] + \log p(U^s) + \log p(V^s).
$$

(9)

It is worth pointing that as stated in Section 3.2, we assume the
stationary factors $U^s$ and $V^s$ represent long-term users’ preferences
and item popularities. Thus, they should also be updated at each
time-step. However, they remain the same between two consecutive
time steps while the dynamic factors keep evolving.

3.4 Prediction Network

The prediction model is based on the generation model described
in Figure 2(b). At any testing time between time steps $T - 1$ and $T$,
to predict a specific ratings of a user $i$ to an item $j$, we first calculate the expectations of the current latent representations $u_i^T$ and $v_j^T$
based on the prior distributions $p_{u,i}^T$ and $p_{v,j}^T$, and the stationary
factors $u_i^s$ and $v_j^s$. The ratings is then predicted based on the dis-
tribution parameterized by the interaction network in Equ. (1), i.e.,
$\mathbb{E}(x_{ij}^T) = f_1(\mathbb{E}(u_i^T), \mathbb{E}(v_j^T))$. Similarly, the variance could also be
predicted as: $\mathbb{V}(x_{ij}^T) = f_2(\mathbb{E}(u_i^T), \mathbb{E}(v_j^T), \sigma_{i,j,T}^2)$. $\sigma_{i,j,T}^2$ is assumed
to be learnable as a function of the hidden states $h^{T-1}_{u,i}$ and $h^{T-1}_{v,j}$ in
our implementation.

4 EXPERIMENTS

In this section, we empirically evaluate the performance of CVRCF
framework by analyzing three major aspects. Q1: What are the
general performance of CVRCF compared with the other baselines?
Q2: What are the temporal drifting dynamics of users and items we
could learned? Q3: What are the sensitivities of the model to the
key hyperparameters? The code of CVRCF is available at GitHub:
https://github.com/song3134/CVRCF.

<table>
<thead>
<tr>
<th>Method</th>
<th>Categories</th>
<th>Streaming</th>
<th>Temporal</th>
<th>Probabilistic</th>
<th>Deep</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>time-SVD++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sD-PMF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sRRN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sRec</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CVRCF (proposed)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

### 4.1 Datasets

Three widely-adopted benchmark datasets shown in Figure 1 are
employed in our experiments. Detailed statistics of them are elabo-
rated as follows:

- **MovieTweetings (MT)** [10]: It is a benchmark dataset con-
sisting of movies ratings that were contained in well-structured
tweets on Twitter. It contains 696,531 ratings (0-10) provided by
53, 275 users to 30, 686 movies. All ratings are time-
associated spanning from 02/28/2013 to 04/07/2018. The
granularity is defined as four weeks.

- **MovieLens-10M (ML-10M)** [14]: It contains ten million
ratings to 10,681 movies by 71,567 users spanning from
1995 to 2009. The granularity is defined as four weeks.

- **Netflix** [28]: The Netflix challenge dataset consists of 100
million ratings by 480,189 users to 17,700 movies from 1999
to 2006. The granularity is defined as two weeks.

### 4.2 Baselines

As our main focus is factorization-based approaches, five repre-
sentative factorization-based baseline algorithms, including two
batch algorithms and three streaming algorithms are selected for
comparison from different perspectives shown in Table 2. Brief
descriptions of these methods are elaborated as follows:

- **PMF** [25]: Probabilistic Matrix Factorization is a conven-
tional recommendation algorithm, which does not consider
temporal information.

- **TimeSVD++** [22]: The temporal-envoled variation of the
classical static factor-based algorithm SVD++. We implement
it with Graphchi [23] C++ package.

- **sD-PMF**: A streaming version of the PMF model combined
with the deep interaction network, which is employed in
the CVRCF Framework. This model is used to test the ef-
fectiveness of the dynamic factors optimized with the RNN
structure in CVRCF.

- **sRec** [5]: Streaming Recommender System is the state-of-
the-art shallow dynamic recommendation model. It is a prob-
abilistic factor-based model optimized with a recursive mean-
field approximation.

- **sRRN** [50]: A streaming variation of Recurrent Recommender
Network (RRN), which is a state-of-the-art deep heuristic
streaming recommendation model.
### 4.3 Experimental Setup

For each dataset, we segment the data along timeline into three parts with ratios $4 : 1 : 5$ serving as training, validation, and testing sets, respectively.

#### 4.3.1 Training Settings.
During the training phase, the training and validation sets serve as the historical datasets to decide the best hyperparameters for all methods. As each user or movie may have too many ratings, to reduce and memory and protect the feasible use of GRU structures, we truncate the training sequences along the timeline into batches for the user and movie chain, respectively. This will affect the RNN effectiveness to some extent, but by varying the number of training epoch, it does not have an obvious influence on the experimental results during our experiments. Moreover, to protect the stationary factor get faster trained, in each epoch, every truncated batch is processed with multiple iterations. The number of this iteration hyperparameter used in the training phase is set based on validation and will be further analyzed in hyperparameter analysis section.

#### 4.3.2 Testing Settings.
During the testing phase, at each time step $t$, the testing is first done to get the prediction of the upcoming ratings $\{x_{t+1}\}$, and then these ratings are assumed to arrive and be used to update the models. Different from dynamic methods, at each update, the static methods are reconstructed from scratch using all the previously arrived testing ratings including the training ratings, while the streaming models only employ the current-step arrived ratings for the current update. Based on this setting, no later data is used to predict any former data and no temporal overlapping is existed between each pair of testing intervals. Besides, for fair comparisons, at each testing step, only ratings for existing users and items are used for testing since some baselines (e.g., PMF and time-SVD++) cannot explicitly cope with new users and items. All the experimental results are the arithmetic average of ten different times runs to ensure the reliability. The performance is evaluated via the root mean square error (RMSE).

#### 4.3.3 Parameter Setting.
Settings of the hyperparameters for all the baselines follow the original papers, which result in their best performance. Hyperparameters in all the methods are selected based on cross-validation using the training and validation sets. For the static baselines PMF and time-SVD++, all of their regularization parameters are chosen over $\{10^{-4}, 10^{-3}, \ldots, 10^2\}$ and the sizes of their latent factors are chosen over $\{20, 40, 60, 80, 100\}$. For streaming methods, the size of the stationary factors for sRRN and CVRCF are chosen to be 20 for all the datasets. The stationary factors for sD-PMF is chosen over $\{20, 40, 60, 80, 100\}$. The size of the dynamic factors of CVRCF is chosen to be 40 including the sizes of both mean and variance parameters. The size of the dynamic factors and the length of the RNN inputs for sRRN is chosen to be the same as CVRCF for fair comparisons. The size of the latent states ($h_u$ & $h_v$) of CVRCF is set to be 20 which is half of the length we used in sRRN. The exponential decay factors are set to be 1 week and 4 weeks for the user RNN and movie RNN, respectively. In the training phase, the truncation hyperparameters of all the RNN-based models are set to be 20, 20, and 10 weeks for the three datasets, respectively, to alleviate the intermediate data explosion.

### 4.4 General Evaluation Results

We first analyze the general performance of CVRCF model by comparing it with different categories of baselines based on the RMSE.

---

#### Table 3: The RMSE results on the three datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Methods</th>
<th>MT</th>
<th>ML-10M</th>
<th>Netflix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>PMF</td>
<td>1.5723</td>
<td>0.8202</td>
<td>0.9421</td>
</tr>
<tr>
<td></td>
<td>time-SVD++</td>
<td>1.4630</td>
<td>0.7985</td>
<td>0.9311</td>
</tr>
<tr>
<td>Streaming</td>
<td>sD-PMF</td>
<td>1.6170</td>
<td>0.9017</td>
<td>0.9992</td>
</tr>
<tr>
<td></td>
<td>sRRN</td>
<td>1.5646</td>
<td>0.8003</td>
<td>0.9236</td>
</tr>
<tr>
<td></td>
<td>sRec</td>
<td>1.4831</td>
<td>0.8121</td>
<td>0.9288</td>
</tr>
<tr>
<td></td>
<td>CVRCF</td>
<td>1.4567</td>
<td>0.7831</td>
<td>0.9050</td>
</tr>
</tbody>
</table>

---

#### Figure 3: Testing RMSE changing curve of four representative methods on ML-10M and Netflix datasets.

(a) Testing RMSE changing curve on MovieLens-10M dataset.

(b) Testing RMSE changing curve on netflix dataset.
results shown in Table 3 and Figure 3. From Table 3, three conclusions could be drawn as follows. First, CVRCF outperforms all baselines on all datasets. Although time-SVD++ could achieve comparable performance on MT and ML-10M dataset, it has to be reconstructed from scratch using all of the historical data at each update. Second, CVRCF highly outperforms sD-PMF, which confirms the effectiveness of the dynamic factors employed in CVRCF for capturing the temporal relationships during the streaming process. Third, comparing with shallow probabilistic model sRec, CVRCF displays prominent improvement, which demonstrate the effectiveness of deep architectures in modeling complex drifting interactions.

To further analyze the time-varying pattern of each method and their performance consistency on different datasets, we display the RMSE changing curves of the four representative methods on two larger datasets ML-10M and Netflix in Figure 3. From the figure, we could observe that on each dataset the performance of all methods shows similar varying patterns and starting form the first testing step, CVRCF consistently achieves the best performance across two datasets with the evolving of the system. Since MovieLens-10M has the longest testing timeline among all three testing datasets, Figure 3(a) illustrates that CVRCF has stable effectiveness on the dataset with strong temporal relationships in long-term evaluation. By comparison, Netflix is a much larger dataset in terms of users and interactions. Results in Figure 3(b) confirms the superiority of the proposed method on large-scale datasets. Finally, as sRRN could be treated as an ablation method of CVRCF without the probabilistic component, the relative improvement of the proposed method on the general performance validates the effectiveness of combining probabilistic approach in capturing the prospective process of streaming data generation.

4.5 Evaluation of Temporal Dynamics
To analyze the temporal drifting dynamics learned from CVRCF, we visualize the learned latent factors including the location factors \( \mathbf{u}^T \) and \( \mathbf{v}^T \) and uncertainty factors \( \mathbf{\sigma}_{u,i,T} \) and \( \mathbf{\sigma}_{u,j,T} \). We conduct exploration on the ML-10M dataset and update the models every half a year during testing.

4.5.1 Drifting of the Location Factors. We first visualize the drifting of the average location factors \( \mathbf{u}^T \) and \( \mathbf{v}^T \) with heatmap shown in Figure 4(a). The X-axis denotes the index of the latent factors and the Y-axis denotes the timeline. Each factor is adjusted with centralization for joint visualization. From the figure, we could discover that the users’ preference factors change more smoothly than movies’ popularity factors, which display a block-wise changing patterns. As we update the model every half a year, the stationary factors of movies especially for the new movies are only updated or learned every half a year, which is consistent with the length of the blocks. Thus, the block-wise structure, which appears only on the movie factors, could be explained as: the movie drifting is more likely to be captured by the stationary factors, while the drifting pattern of the users is more likely to be captured by dynamic factors. Since the dynamic factors and stationary factors are defined to capture the short-term and long-term preference, respectively, the finding is also consistent with the fact that users preference usually change more frequently compared to movie popularities.

4.5.2 Drifting of the Uncertainty Factors. Figure 4(b) displays the drifting of the average uncertainty factors learned from CVRCF. Each column is first normalized with \( \ell\infty \)-norm. There are two major observations we could find from Figure 4(b). From an overall perspective, with the evolving of the system, the variances of the learned dynamic factors decrease. This is because the incremental ratings provide more information for each user and item, and reduce the uncertainties of the whole system during the testing phase. From the local perspective, at some time steps, the variance of the latent factors are sharply increased and then slowly decreased. This is because, at some time steps, users and movies increase are dramatically. The cold-start problem introduced by the incremental users and items may raise the uncertainties of the system within a short time but would be alleviated with time goes by. In other words, although new users and items are continually enrolled, the number of ratings related to them could be deficient at first and then increasing over time.

4.6 Hyperparameter Sensitivity Analysis
Finally, we study the sensitivity of CVRCF to different hyperparameters using the ML-10M dataset. We pick five hyperparameters, which are the most influential ones in our experiments, and analyze their effects by coupling some of them. These pairwise effects are displayed in Figure 5.

4.6.1 Training Epochs & Training Batch Iterations. We first analyze the pairwise effects of the training epoch and the training batch iterations. Figure 5(a) shows that these two parameters highly affects the learning process and may cause overfitting or underfitting when the product of them are too large or too small. With the number of training batch iteration increasing, less epoch should be adopted to protect the testing effectiveness. This may be because: since the stationary factors are outside the RNNs and have high
4.6.3 Granularities. Finally, we explore the effect of the granularities. We assume the two granularities defined for users and movies could be different for a more general treatment. From Figure 5(c), we can see that although different granularities do affect the results, their influences are shown to be very trivial based on the scale of the Z-axis. Moreover, user granularity seems to have larger effects than movie granularity and its optimal value is shown to be lower than movie granularity. This may illustrate that the users’ preferences are varying more frequently than the items’ popularities.

5 RELATED WORK

Streaming Recommender Systems. Beyond traditional static settings, streaming recommender systems have attracted widespread concerns in coping with the high data velocity and their naturally incremental properties [1, 5]. Different from static time-aware models [12, 19, 20, 22, 48], which only take account of temporal dynamics without updating in an streaming fashion, streaming recommender systems dynamically encode temporal information and generate response instantaneously [6, 7, 36, 38]. Some existing works focus on extending classical memory-based recommendation algorithms into online fashions to address the streaming challenges such as [4] and [41]. Besides memory-based methods, model-based methods [8, 9, 30] are becoming more and more popular in recent years, which conducts recommendation based on well-trained models rather than explicitly aggregating and prediction based on the similarity relationships. Díaz-Aviles et al. leverage the active learning strategy to sample and maintain a delicately designed reservoir, thus providing a pairwise matrix factorization approach for streaming recommendation. Chang et al. [5] exploit continuous Markov process (Brownian motion) to model the temporal drifting of users and items, which introduces a principled way to model data streams. Wang et al. [46] propose a streaming ranking-based framework based on Bayesian Personalized Ranking [29] to address the user interest drifting as well as system overload problem. Although many recent advances based on deep neural networks especially RNNs have been made to model streaming inputs and capture the complex temporal dynamics [2, 18, 50], most of them overlook the causality inherited in the data generation process, which is one of the main aspects considered in our framework via the deep Bayesian learning.

Deep Recommender Systems. Deep learning techniques have brought vast vitality and achieve dramatic improvement in recommender systems [52]. They have been adopted in various recommendation tasks as well as accommodating different data sources [13, 43, 50]. From the perspective of the general framework, deep recommender systems could be categorized into solely deep models, which conduct recommendations based only on deep frameworks [13, 17, 37]; and integration models, which integrate deep techniques with traditional recommender systems [43–45]. From the perspective of deep frameworks, these models could also be divided into: (1) single deep models, which are built upon single neural building blocks such as multi-layer perceptron [17], convolutional neural network [47], and recurrent neural network [50]; and (2) composite models, which are constructed with different deep learning techniques [51]. From the first viewpoint of deivision, the proposed framework could be categorized as an integration model, which combines and probabilistic recommender systems with deep learning models. It is also a hybrid deep models, which jointly incorporates RNN and MLP structures. The coupled variational inference structure also provides its’ uniqueness comparing to other streaming deep recommender systems.

Figure 5: Analysis of five key hyperparameters on ML-10M datasets.
6 CONCLUSION AND FUTURE WORK

In this paper, we focus on the recommendation problem under streaming setting and propose a deep streaming recommender system - CVRCF. CVRCF incorporates deep architectures into traditional factorization-based model and encodes the temporal relationship with Gaussian-Markov components. Standing upon the sequential variational inference, CVRCF is optimized leveraging a cross variational GRU network and could continually update under the streaming setting. By conducting experiments on various real-world benchmark datasets, we empirically validate the effectiveness of our proposed framework, explore the learned drifting patterns, and validate the stability of our framework. Future work will center on exploring different assumptions of stochastic processes of the dynamic factors and incorporate other deep learning structures, such as graph neural networks, into the proposed framework.

7 ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their helpful comments. This work is in, part, supported by DARPA under grant #W911NF-16-1-0565 and #FA8750-17-2-0116, and NSF under grant #IIS-1657196 and #IIS-1718840. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[12] Nan Du, Yichen Wang, Niao He, Jimei Sun, and Le Song. 2015. Time-sensitive recommendation from recurrent user activities. In NIPS.
[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In UAI.
[36] Qiangguan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-aspect streaming tensor completion. In KDD.