Streaming Recommender Systems

Shiyu Chang[1], Yang Zhang[1], Jiliang Tang[2],
Dawei Yin[3], Yi Chang[4], Mark A. Hasegawa-Johnson[1], Thomas S. Huang[1]
[1] Beckman Institute, University of Illinois at Urbana-Champaign, IL 61801.
[2] Computer Science and Engineering, Michigan State University. East Lansing, MI 48824.
[3] Data Science Lab, JD.com. Beijing, 100101, China.
[4] Search Technology Lab, Huawei Research America. Santa Clara, CA 95050
{chang87, yzhan143, jhasegaw, t-huang1}@illinois.edu,
tangjili@msu.edu, {yindawei, yichang}@acm.org

ABSTRACT

The increasing popularity of real-world recommender systems produces data continuously and rapidly, and it becomes more realistic to study recommender systems under streaming scenarios. Data streams present distinct properties such as temporally ordered, continuous and high-velocity, which poses tremendous challenges to traditional recommender systems. In this paper, we investigate the problem of recommendation with stream inputs. In particular, we provide a principled framework termed sREC, which provides explicit continuous-time random process models of the creation of users and topics, and of the evolution of their interests. A variational Bayesian approach called recursive meanfield approximation is proposed, which permits computationally efficient instantaneous on-line inference. Experimental results on several real-world datasets demonstrate the advantages of our sREC over other state-of-the-arts.

Keywords

Streaming, recommender system, online learning, continuous time, data stream.

1. INTRODUCTION

Recommender systems help to overcome information overload by providing personalized suggestions from a plethora of choices based on the historical data. The pervasive use of real-world recommender systems such as eBay and Netflix generate massive data at an unprecedented rate. For example, more than 10 million transactions are made per day in eBay[1] and Netflix gained more than three million subscribers from mid-March 2013 to April 2013[2]. Such data is temporally ordered, continuous, high-velocity and time varying, which determines the streaming nature of data in recommender systems. Hence it is more realistic to study recommender systems using a streaming data framework.

[1]`http://www.webretailer.com/articles/ebay-statistics.asp`
[2]`https://en.wikipedia.org/wiki/Netflix`

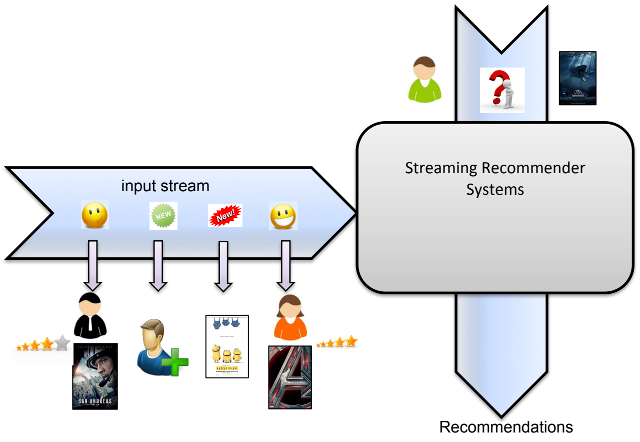


Figure 1: An Illustration of Streaming Recommender Systems. Online update and prediction are made based on three types of events: user feedback activities, new users, and new items.

In recent years, there is a large literature exploiting temporal information [9, 14, 28, 29] and it is evident that explicitly modeling temporal dynamics greatly improves the recommendation performance [5, 14, 27]. However, the vast majority of those systems does not consider the input data as streams. Recommendation under streaming settings needs to tackle the following challenges simultaneously:

Real-time updating: One inherent characteristic of data streams is their high velocity; hence the recommender system needs to update and response instantaneously in order to catch users' instant intention and demands.

Unknown size: New users or freshly posted items arrive continuously in data streams. For example, there were more than 21 million new products offered on the main Amazon USA websites from December 2013 to August 2014[3]. Hence the number of users and the size of recommendation lists are unknown in advance. Nevertheless, many existing algorithms[13] assume the availability of such information.

Concept shift: Data stream evolution leads to concept shifts, *e.g.*, a new product launch reduces the popularity of previous versions. Likewise, user preferences drift over time. The recommender system should have the ability to

[3]`http://export-x.com/2014/08/14/many-products-amazon-sell-2/`

capture such signals and timely adapt its recommendations accordingly.

This paper defines a streaming recommender system with real-time updates for a shifting concept pool of unknown size. An illustrative example of the proposed streaming recommender systems is demonstrated in figure 1. The input streams are modeled as three types of events: user feedback activities, new users, and new items. The system continuously updates its model to capture dynamics and pursue real-time recommendations. In particular, we tackle all three challenges simultaneously by a novel streaming recommender system framework (sRec). The major contributions are three-folds.

- We provide a principled way to handle data as streams for effective recommendations.
- We propose a novel recommender system sRec , which captures temporal dynamics under steaming settings and makes real-time recommendations.
- We conduct extensive experiments on various real-world datasets to understand the mechanism of the proposed framework.

2. MODEL FORMULATION

We model the streaming setting of recommender systems by assuming a set of continuously time-varying partially known user-item relationships as $\{R_{ij}^t\}$, where each R_{ij}^t represents the rating from user i to item j at a given time t, which is modeled as a random variable. We assume that time is continuous, $t \in \mathbb{R}_+$. In addition, $m^t \in \mathbb{Z}_+$ and $n^t \in \mathbb{Z}_+$ denote the number of users and items at time t. The numbers of users and items are dynamically changing over time.

Moreover, these time-dependent ratings are partially observed. We denote r_{ij}^t as the deterministic observed value of the random variable R_{ij}^t; $\mathcal{R}$ as a set that contains these observed rating values r_{ij}^t; and $\mathcal{R}^{0:t}$ as the subset of the observed ratings no later than t. Here, we want to emphasize that the user-item relationship at different times are considered different: $r_{ij}^t \in \mathcal{R}$ does **not** imply $r_{ij}^{t'} \in \mathcal{R}$, $\forall t' > t$ since we assume that the user-item relationships are inherently changing. One advantage is that we no longer assume that an item can be rated or adopted only once by a user. In contrast, users have the flexibility to edit or even delete ratings previously recorded.

2.1 Modeling User-Item Relationships

Ratings, discrete and ordered, are modeled by an ordered probit model [10], thus rating R_{ij}^t is a discretization of some hidden variable X_{ij}^t that depicts the "likeness" of user i to item j. Formally, we have $R_{ij}^t = k$, if $X_{ij}^t \in (\pi_k, \pi_{k+1}]$, where $k \in \{1, 2, ..., K\}$, and K is the total number of discretized levels. $\{\pi_k\}_{k=1}^{K+1}$ is a set of partition thresholds, which divide the real line $\mathbb{R}$ into segments where consecutive discrete scores are assigned. The boundaries are given by $\pi_1 = -\infty$, $\pi_{K+1} = \infty$. The intuition behind such a model is that it can easily handle both explicit and implicit feedback. For instance, Netflix uses scaled ratings ranging from 1 to 5, while Last.fm recommends music to target user based on their listening behaviors. In the first case, the rating type is explicit, and we can directly set $K = 5$. On the contrary, the probit model handles implicit feedback by treating each event (*e.g.* listen, download, click, *etc.*) at time t as a binary prediction. Then, a single threshold is sufficient.

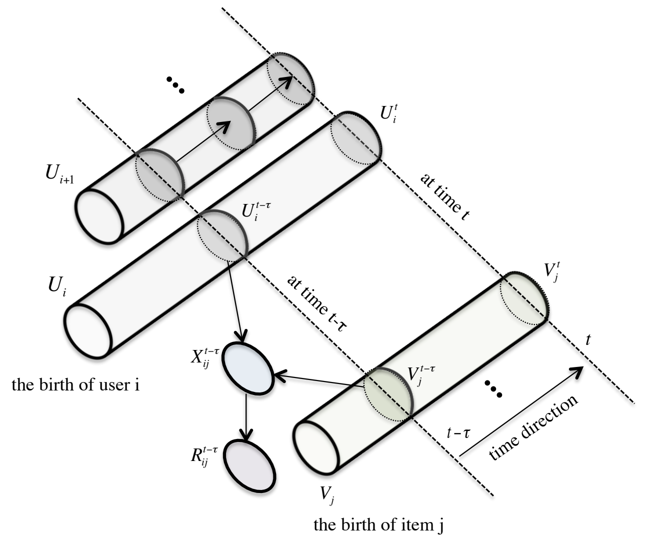


Figure 2: Graphical illustration of the model. Each tube structure denotes the continuous Markov process of each time-varying user/item topic, with different intersections denoting the user/item topics at different times. The time axis goes up in parallel to these tubes, with the top intersections corresponding to the current time. Different users/items are born at different times, so the tubes vary in length and position of their bottom intersections. A rating that user i assigns to item j at time t' (in the figure $t' = t - \tau$ as an example), $R_{ij}^{t'}$, depends on the hidden likeness $X_{ij}^{t'}$, which in turn depends on the hidden topics of user i and item j at time t'.

The hidden user-item "likeness" at t, X_{ij}^t, is modeled as

$$X_{ij}^t = (U_i^t)^T V_j^t + E_{ij}^t, \tag{1}$$

where U_i^t is a d-dimensional hidden topic vector for user i at t. Likewise, V_j^t is a d-dimensional hidden topic vector for item j at t. E_{ij}^t is a Gaussian perturbation with mean 0 and variance σ_E^2.

2.2 Temporal Dynamics

Temporal dynamics of U_i^t and V_j^t incorporate both hidden topic evolution and new user/item introduction. Specifically, the hidden topic vectors of existing users follow Brownian motion

$$U_i^t | U_i^{t-\tau} \sim \mathcal{N}\left(U_i^{t-\tau}, \sigma_U^2 \tau I\right), \tau > 0. \tag{2}$$

For new users, let $t_U(i)$ be the birth time of user i. We model the initial distribution (at the birth time), namely the distribution of $U_i^{t_U(i)}$, as the average of the posterior expectations of all existing user factors at $t_U(i)$ with Gaussian perturbations. We essentially assumes that the preference prior of a new user follows the general interests of others:

$$\begin{aligned} &U_i^{t_U(i)} | \{R_{ij}^t : t < t_U(i)\} \\ &\sim \mathcal{N}\left(\underset{k:t_U(k)<t_U(i)}{\text{mean}} \mathbb{E}\left(U_k^{t_U(i)} | \{R_{ij}^t : t < t_U(i)\}\right), \sigma_{U0}^2 I\right). \end{aligned} \tag{3}$$

where $\underset{k:t_U(k)<t_U(i)}{\text{mean}}$ denotes averaging over all existing users. Furthermore, equation (3) models the birth of all new users

but for the very first users at $t = 0$, for whom we assume a standard Gaussian prior as:

$$U_i^0 \sim \mathcal{N}(0, I), \forall i, t_U(i) = 0. \tag{4}$$

Temporal dynamics as well as the initial distributions for the item factors V_j^t are similar to those shown in equations (2)-(4) by replacing $U_i^{(\cdot)}$ to $V_j^{(\cdot)}$, $t_U(i)$ to $t_V(j)$, and σ_U^2 and σ_{U0}^2 to σ_V^2 and σ_{V0}^2, respectively.

2.3 Model Summary and Notation

To sum up, the proposed framework consists of the following hidden variables:

$$\mathcal{Z} = \left\{U_i^t, V_j^t, X_{ij}^t\right\} \cup \left\{R_{ij}^t : r_{ij}^t \notin \mathcal{R}\right\};$$

observed variables:$\left\{R_{ij}^t : r_{ij}^t \in \mathcal{R}\right\}$; and parameters to be estimated

$$\Theta = \left\{\sigma_E^2, \sigma_U^2, \sigma_V^2\right\}. \tag{5}$$

For better understanding, a graphical model is shown in figure 2. Moreover, before introducing the inference and training scheme for the model, we list the notations that we will use throughout the rest of the paper.

Time related notations and terminologies:

- An event: a new user/item introduced, or a rating assigned;
- t and t': : the current time and any arbitrary time, respectively;
- $t_U(i), t_V(j)$: the birth time of user i and item j, respectively;
- $\tau(t)$ and $\tau'(t)$: the time between current and the most recent event time preceding and proceeding, respectively, but not including, the reference time t;
- $\tau_U(i,t)$ and $\tau_V(j,t)$: the time between current and the most recent event time preceding, but not including, the reference time t that involves user i and item j, respectively;
- $\tau_U'(i,t)$ and $\tau_V'(j,t)$: the time between current and the up-coming event time from, but not including, time t for user i and item j;
- $\mathcal{T}$: the set of all event times;
- $\mathcal{T}_U(i), \mathcal{T}_V(j)$: the set of all event times that are related to user i and item j respectively.

Random variables and distributions:

- r_{ij}^t: the observed value of R_{ij}^t;
- $\mathcal{R}$: the set of values of all observed ratings r_{ij}^t;
- $\mathcal{R}^{0:t}$: the set of values of all observed ratings no later than t;
- $\mathcal{R}^t$: the set of values of all observed ratings at time t;
- $\mathcal{X}, \mathcal{X}^{0:t}, \mathcal{X}^t$: the set of random variables X_{ij}^t whose corresponding ratings have observed values in $\mathcal{R}$, $\mathcal{R}^{0:t}$, $\mathcal{R}^t$ respectively;
- $\bar{\mathcal{X}}^t$, the set of random variables X_{ij}^t at time t whose corresponding ratings are not observed;
- $p_{A|B}$: the probability density function of random variable A conditional on B, function argument omitted.

3. STREAMING RECOMMENDER SYSTEM

The proposed framework provides two components to address the following two problems as shown in figure 3:

- **Online prediction**: Based on observations UP TO the current time, how can we predict an unseen rating?

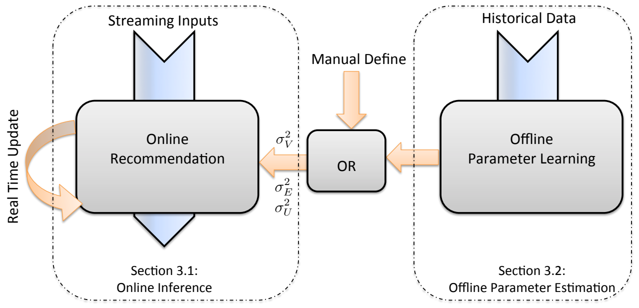


Figure 3: Overall framework illustration. The sREC framework consists of two components. The first component is online update and prediction, which provides a principled way to handle data streams. The second component provides a data driven method to estimate meaningful parameters from historical data.

- **Offline parameter estimation (learning)**: Given a subset of data, how can we estimate the parameters Θ defined in equation (5)?

For the first problem, we will apply posterior variational inference as shown in section 3.1, which leads to a very efficient streaming scheme: all the predictions are based on some posterior moments, which get updated only when a relevant event happens. Furthermore, the update only depends on the incoming event and the posterior moments at the previous event, but not on other historical events. For the second problem, we will apply variational EM algorithm, which will be derived in section 3.2.

3.1 Online Inference

Given past observations, for any unseen ratings *i.e.* $R_{ij}^t : r_{ij}^t \notin \mathcal{R}^{0:t}$, the inferred rating $\hat{R}_{ij}^t$ is given by posterior expectation as:

$$\hat{R}_{ij}^t = k, \text{ if } \mathbb{E}[X_{ij}^t|\mathcal{R}^{0:t}] \in (\pi_k, \pi_{k+1}]. \tag{6}$$

The predicted rating depends on the online posterior expectation of X_{ij}^t, which in turn depends on posterior distributions of other hidden variables.

3.1.1 Recursive Meanfield Approximation

According to equation (6), online inference needs to evaluate the posterior distribution $p_{\mathcal{X}^t, U^t, V^t|\mathcal{R}^{0:t}}, \forall t$, which can be decomposed into $p_{\mathcal{X}^t, U^t, V^t|\mathcal{R}^{0:t}} \cdot p_{\bar{\mathcal{X}}^t|U^t, V^t}$ by the Markov property. While the second term is readily defined by equation (1), it is challenging to calculate the first term due to the highly nonlinear model assumptions. Therefore, we propose a new method, called recursive meanfield approximation, to obtain an approximate posterior distribution $q_{\mathcal{X}^{t'}, U^{t'}, V^{t'}|\mathcal{R}^{0:t'}}$, which is defined alternatively and recursively by the following three equations.

First, $\forall t' \in \mathcal{T}$, $q_{\mathcal{X}^{t'}, U^{t'}, V^{t'}|\mathcal{R}^{0:t'}}$ is considered to be independent among $U^{t'}$, $V^{t'}$ and $X^{t'}$, and approximates the distribution induced by the Markov property

$$\begin{aligned} q_{\mathcal{X}^{t'}, U^{t'}, V^{t'}|\mathcal{R}^{0:t'}} &= q_{\mathcal{X}^{t'}|\mathcal{R}^{0:t'}} q_{U^{t'}|\mathcal{R}^{0:t'}} q_{V^{t'}|\mathcal{R}^{0:t'}} \\ &= \underset{q = q_{U^{t'}} q_{V^{t'}} q_{\mathcal{X}^{t'}}}{\operatorname{argmin}} \ \mathrm{KL}\left(q_{\mathcal{X}^{t'}, U^{t'}, V^{t'}|\mathcal{R}^{0:t'-\tau(t')}} p_{\mathcal{R}^{t'}|\mathcal{X}^{t'}} \| q\right). \end{aligned} \tag{7}$$

Second, $\forall t' \notin \mathcal{T}$, since there is no event at t', $\mathcal{R}^{0:t'} = \mathcal{R}^{0:t'-\tau(t')}$, thus:

$$q_{\mathcal{X}^{t'},U^{t'},V^{t'}|\mathcal{R}^{0:t'}} = q_{\mathcal{X}^{t'},U^{t'},V^{t'}|\mathcal{R}^{0:t'-\tau(t')}}. \tag{8}$$

Finally, both (7) and (8) depend on $q_{\mathcal{X}^{t'},U^{t'},V^{t'}|\mathcal{R}^{0:t'-\tau(t')}}$, which is defined by the Markov property:

$$\begin{aligned}&q_{\mathcal{X}^{t'},U^{t'},V^{t'}|\mathcal{R}^{0:t'-\tau(t')}}\\ =&\int q_{U^{t'-\tau(t')}|\mathcal{R}^{0:t'-\tau(t')}} q_{V^{t'-\tau(t')}|\mathcal{R}^{0:t'-\tau(t')}} p_{U^{t'}|U^{t'-\tau(t')}} \quad (9)\\ &\cdot p_{V^{t'}|V^{t'-\tau(t')}} p_{\mathcal{X}^{t'}|U^{t'},V^{t'}} p_{\mathcal{R}^{t'}|\mathcal{X}^{t'}} dU^{t'-\tau(t')} dV^{t'-\tau(t')}.\end{aligned}$$

The posterior distributions at non-event times are expressed by those at the most recent event times, hence the system only needs to keep track of the posterior distributions at the most recent event time, and updates them only when an event occurs, which makes the system efficient.

The following useful facts are stated here without proof.

Theorem 3.1. *For* **all** *$t' \in \mathcal{T}$, under recursive meanfield approximate distribution $q_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:t'}}$ and $q_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:t'-\tau(t')}}$*

- *$U^{t'}$ and $V^{t'}$ follow multivariate normal distribution.*
- *All individual user topics $U_i^{t'}$ and item topics $V_j^{t'}$ are jointly independent.*

The following corollary can be derived from theorem 3.1.

Corollary 3.1.1. *$\forall t' \in \mathcal{T}$, for user i and item j that are not relevant to the current ratings, i.e. $r_{ij}^{t'} \notin \mathcal{R}^{t'}$.*

$$\begin{aligned}\mathbb{E}_q(U_i^{t'}|\mathcal{R}^{0:t'}) &= \mathbb{E}_q(U_i^{t'}|\mathcal{R}^{0:t'-\tau(t')}) = \mathbb{E}_q(U_i^{t'}|\mathcal{R}^{0:t'-\tau_U(i,t')}),\\ Cov_q(U_i^{t'}|\mathcal{R}^{0:t'}) &= Cov_q(U_i^{t'}|\mathcal{R}^{0:t'-\tau(t')}) + \sigma_U^2\tau(t')I\\ &= Cov_q(U_i^{t'}|\mathcal{R}^{0:t'-\tau_U(i,t')}) + \sigma_U^2\tau_U(t')I.\end{aligned} \tag{10}$$

and similarly for V_j.

Theorem 3.1 and corollary 3.1.1 essentially state that 1) to keep track of the posterior distributions, we only need to keep track of their first and second moments; 2) we can update only those U_i^t and V_j^t that are associated with events at time t and the rest are unaffected; and 3) the update does not depend on other users or items. The solution to the variational problem defined in (7) reveals the updating equations for U_i^t,V_j^t and X_{ij}^t, $\forall t \in \mathcal{T}$.

Update Equations for U_i^t and V_j^t at Event Times: For any U_i^t associated with an event at time t, the posterior moments are given as

$$\begin{aligned}&\mathrm{Cov}_q\left(U_i^t|\mathcal{R}^{0:t}\right) =\\ &\left[\sigma_E^{-2}\sum_{j:r_{ij}^t\in\mathcal{R}^t}\mathbb{E}_q(V_j^t(V_j^t)^T|\mathcal{R}^{0:t}) + \mathrm{Cov}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})^{-1}\right]^{-1},\\ &\mathbb{E}_q\left(U_i^t|\mathcal{R}^{0:t}\right) =\\ &\mathrm{Cov}_q\left(U_i^t|\mathcal{R}^{0:t}\right)\Big(\sigma_E^{-2}\sum_{j:r_{ij}^t\in\mathcal{R}^t}\mathbb{E}_q\left(V_j^t|\mathcal{R}^{0:t}\right)\mathbb{E}_q\left(X_{ij}^t|\mathcal{R}^{0:t}\right) +\\ &\mathrm{Cov}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})^{-1}\mathbb{E}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})\Big),\end{aligned} \tag{11}$$

where $\mathbb{E}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})$ and $\mathrm{Cov}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})$ are given by equations (10), (3), or (4), which correspond to existing users, newly born users, and very first users, respectively.

Algorithm 1: Online Posterior Moments Update at Event Time

```
input  : Current time $t$ (must be an event instance); the
         event type(s); currently assigned rating(s) $\mathcal{R}^t$; last
         updated posterior moments of $\{U_i^t\}$, $\{V_j^t\}$.
output: Updated posterior moments of $\{U_i^t\}$, $\{V_j^t\}$ and
         $\{X_{ij}^t\}$ that are associated with the events at
         current time.
for $i$ : user $i$ borns at time $t$ do
    Initialize $\mathbb{E}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)}), \mathrm{Cov}_q(U_i^t|\mathcal{R}^{0:t-\tau(t)})$
      according to equations (3) or (4);
end
for $j$ : item $j$ borns at $t$ do
    Initialize $\mathbb{E}_q(V_j^t|\mathcal{R}^{0:t-\tau(t)}), \mathrm{Cov}_q(V_j^t|\mathcal{R}^{0:t-\tau(t)})$ by
      symmetry to equations (3) or (4);
end
while convergence not yet reached do
    for $i$ : user $i$ rates at time $t$ do
        Update $\mathbb{E}_q(U_i^t|\mathcal{R}^{0:t})$, $\mathrm{Cov}_q(U_i^t|\mathcal{R}^{0:t})$ according to
          equation (11);
    end
    for $j$ : item $j$ is rated at time $t$ do
        Update $\mathbb{E}_q(V_j^t|\mathcal{R}^{0:t})$, $\mathrm{Cov}_q(V_j^t|\mathcal{R}^{0:t})$ by symmetry to
          equation (11);
    end
    for $(i,j)$: user $i$ rates item $j$ at time $t$ do
        Update $\mathbb{E}_q(X_{ij}^t|\mathcal{R}^{0:t})$ according to equation (14).
    end
end
```

The update for V_j^t is identical, except that U and V are interchanged, and subscripts i and j are interchanged.

Update Equations for X_{ij}^t at Event Times: On the other hand, $q_{X_{ij}^t|\mathcal{R}^{0:t}}$ is a truncated Gaussian distribution with the Gaussian mean

$$\mu_{ij}^t = \mathbb{E}_q\left(U_i^t|\mathcal{R}^{0:t}\right)^T\mathbb{E}_q\left(V_j^t|\mathcal{R}^{0:t}\right), \tag{12}$$

and variance σ_E^2. The truncation interval is between $(\pi_{r_{ij}^t}, \pi_{r_{ij}^t+1}]$. Define

$$e_{ij}^t = \frac{\pi_{r_{ij}^t} - \mu_{ij}^t}{\sigma_E}, \text{ and } f_{ij}^t = \frac{\pi_{r_{ij}^t+1} - \mu_{ij}^t}{\sigma_E}. \tag{13}$$

The expectation of this truncated Gaussian can be expressed as

$$\mathbb{E}_q(X_{ij}^t|\mathcal{R}^{0:t}) = \mu_{ij}^t + \sigma_E\frac{\phi(e_{ij}^t) - \phi(f_{ij}^t)}{\Phi(f_{ij}^t) - \Phi(e_{ij}^t)}, \tag{14}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the pdf and cdf of the standard normal distribution respectively.

Update Equation for X_{ij}^t at Non-event Times: With all the online posterior expectations derived, we are now able to obtain an approximation of $\mathbb{E}\left(X_{ij}^t|\mathcal{R}^{0:t}\right)$, which is essential for rating prediction as in (6):

$$\begin{aligned}&\mathbb{E}(X_{ij}^t|\mathcal{R}^{0:t}) \approx \mathbb{E}_q(X_{ij}^t|\mathcal{R}^{0:t}) = \mathbb{E}_q((U_i^t)^TV_j^t|\mathcal{R}^{0:t})\\ =&\mathbb{E}_q(U_i^{t-\tau_U(i,t)}|\mathcal{R}^{0:t-\tau_U(i,t)})^T\mathbb{E}_q(V_j^{t-\tau_V(j,t)}|\mathcal{R}^{0:t-\tau_V(j,t)}).\end{aligned} \tag{15}$$

The last equality is given by corollary 3.1.1.

3.1.2 Algorithm table and Order Complexity

To sum up, the algorithm of updating the posterior moments at event times is given in algorithm 1. The posterior moments of a user/item topic are updated only when there is an event associated with it. So the total number of updates is equal to the total number of events. In the update for each new user/item, there is a summation over all existing user/item topics of dimension d, which can be accelerated by setting a global sum of all the existing user/item topics. This global sum is updated only once for every event. In the update for each new rating, there is an inversion of a size-d matrix, which needs no more than $O\left(d^3\right)$ operations. Then the total computation complexity over the entire time $[0,T]$ is $O\left(I\left|\mathcal{R}^{0:T}\right|d^3 + \left(m^t + n^t + \left|\mathcal{R}^{0:T}\right|\right)d\right)$ where I is the number iterations to reach convergence for each time.

3.2 Offline Parameter Estimation

The online streaming prediction relies on the set of parameters Θ, which can be either manually predefined by domain expertise, or estimated from historical data. Define T as the end time of the historical data. This section applies expectation maximization (EM) to learn the model with incomplete observations by first defining the complete data as $(\mathcal{Z}'^{0:T}, \mathcal{R}^{0:T})$ where $\mathcal{Z}'^{0:T}$ is a subset of $\mathcal{Z}$ that are associated with events, or more concretely

$$\mathcal{Z}'^{0:T} = \{U_i^t : t \in \mathcal{T}_U(i)\} \cup \{V_j^t : t \in \mathcal{T}_V(j)\} \cup \{X_{ij}^t : r_{ij}^t \in \mathcal{R}\}.$$

3.2.1 M-step

The auxiliary function, or Q function, is given by

$$Q(\Theta|\hat{\Theta}^{(k)}) = \mathbb{E}_{\mathcal{Z}'^{0:T}|\mathcal{R}^{0:T}}\left(\log p_{\mathcal{Z}'^{0:T},\mathcal{R}^{0:T};\Theta}\right).$$

Update σ_E^2:
According the first order condition, we obtain the optimal estimation formula of σ_E^2 as

$$(\hat{\sigma}_E^2)^{(k+1)} = \frac{1}{|\mathcal{R}^{0:T}|}\sum_{i,j,t':r_{ij}^{t'}\in\mathcal{R}^{0:T}} \mathbb{E}_p\left[\left(X_{ij}^{t'} - U_i^{t'T}V_j^{t'}\right)^2 |\mathcal{R}^{0:T};\hat{\Theta}^{(k)}\right], \quad (16)$$

where $|\mathcal{R}^{0:T}|$ denotes the total number of ratings in the historical data from time 0 to T.

Update σ_U^2 and σ_V^2:
Similar to the σ_E^2 case, the first order condition yields

$$(\hat{\sigma}_U^2)^{(k+1)} = \frac{1}{C}\sum_{i,t'\in\mathcal{T}_U(i)\backslash t_U(i)} \frac{\mathbb{E}_p\left[\|U_i^{t'} - U_i^{t'-\tau_U(i,t')}\|^2|\mathcal{R}^{0:T};\hat{\Theta}^{(k)}\right]}{t' - \tau_U(i,t')}. \quad (17)$$

C is a constant defined as the total number of user rating events. Furthermore, the estimation formula for σ_V^2 is almost identical to σ_U^2, which is omitted.

3.2.2 E-step

The exact value of all offline posterior moments listed in equations (16) and (17) cannot be obtained due to the intractable inference of the posterior distributions

$$p_{U^{t'},U^{t'-\tau(t')}|\mathcal{R}^{0:T}},\; p_{V^{t'},V^{t'-\tau(t')}|\mathcal{R}^{0:T}},\; p_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:T}}.$$

However, we can utilize the result from the online inference to recursively approximate the distributions by first defining the approximated offline posterior distributions as

$$q_{U^{t'},U^{t'-\tau(t')}|\mathcal{R}^{0:T}},\; q_{V^{t'},V^{t'-\tau(t')}|\mathcal{R}^{0:T}},\; q_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:T}},$$

which are applied to the estimation of σ_U^2, σ_V^2 and σ_E^2 respectively.

First, $q_{U^{t'},U^{t'-\tau(t')}|\mathcal{R}^{0:T}}$ is defined recursively by Markov property

$$\begin{aligned} q_{U^{t'},U^{t'-\tau(t')}|\mathcal{R}^{0:T}} &= q_{U^{t'}|\mathcal{R}^{0:T}}q_{U^{t'-\tau(t')}|U^{t'},\mathcal{R}^{0:t'-\tau(t')}} \\ &= q_{U^{t'}|\mathcal{R}^{0:T}}\frac{q_{U^{t'-\tau(t')}|\mathcal{R}^{0:t'-\tau(t')}}p_{U^{t'}|U^{t'-\tau(t')}}}{\int q_{U^{t'-\tau(t')}|\mathcal{R}^{0:t'-\tau(t')}}p_{U^{t'}|U^{t'-\tau(t')}}dU^{t'}}, \end{aligned}$$

where the first term is given backward-recursively by marginalizing $q_{U^{t'+\tau'(t')},U^{t'}|\mathcal{R}^{0:T}}$; and $q_{U^{t'-\tau(t')}|\mathcal{R}^{0:t'-\tau(t')}}$ in the fraction is given by online inference. $q_{V^{t'},V^{t'-\tau(t')}|\mathcal{R}^{0:T}}$ is defined similarly and is omitted.

Second, $q_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:T}}$, similar to (7), assumes $U^{t'}$, $V^{t'}$, $\mathcal{X}^{t'}$ are independent

$$q_{\mathcal{X}^{t'},U^{t'},V^{t'}|\mathcal{R}^{0:T}} = q_{\mathcal{X}^{t'}|\mathcal{R}^{0:T}}q_{U^{t'}|\mathcal{R}^{0:T}}q_{V^{t'}|\mathcal{R}^{0:T}},$$

where $q_{U^{t'}|\mathcal{R}^{0:T}}$ and $q_{V^{t'}|\mathcal{R}^{0:T}}$ are already defined, and $q_{\mathcal{X}^{t'}|\mathcal{R}^{0:T}}$ is set to minimize the KL divergence to the true distribution.

$$q_{\mathcal{X}^{t'}|\mathcal{R}^{0:T}} = \underset{q}{\operatorname{argmin}}\ \mathrm{KL}\left(p_{U^{t'},V^{t'},\mathcal{X}^{t'}|\mathcal{R}^{0:T}}\|q\cdot q_{U^{t'}|\mathcal{R}^{0:T}}q_{V^{t'}|\mathcal{R}^{0:T}}\right).$$

Similar to the online case, these approximated distributions have good properties, which will be stated without proof.

COROLLARY 3.1.2. *Under the approximated offline distribution $q_{U^{t'},U^{t'-\tau(t')}|\mathcal{R}^{0:T}}$ and $q_{V^{t'}V^{t'-\tau(t')}|\mathcal{R}^{0:T}}$, all individual user topics $U_i^{t'}$ and item topics $V_j^{t'}$ are independent.*

Using the corollary 3.1.2, all the offline posterior expectations listed in equations (16) and (17) can be computed.

4. EXPERIMENTS

To assess the effectiveness of the proposed framework, we collect three real-world datasets and their detailed descriptions are as follows:

MovieLens Latest Small: It consists of 100,000 ratings to 8,570 movies by 706 users. In addition, all ratings are associated with timestamps ranging from the year 1996 to 2015. The ratings are given on a half star scale from 0.5 to 5. In abbreviate, we use *ML-latest* for the following subsections.

MovieLens-10M: The dataset contains 10 million movie ratings from 1995 to 2009. The rating scale and temporal information are similar to those of the *ML-latest* dataset. In total, there are 10,000 movies and 71,000 users. We use *ML-10M* to represent the dataset.

Netflix: The *Netflix* dataset contains 100 million movie ratings from 1999 to 2006 that are distributed across 17,7700 movies and 480,189 users. The rating scale is from 1 to 5. Furthermore, the temporal granularity is a day.

4.1 Baseline Methods

We compare our proposed framework sREC with several representative recommender systems including:

PMF [19]: Probabilistic Matrix Factorization is a classical recommendation algorithm that is widely used.

MDCR [2]: Multi-Dimensional Collaborative Recommendation is a tensor-based matrix factorization algorithm, which can potentially incorporate both the temporal and spatial information.

Time-SVD++ [14]: Time-variant version of the Netflix winning algorithm SVD++ [13]. In addition, the model can be evolved efficiently via an online updating scheme.

GPFM [20]: Gaussian Process Factorization Machines is a variant of the factorization machines [21], which is non-linear, probabilistic and time-aware.

In summary, PMF is a static model, and all the other three baselines leverage the temporal factor in different ways. We utilize the existing C++ implementations from GraphChi [15] for PMF and Time-SVD++, while the code for GPFM is also available online[4].

4.2 Experimental Settings

We divide all data into halves. We call the first half the "base training set", which is used for parameter estimation; and the remaining half is the "candidate set". The base training set can be considered as the historical data pool while the candidate set mimics the streaming inputs. Both the proposed method and other baseline algorithms utilize the based training set to determine the best hyper parameters. For instance, the latent dimensionality for each algorithms are determined independently using this base training set. For online prediction, the actual testing set is generated from the candidate set by randomly selecting a reference time first. Such a reference time can be considered as the "current time". Then, our task is to predict user-item ratings for the following week starting from the reference time. All the ratings prior to the reference time are used for the online inference. The sequential order of data streams ensures a prospective inference procedure: no future ratings can be used to predict the past behaviors. It is worth mentioning that, except our sRec model, other baselines cannot explicitly handle new user/item introduction during the testing phase. Therefore, for a fair comparison, all the testing ratings are from the existing users and items which have appeared in the training set.

We evaluate the performance via the root mean square error (RMSE), which is a widely used metric in recommendation. Furthermore, in order to ensure reliability, all experimental results were averaged over 10 different runs. In addition, to keep the temporal variability off the testing set, there is no temporal overlapping between any testing sets.

4.3 Experimental Results

The best performance of each method on all three datasets is reported in table 1. We observe that the proposed algorithm achieves the best performance across all three datasets. It is evident that explicitly modeling user/item dynamics significantly boosts the recommendation performance under the streaming setting. The second best algorithm is Time-SVD++, which is better than the other three baselines consistently. This is because, Time-SVD++ models temporal information in a more suitable way under the streaming settings. However, the inflexibility to the temporal granularity makes it insufficient to capture any volatile changes. On the other hand, the tensor-based method MDCR yields the worst performance. One potential reason is that the algorithm considers temporal information in a retrospective way, which is obviously inadequate to capture the prospective process of data generation. Furthermore, the PMF method does not utilize any time information, however, its performance is still better than MDCR. This result implies that inappropriate temporal modeling even hurts the performance. Last, GPFM suffers from the problem of scalability, thus it provides no result for the Netflix dataset.

[4]`https://github.com/trungngv/gpfm`

Table 1: Performance comparison in terms of RMSE. The best performance is highlighted in bold.

	ML-Latest	ML-10M	Netflix
PMF	0.7372	0.8814	0.8610
MDCR	0.7604	0.9349	0.9326
GPFM	0.7710	0.9114	NA
Time-SVD++	0.7141	0.8579	0.8446
sRec	**0.6780**	**0.7957**	**0.8093**

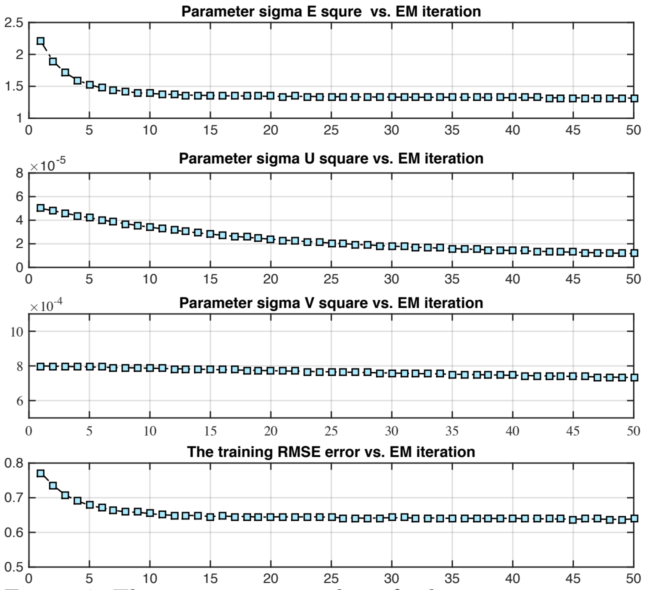


Figure 4: The convergence analysis for learning parameters on the *ML-Latest* dataset.

4.4 Parameter Understanding

In this subsection, we investigate the physical meanings of our learned parameters. The major parameters for the proposed method are σ_E^2, σ_U^2 and σ_V^2. According to our experimental settings, these parameters are learned from the so-called historical data. Note that, in this part, we will only show the results on the *ML-Latest* dataset. The results for other datasets are similar.

We first perform empirical convergence analysis for the proposed sRec method. Figure 4 demonstrates the convergence paths of all three parameters as well as the training error against EM iterations. As we can see, all the parameters converge, and the RMSE on the training set is saturated around the 10th iteration. The converged values of σ_E^2, σ_U^2 and σ_V^2 are 1.32, 1.17×10^{-5} and 7.33×10^{-4} respectively while the training RMSE is around 0.64. Next, we analyze the interpretations of the values of these parameters.

The first parameter can be understood as the impact of other unknown factors on ratings. The larger the value of σ_E^2, the more susceptible the rating is to variations in unknown factors, and hence the less predictable by sRec.

More interestingly, σ_U^2 and σ_V^2 are intuitively considered as the evolution speed of user/item taste based on our model

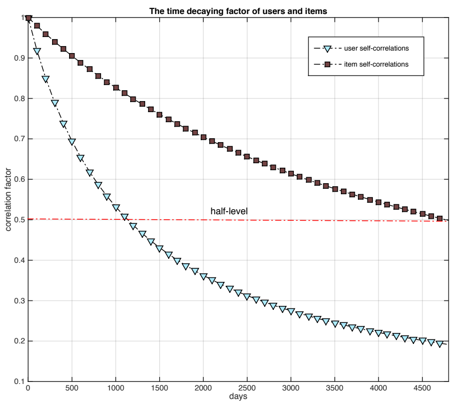


Figure 5: Correlation Decay with time. The half-time for user and item is 3.12 years and 13.06 years respectively.

assumptions in a global sense (across the whole population of the users/items). We relate the learned values of these two parameters to our intuitions in a quantitative way via the following posterior correlation function:

$$\mathrm{Corr}(U_i^{t'}, U_i^{t'+\delta_t}|\mathcal{R}^{1:t'})^2 = \frac{\mathrm{Tr}(\mathrm{Cov}(U_i^{t'}|\mathcal{R}^{1:t'})}{\mathrm{Tr}(\mathrm{Cov}(U_i^{t'}|\mathcal{R}^{1:t'}) + d\delta_t\sigma_U^2}. \quad (18)$$

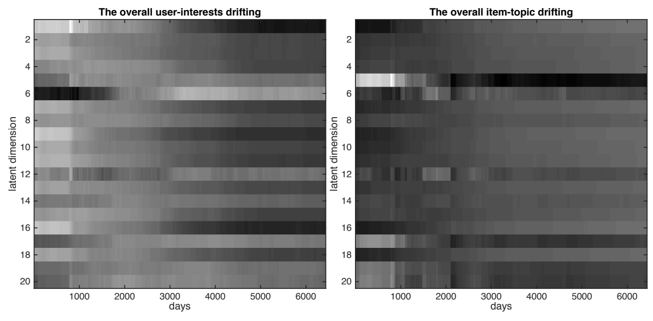
This quantity is a function of σ_U^2, which measures the user auto-correlation from a reference time t' to some future time $t' + \delta_t$. The range of such a measurement is from 0 to 1. The item correlation function is identical to equation (18) by replacing the corresponding U with V. We vary δ_t and plot the mean of these correlations for all users and items in figure 5. The reference time t' is equal to the last time instance in the training data, and σ_U^2 and σ_V^2 are obtained from the offline learning.

Furthermore, one critical value for the correlation measure is 0.5, which is usually referred as the "half-time". In figure 5, the half-time for both users and items are the intersections of the red dashed line to their own curves. The"half-time" for users is around 3.12 years; while the one for items is 13.06 years, which can be understood as: a rating provided by a user can only be trusted to reflect half his/her taste for 3.12 years. Although we indeed observe the change of item topics over time, the change is significantly slower than the user ones. This finding can be used to explain why many existing recommender systems only consider the dynamics of user topics while assuming fixed item topics.

4.5 Temporal Drifting

We demonstrate the proposed sRec can capture the temporal drifting phenomenon from another perspective. We recorded all the posterior expectations of user/item topics as a function of time. Figure 6 shows the evolution of latent factors with the *ML-Latest* dataset, which x-axis denotes time and y-axis is the latent dimension. The intensity of the user latent representation reflects the user likeness to a specific hidden topic. The darker the color in figure 6a, the less emphases users lay on the certain topic. On the other hand, the item latent vectors indicate the assignment of items to these latent topics. The observation is consistent with our experimental results in section 4.4: the user tastes change more remarkably compared to item topics.

Furthermore, several topics evolve much more drastically compared to others. Notable ones include the 1st, 6th, 12th



(a) User (b) Item

Figure 6: The averaged latent topics for both users (6a) and items (6b) at different time.

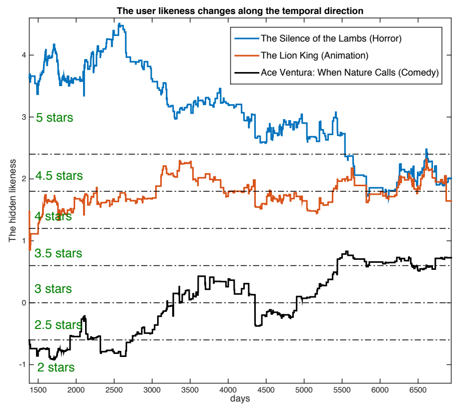


Figure 7: An example of user interest evolving over time through predicted ratings for some representative movies.

and 16th user topics as well as the 5th item topic. To understand the reason, we provide an example using the 6th user topic, which has a volatile shift in interests around the day 1,500 (year 2001). Table 2 shows the top movies that have strongest responses at the 6th latent topics. Most of the movies are the popular commercial type with the majority of their ratings provided after day 1500. Moreover, the average rating for the most of these movies is substantially higher than that of the whole dataset, which is 3.49. Another indicatory event is the movie "The Lord of Rings" that was also released in that year. This might potentially correlate to user taste changes and affect the latent representation.

We next present a qualitative result to demonstrate the advantage of our sRec model in tracking user instant preference by plotting the predicted "likeness" (posterior expectation of X_{ij}) for one of the most active users from the data to a set of movies as a function of time. In figure 7, the preference of an individual user continuously evolves on those three selected movies. An initial strong interest in "The Silence of the Lamb" decreases slightly with time. In contrast, the comedy movie "Ace Ventura" becomes more and more preferable. The last movie "The Lion King" is considered as a favorable movie by the user across the entire time.

5. RELATED WORK

In this section, we focus on two types of related work: the first one is about online updating for recommender systems; and the other one is about time-aware recommendations.

Table 2: The statistics of the top movies that contain strong responses at their 6th latent dimension.

Topic 6	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6
Movie name	Start War V	There's Something About Mary	Alien	The Lord of Rings I	Star Wars IV	Star War VI
Genre	Action, Sci-Fi Adventure	Comedy, Romance	Horror, Sci-Fi	Adventure, Fantasy	Action, Sci-Fi Adventure	Action, Sci-Fi Adventure
Release time	1980	1998	1979	2001	1977	1983
Percentage of ratings after day 1500	75.43%	88.39%	78.87%	100%	69.15%	70.46%
Average ratings after day 1500	4.17	3.64	4.02	4.04	4.14	3.99

5.1 Online Recommender System

Most previous works focus on traditional batch regression problem [20, 24, 25]. However, the recommender system by nature is an incremental process. That is, users' feedback are performed in a sequence, and their interests are also dynamically evolving. To capture the system evolution, once a user feedback is performed, the recommender system should be able to get updated. Authors in [1] proposed a fast online bilinear factor model to learn item-specific factors through online regression, where each item can perform independent updates. Hence, this procedure is fast, scalable and easily parallelizable. [22] introduced regularized kernel matrix factorization method where kernels provide a flexible way to model nonlinear interactions and an online-update scheme. Das *et al.* [7] addressed the large data and dynamic content problem in recommender systems, and proposed online models to generate personalized recommendations for Google News users. Diaz *et al.* [8] presented Stream Ranking Matrix Factorization, which uses a pairwise approach to matrix factorization in order to optimize the personalized ranking of topics and follows a selective sampling strategy to perform incremental model updates based on active learning principles. [6] extended the online ranking technique and proposed a temporal recommender system: when posting tweets, users can get recommendations of topics (hashtags) according to their real-time interests. Furthermore, users can also generate fast feedback according to the obtained recommendations. However, the above methods focus on online update and learning but neglect the temporal ordering information of the input data.

Recently, beyond the online learning methodology, the concept of real-time recommender systems is introduced, which emphasizes on the scalability and real-time pruning in recommender systems. Authors in [12] present a practical scalable item-based collaborative filtering (CF) algorithm, with the characteristics such as robustness to the implicit feedback problem, incremental update, and real-time pruning. Zhao *et. al.* [30] utilize a selective sampling scheme under the PMF framework [19] to achieve interactive updates. StreamRec [4] is invented based on a scalable CF recommendation model, which emphasized on the databases aspect of a stream processing system. A neighborhood-based method for streaming recommendations is discussed in [23]. The proposed framework is substantially different from the above real-time recommender systems: aforementioned systems are memory based methods and designed only for certain applications, while our method provides a principled way to handle data as streams.

5.2 Time-aware Recommendations

An important factor of time-aware recommendations is how to explicitly model users' interest change over time. In collaborative filtering, modeling temporal information has already shown its success (*e.g.* Ding and Li [9], Koren [14], Yin *et al.* [28] and Xiang *et al.* [26]). Zhang *et al.*[29] investigated the recurrence dynamics of social tagging. Xiong *et al.* [27] used tensor factorization approach to model temporal factor, where the latent factors are under the Markovian assumption. Bhargava *et al.* [2] further extended the tensor factorization to handle not only temporal but also geographical information. Liang *et al.* [16] proposed to use the implicit information network, and the temporal information of micro-blogs to find semantically and temporally relevant topics in order to profile user interest drifts. Liu *et al.* [17] extended collaborative filtering to a sequential matrix factorization model to enable time-aware parameterization with temporal smoothness regularization. Both [11] and [18] model the temporal dynamics using Kalman Filtering. Recently, Chang *et al.* [5] unified the task of link prediction and one-bit recommendation as a streaming positive-unlabeled learning problem. However, we differ from these methods proposed by providing more principled inference procedures. In addition, our data-driven technique automatic estimate all the parameters that have physical meanings (*i.e.* the half-time). Furthermore, as the temporal factors catching more attention, Burke *et al.* [3] studied an evolutional method, which can provide meaningful insights. Although many studies have been conducted in the temporal aspect of the data, most of them are retrospective and overlook the causality of the data generation process.

6. CONCLUSION

Data arrive at real-world recommender systems rapidly and continuously. The velocity and volume at which data is coming suggest the use of streaming settings for recommendations. We delineate three challenges for recommendations under streaming settings, which are real-time updating, unknown sizes and concept shift. We leverage these difficulties by proposing a streaming recommender system sRec in this paper. sRec manages stream inputs via a continuous-time random process. In addition, the proposed framework is not only able to track the dynamic changes of user/item topics, but also provides real-time recommendations in a prospective way. Further, we conduct experiments on three real-world datasets and sRec significantly outperforms other state-of-the-arts. It provides us an encouraging feedback to model data as streams.

7. ACKNOWLEDGMENT

This research was supported in part to Shiyu Chang and Thomas Huang by a research grant from Jump Labs. Yang Zhang and Mark Hasegawa-Johnson was in part supported by the AHRQ grant number 1-483711-392030-191100.

8. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *SIG KDD*, 2010.

[2] P. Bhargava, T. Phan, J. Zhou, and J. Lee. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *WWW*, 2015.

[3] R. Burke. Evaluating the dynamic properties of recommendation algorithms. In *ACM RecSys*, 2010.

[4] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. Mokbel. Streamrec: A real-time recommender system. In *SIGMOD*, 2011.

[5] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang. Positive-unlabeled learning in streaming networks. In *ACM SIGKDD*, 2016.

[6] C. Chen, H. Yin, J. Yao, and B. Cui. Terec: A temporal recommender system over tweet stream. *Proc. VLDB Endow.*, 6(12), Aug. 2013.

[7] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *WWW*, 2007.

[8] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *RecSys*, 2012.

[9] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM*, 2005.

[10] W. H. Greene. *Econometric analysis*. Granite Hill Publishers, 2008.

[11] S. Gultekin and J. Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *ICDM*, 2014.

[12] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *SIGMOD*, 2015.

[13] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD*, 2008.

[14] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4), 2010.

[15] A. Kyrola, G. E. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a pc. In *OSDI*, volume 12, pages 31–46, 2012.

[16] H. Liang, Y. Xu, D. Tjondronegoro, and P. Christen. Time-aware topic recommendation based on micro-blogs. In *CIKM*, 2012.

[17] N. N. Liu, L. He, and M. Zhao. Social temporal collaborative ranking for context aware movie recommendation. *ACM Trans. Intell. Syst. Technol.*, 4(1), Feb. 2013.

[18] Z. Lu, D. Agarwal, and I. S. Dhillon. A spatio-temporal approach to collaborative filtering. In *Recsys*, 2009.

[19] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[20] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *ACM SIGIR*. ACM, 2014.

[21] S. Rendle. Factorization machines. In *ICDM*, 2010.

[22] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Recsys*, 2008.

[23] K. Subbian, C. Aggarwal, and K. Hegde. Recommendations for streaming data. In *CIKM*, 2016.

[24] S. Wang, J. Tang, and H. Liu. Clare:a joint approach to label classification and tag recommendation. In *AAAI*, 2017.

[25] S. Wang, J. Tang, Y. Wang, and H. Liu. Exploring implicit hierarchical structures for recommender systems. In *IJCAI*, 2015.

[26] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *ACM SIGKDD*, 2010.

[27] L. Xiong, X. Chen, T. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, 2010.

[28] D. Yin, L. Hong, Z. Xue, and B. D. D. 0001. Temporal dynamics of user interests in tagging systems. In *AAAI*, 2011.

[29] D. Zhang, R. Mao, and W. Li. The recurrence dynamics of social tagging. In *WWW*, 2009.

[30] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *CIKM*, 2013.