Recurrent Coevolutionary Feature Embedding Processes for Recommendation

Hanjun Dai*, Yichen Wang*, Rakshit Trivedi, Le Song
College of Computing, Georgia Institute of Technology
{yichen.wang, hanjundai, rstrivedi}@gatech.edu, lsong@cc.gatech.edu

ABSTRACT

Recommender systems often use latent features to explain the behaviors of users and capture the properties of items. As users interact with different items over time, user and item features can influence each other, evolve and co-evolve over time. To accurately capture the fine grained *nonlinear* coevolution of these features, we propose a recurrent coevolutionary feature embedding process model, which combines recurrent neural network (RNN) with a multidimensional point process model. The RNN learns a nonlinear representation of user and item features which take into account mutual influence between user and item features, and the feature evolution over time. We also develop an efficient stochastic gradient algorithm for learning the model parameters, which can readily scale up to millions of events. Experiments on diverse real-world datasets demonstrate significant improvements in user behavior prediction compared to state-of-the-arts.

1. INTRODUCTION

E-commerce platforms and social service websites, such as Reddit, Amazon, and Netflix, attracts thousands of users every second. Effectively recommending the appropriate service items to users is a fundamentally important task for these online services. It can significantly boost the user activities on these sites and leads to increased product purchases and advertisement clicks.

"You are what you eat and you think what you read." The interactions between users and items play a critical role in driving the evolution of user interests and item features. For example, for music streaming services, a long-time fan of Rock music listens to an interesting Blues one day, and starts to listen to more Blues in stead of Rock music. Similarly, a single music may also serve different audiences at different times. For example, a music initially targeted for an older generation may become popular among the young, and the features of this music need to be updated.

*Authors have equal contributions.

Recsys Workshop on Deep Learning for Recommendation Systems (DLRS '16), September 15 2016, Boston, MA, USA

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Further more, as users interact with different items, users' interests and items' features can also *co-evolve* over time, *i.e.*, their features are intertwined and can influence each other:

- *user* $\rightarrow$ *item*. For instance, in online discussion forums, such as Reddit, although a group (item) is initially created for statistics topics, users with very different interest profiles can join this group. Therefore, the participants can shape the features of the group through their postings and responses. It is likely that this group can eventually become one about deep learning simply because most users here concern about deep learning.
- *item* $\rightarrow$ *user*. As the group is evolving towards topics on deep learning, some users may become more interested in deep learning topics, and they may participate in other specialized groups on deep learning. On the opposite side, some users may gradually gain interests in pure math groups, lose interests in statistics and become inactive in this group.

Such co-evolutionary nature of user-item interactions raises very important questions on how to model them and how to learn them from observed data. Further more, nowadays large amount of user-item interaction data are becoming increasingly available online. In addition to the precise time-stamps of the interactions, many datasets also contain additional context such as text, image, and video. There is urgent need to design new models, and learning and inference algorithms to leverage the huge potential of such data.

However, existing methods either treat the temporal user-item interactions data as a static graph or use epoch based methods such as tensor factorization to learn the latent features [**?**]. These methods are not able to capture the fine grained temporal dynamics of user-item interactions. Recent point process based models treat time as a random variable and improves over the traditional methods significantly [1]. However, point process based methods typically make strong assumptions about the function form of the generative processes, which may not reflect the reality or may not be accurate enough to capture the complex and *nonlinear* user-item influence in real world. Moreover, it is not easy to incorporate the observed context features in such point process model.

How can we obtain a more expressive model to capture the co-evolution features of user-item interactions, and learn such a model from large volume of data? To tackle this challenge, in this paper, we combine recurrent neural network (RNN) with multivariate point process models [2], and propose a recurrent coevolutionary feature embedding process framework. In particular, our work makes the following contributions:

- We propose a novel model that captures the *nonlinear* co-evolution nature of users' and items' latent features. Our model assigns an evolving feature embedding process for each user and item, and the co-evolution of these latent feature processes is considered using two parallel components: (i) *item* $\rightarrow$ *user* component, a user's latent feature is determined by the nonlinear embedding of latent features of the items he interacted with; and (ii) *user* $\rightarrow$ *item* component, conversely, an item's latent features are also determined by the latent features of the users who interact with the item.

- We use recurrent neural network to parametrize the nonlinear embedding and it can also take into account the presence of potentially high dimensional observed context features.

- We evaluate our method over multiple datasets, verifying that our method can lead to significant improvements in user behavior prediction compared to previous state-of-the-arts. Precise time prediction is especially novel and not possible by most prior work.

2. RELATED WORK

Recent work predominantly fix the latent features assigned to each user and item [3, 4, 5, 6, 7, 8, 9, 10]. In more sophisticated methods, the time is divided into epochs, and static latent feature models are applied to each epoch to capture some temporal aspects of the data [11, 12, 13, 12, 13, 14, 15, 16, 17, 18, 19, 20]. For such methods, it is not clear how to choose the epoch length parameter. First, different users may have very different timescale when they interact with those service items, making it difficult to choose a unified epoch length. Second, it is not easy for these methods to answer time-sensitive queries such as when a user will return to the service item. The predictions are only in the resolution of the chosen epoch length. Recently, [1] proposed a low-rank point process based model for time- sensitive recommendations from recurrent user activities. However, it fails to capture the heterogeneous coevolutionary properties of user-item interactions.

In the deep learning community, [21] proposed collaborative deep learning, a hierarchical Bayesian model that jointly performs learning for the content features and collaborative filtering for the ratings matrix. This method considers interaction data as static graph and also does not capture latent coevolutionary properties of user-item interactions. [22] applied recurrent neural network based approach to recommender systems. Specifically, they adopt item-to-item recommendation approach but use session based data with temporal ordering to capture influences of past interactions in particular session. However, it does not consider evolving and co-evolving features of users and items interacting with each other, partly because it is designed for the scenario where user information is not available. Finally, our work is inspired from newly proposed recurrent marked temporal point process framework [23] that builds a connection between RNN and Point Processes. However, [23] focuses on the task of next event prediction given a sequence of past events for an entity and is only designed for one-dimension point process. Significant generations and extensions are needed for the recommendation system setting with feature coevolution.

3. BACKGROUND ON TEMPORAL POINT PROCESSES

A temporal point process [24, 25] is a random process whose realization consists of a list of discrete events localized in time, $\{t_i\}$ with $t_i \in \mathbb{R}^+$ and $i \in \mathbb{Z}^+$. Equivalently, a given temporal point process can be represented as a counting process, $N(t)$, which records the number of events before time t. An important way to characterize temporal point processes is via the conditional intensity function $\lambda(t)$, a stochastic model for the time of the next event given all the previous events. Formally, $\lambda(t)dt$ is the conditional probability of observing an event in a small window $[t, t+dt)$ given the history $\mathcal{H}(t)$ up to t and that the event has not happen before t, *i.e.*,

$$\lambda(t)\mathrm{d}t := \mathbb{P}\left\{\text{event in } [t, t+\mathrm{d}t)|\mathcal{H}(t)\right\} = \mathbb{E}[\mathrm{d}N(t)|\mathcal{H}(t)]$$

, where one typically assumes that only one event can happen in a small window of size $\mathrm{d}t$, *i.e.*, $\mathrm{d}N(t) \in \{0, 1\}$.

Then, given a time $t \geqslant 0$, we can also characterize the conditional probability that no event happens during $[0, t)$ as [26]:

$$S(t) = \exp(-\int_0^t \lambda(\tau)\,\mathrm{d}\tau)$$

and the conditional density that an event occurs at time t is defined as

$$f(t) = \lambda(t)\,S(t) \tag{1}$$

The function form of the intensity $\lambda(t)$ is often designed to capture the phenomena of interests. Commonly used form includes:

- Hawkes processes [27, 28], whose intensity models the excitation between events, *i.e.*, $\lambda(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i)$, where $\kappa_\omega(t) := \exp(-\omega t)\mathbb{I}[t \geqslant 0]$ is an exponential triggering kernel, $\mu \geqslant 0$ is a baseline intensity independent of the history. Here, the occurrence of each historical event increases the intensity by a certain amount determined by the kernel κ_ω and the weight $\alpha \geqslant 0$, making the intensity history dependent and a stochastic process by itself.

- Rayleigh process, whose intensity function is

$$\lambda(t) = \alpha t \tag{2}$$

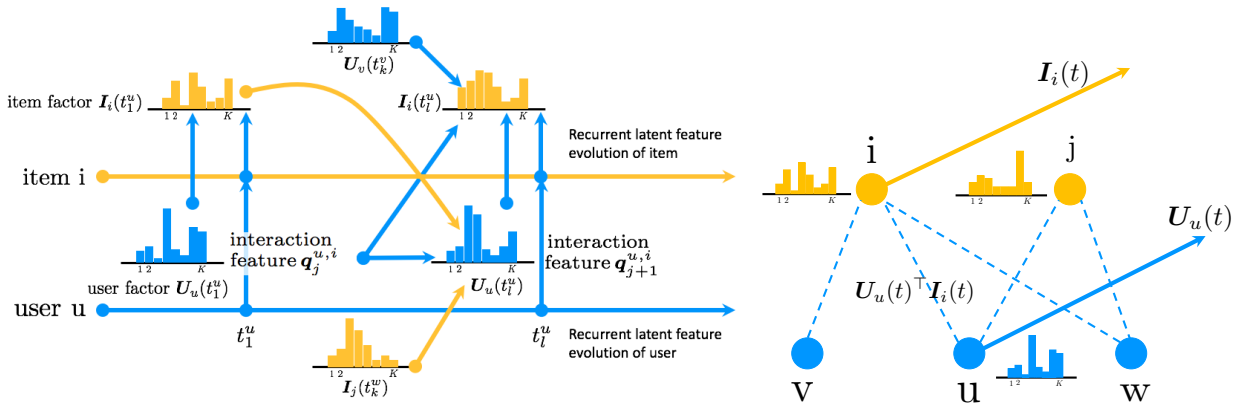
  where $\alpha > 0$ is the weight parameter.

4. RECURRENT COEVOLUTIONARY FEATURE EMBEDDING PROCESSES

In this section, we present the generative framework for modeling the temporal dynamics of user-item interactions. We first explicitly capture the co-evolving nature of users' and items' latent feature. Then, based on the compatibility between the users' and items' latent feature, we model the user-item interactions by a temporal point process and parametrize the intensity function by the compatibility.

4.1 Event representation

Given m users and n items, we denote the ordered list of N observed events as $\mathcal{O} = \{e_j = (u_j, i_j, t_j, q_j)\}_{j=1}^N$ on time window $[0, T]$, where $t_1 \leqslant \ldots \leqslant N$. Each event is modeled as the tuple (u_j, i_j, t_j, q_j), where $u_j \in \{1, \ldots, m\}$,



(a) Co-evolving feature embedding processes. (b) Temporal interactions.

Figure 1: Model illustration for user u (blue) and item i (yellow). (a) User features and item features influence each other and co-evolve over time. At time t_l^u, the latent feature $I_i(t_l^u)$ of item i (in yellow) is influenced by the users' feature ($\boldsymbol{U}_u(t_l^u)$, $\boldsymbol{U}_v(t_k^v)$), and the cumulative past interaction features ($\boldsymbol{q}_j^{u,i}$, $\boldsymbol{q}_{j+1}^{u,i}$) that captured by RNN. Conversely, $\boldsymbol{U}_u(t_l^u)$ is influenced by the item features ($\boldsymbol{I}_i(t_1^u)$,$\boldsymbol{I}_j(t_k^w)$), and by past interaction feature $\boldsymbol{q}_j^{u,i}$. (b) The temporal interacting process where the tendency that a user will interact with an item depends on the compatibility of their latent features. u is more likely to interact with i than j.

$i_j \in \{1, \ldots, n\}$, $t_j \in \mathbb{R}^+$, which means that the interaction between user u_j, item i_j at time t_j, with the interaction context $q_j \in \mathbb{R}^d$. Here q_j can be a high dimension vector such as the text review, or simply the embedding of static user/item features such as user's profile and item's categorical features. For notation simplicity, we define

- $\mathcal{O}^u = \{e_j^u = (i_j^u, t_j^u, q_j^u)\}_{j=1}^{|\mathcal{O}^u|}$ as the ordered listed of all events related to user u.
- Similarly we have $\mathcal{O}^i = \{e_j^i = (u_j^i, t_j^i, q_j^i)\}_{j=1}^{|\mathcal{O}^i|}$ as the ordered list of all events related to item i.

We also set $t_0^i = t_0^u = 0$ for all the users and items. We will also use t_k- to denote the time point just before time t_k.

4.2 Recurrent feature embedding processes

We associate latent features $\boldsymbol{U}_u(t) \in \mathbb{R}^k$ with each user u and $\boldsymbol{I}_i(t) \in \mathbb{R}^k$ with each item i. These features represent the subtle properties which cannot be directly observed, such as the interests of a user and the semantic topics of an item. Specifically, we model the *drift, evolution, and co-evolution* of $\boldsymbol{U}_u(t)$ and $\boldsymbol{I}_i(t)$ as follows a piecewise constant function of time and has jumps only at event times. Specifically, we have define:

- **User embedding process.** For each user u, the corresponding embedding after user u's k-th event $e_k^u = (i_k^u, t_k^u, q_k^u)$ can be formulated as:

$$\boldsymbol{U}_u(t_k^u) = \sigma\Big(\underbrace{\boldsymbol{W}_1(t_k^u - t_{k-1}^u)}_{\text{temporal drift}} + \underbrace{\boldsymbol{W}_2 \boldsymbol{U}_u(t_{k-1}^u)}_{\text{self evolution}} + \underbrace{\boldsymbol{W}_3 \boldsymbol{I}_{i_k}(t_k^u-)}_{\text{co-evolution: item feature}} + \underbrace{\boldsymbol{W}_4 \boldsymbol{q}_k^{u,i_k}}_{\text{interaction feature}} \Big) \quad (3)$$

- **Item embedding process.** For each item i, we specify $\boldsymbol{I}_i(t)$ at time t_k^i as:

$$\boldsymbol{I}_i(t_k^i) = \sigma\Big(\underbrace{\boldsymbol{V}_1(t_k^i - t_{k-1}^i)}_{\text{temporal drift}} + \underbrace{\boldsymbol{V}_2 \boldsymbol{I}_i(t_{k-1}^i)}_{\text{self evolution}} + \underbrace{\boldsymbol{V}_3 \boldsymbol{U}_{u_k}(t_k^i-)}_{\text{co-evolution: item feature}} + \underbrace{\boldsymbol{V}_4 \boldsymbol{q}_k^{i,u_k}}_{\text{interaction feature}} \Big) \quad (4)$$

where $t-$ means the time point just before time t, $\boldsymbol{W}_4, \boldsymbol{V}_4 \in \boldsymbol{R}^{k \times d}$ are the embedding matrices mapping from the explicit high-dimensional feature space into the low-rank latent feature space and $\boldsymbol{W}_i, \boldsymbol{V}_i \in \boldsymbol{R}^{K \times K}, i = 1, 2, 3$ are weights parameters.

$\sigma(\cdot)$ is the nonlinear activation function, such as commonly used ReLU, Tanh, or Sigmoid. For simplicity, we use basic recurrent neural network to formulate the recurrence, but it is also straightforward to extend it using GRU or LSTM to gain more expressive power. Figure 1 summarizes the basic setting of our model.

Here both the user and item's feature embedding processes are piecewise constant functions of time and only updated if an interaction event happens. A user's attribute changes only when he had a new interaction with some item. For example, a user's taste for music would change only when he listened to some new or old musics. Also, an item's attribute would change only when some user interacts with it. Hence, the key idea is we only need to model the points when the embedding needs to evolve. Next we discuss the rationale of each term in detail:

- **Temporal drift**. The first term is defined based on the time difference between consecutive events of specific user or item. It allows the basic features of users (*e.g.*, a user's self-crafted interests) and items (*e.g.*, textual categories and descriptions) to smoothly drift through time. Such changes of basic features normally are caused by external influences.

- **Self evolution**. The current user feature should also be influenced by its feature at the earlier time. This captures the intrinsic evolution of user/item features. For example, a user's current taste should be more or less similar to his/her tastes two days ago.

- **Evolution with interaction features**. Users' and items' features can evolve and be influenced by the characteristics of their interactions. For instance, the genre changes of movies indicate the changing tastes of users. The theme of a chatting-group can be easily

shifted to certain topics of the involved discussions. In consequence, this term captures the influence of the current interaction features to the changes of the latent user (item) features.

- **User-item coevolution**. Users' and items' latent features can mutually influence each other. This term captures the two parallel processes. First, a user's latent feature is determined by the latent features of the items he interacted with. At each time t_k, the latent item feature is $\boldsymbol{I}_{i_k}(t_k^u)$. In our model, we capture both the temporal influence and feature of each history item as a latent process. Conversely, an item's latent features are determined by the latent features of the user who just interacts with the item.

- **Interaction feature**. The interaction feature is the additional information/data happened in the user-item interactions. For example, in online discussion forums such as Reddit, the interaction feature is the posts and comments made by the user. In the online review sites such as Yelp, it is the reviews of the businesses.

To summarize, each feature embedding process evolves according to the respective base temporal user (item) features and also are mutually dependent on each other due to the endogenous influences from the interaction features and the entangled latent features.

4.3 User-item interactions as temporal point processes

For each user, we model the recurrent occurrences of user u's interaction with all items as a multi-dimensional temporal point process, with each item as one dimension. In particular, the intensity in the i-th dimension (item i) is modeled as a Rayleigh process:

$$\lambda^{u,i}(t-t_0) = \underbrace{\exp\left(U_u(t-)^\top I_i(t-)\right)}_{\text{user-item compatibility}} * \underbrace{(t-t_0)}_{\text{time lapse}} \qquad (5)$$

where $t \geqslant t_0$, and $t-$ means the time point just before time t. The rationale behind this formulation is three fold:

- *Time as a random variable.* Instead of discretizing the time into epochs as in traditional mehtods [16, 17, 18, 19, 20], we explicitly model the timing of each interaction event as a random variable, which naturally captures the heterogeneity of the temporal interactions between users and items.
- *Short term preference.* The probability for user u to interact with item i at time t depends on the compatibility of their instantaneous latent features. Such compatibility is evaluated through the inner product of their latent features at the last time t_0. It serves as α in (2).
- *Rayleigh time distribution.* The user and item feature embeddings are piecewise constant, and we use this Rayleigh term to make the intensity function to be piecewise linear. This formulation assumes a Rayleigh distribution for the time intervals between consecutive events in each dimension [29]. It is well-adapted to modeling fads, where the infection likelihood f in (1) rises to a peak and then drops extremely rapidly. Furthermore, it is computationally easy to compute integration and get analytic form of f. One can then use f to make item recommendation by finding the dimension that reaches the peak.

Because $\boldsymbol{U}_u(t)$ and $\boldsymbol{I}_i(t)$ co-evolve through time, their inner-product measures a general representation of the cumulative influence from the past interactions to the occurrence of the current event. When the product is positive, it indicates a self-exciting behavior that most recent activities will trigger more events in the near future. For instance, one may repeatedly listen to a newly bought album within a short-time window. When the product becomes negative, it represents a self-correcting behavior that most recent interactions will decrease the chance of more future events. For example, after one keeps listening to the same album for a long time, he may become bored and thus changes interests to other items.

Given a collection of events recorded within a time window $[0, T)$, we can further estimate the parameters using maximum likelihood estimation of all events. The joint negative log-likelihood is [30]:

$$\ell = -\sum_{j=1}^{N} \log\left(\lambda^{u_j,i_j}(t_j)\right) - \sum_{u=1}^{m}\sum_{i=1}^{n}\int_0^T \lambda^{u,i}(\tau)\, d\tau \qquad (6)$$

where each event from $\mathcal{O}$ will have one term in the the first summation, and the each pair of potential item-user interaction will have one term in the second double summation. One advantage of point process formulation is that the non-presence of an interaction at particular point in time is nicely taken into account in survival terms in the second double summation.

5. PARAMETER LEARNING

Having presented the model, in this section, we propose an efficient algortihm to learn the parameters. Though we presented batch objective function in Equation 6, we seek to use stochastic methods to learn the embedding parameters $\{V_i\}_{i=1}^4$ and $\{W_i\}_{i=1}^4$. The Adam Optimizer [31] is used in our experiment, since it has shown good performance in training RNNs., and use gradient clip to avoid gradient explosion.

The Back Propagation Through Time (BPTT) is the standard way to train a RNN. To make the back propagation tractable, one typically needs to do truncation during training. Different from traditional sequential data where one can easily break the sequences into multiple segments to make the BPTT trackable, here all the events are related to each other by the user-item bipartite graph, which makes it hard to decompose.

To do this, we first order all the events globally and then do mini-batch training in a sliding window fashion. Each time when conducting feed forward and back propagation, we take the consecutive events within current sliding window to build the computational graph. In our case the truncation is on the global timeline, instead over individual independent sequencs. Another benefit of ordering events globally is that it allows us to keep the user and item latent features that could be used for the future mini-batch training. Figure 2 illustrates our training method.

Since the user-item interactions vary a lot across mini-batches, the corresponding computational graph also changes greatly. To make the learning efficient, we use the graph embedding framework [32] which allows training deep learning models where each term in the objective has a different computational graphs but with shared parameters.

Next, we discuss in details on gradient computation. First, note that the intensity function $\lambda^{u,i}(t)$ is piecewise linear, hence the integration in (6) can also be computed in a piecewise fashion with closed form, where the number of pieces equals to the total number of events happened to user u and item i separately.

Computing gradient For illustration purpose, we here use Sigmoid as the nonlinear activation function σ. In order to get gradient with respect to parameter $\boldsymbol{W}$s, we first compute gradients with respect to each varying points of embeddings. For user u's embedding after his k-th event, the corresponding partial derivatives are computed by:

$$\frac{\partial \ell}{\partial U_u(t_k^u)} = \underbrace{-I_{i_k^u}}_{\text{from intensity}} + \underbrace{\sum_{i=1}^{n} \frac{\partial \int_{t_k^u}^{t_{k+1}^u} \lambda^{u,i}(\tau) d\tau}{\partial U_u(t_k^u)}}_{\text{from survival}} + \tag{7}$$

$$\underbrace{\frac{\partial \ell}{\partial U_u(t_{k+1}^u)} \odot (1 - U_u(t_{k+1}^u)) \odot U_u(t_{k+1}^u) \boldsymbol{W}_2}_{\text{from user } u\text{'s next embedding}}$$

$$+ \underbrace{\frac{\partial \ell}{\partial I_{i_{k+1}^u}(t_{k+1}^u)} \odot (1 - I_{i_{k+1}^u}(t_{k+1}^u)) \odot I_{i_{k+1}^u}(t_{k+1}^u)}_{\text{from user } u\text{'s next item embedding}}$$

where $\odot$ denotes element-wise multiplication.

The gradient coming from the second term (*i.e.*, the survival term) is also easy to compute, since the Rayleigh distribution has closed form of survival function. For a certain item i, if its feature doesn't changed between time interval $[t_k^u, t_{k+1}^u]$, then we have

$$\frac{\partial \int_{t_k^u}^{t_{k+1}^u} \lambda^{u,i}(\tau) d\tau}{\partial U_u(t_k^u)} = \frac{(t_{k+1}^u - t_k^u)^2}{2} \exp\left(U_u(t_k^u)^\top I_i(t_k^u) I_i(t_k^u)\right) \tag{8}$$

On the other hand, if the embedding of item i changes during this time interval, then we should break this interval into segments and compute the summation of gradients in each segment in a way similar to (8). Thus, we are able to compute the gradients with respect to $\boldsymbol{W}_i, i \in \{1, 2, 3, 4\}$ as follows.

$$\frac{\partial \ell}{\partial \boldsymbol{W}_1} = \sum_{u=1}^{m} \sum_{k} \frac{\partial \ell}{\partial U_u(t_k^u)} \odot (1 - U_u(t_k^u)) \odot U_u(t_k^u)(t_k^u - t_{k-1}^u)$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_2} = \sum_{u=1}^{m} \sum_{k} \left(\frac{\partial \ell}{\partial U_u(t_k^u)} \odot (1 - U_u(t_k^u)) \odot U_u(t_k^u) \right) U_u(t_{k-1}^u)^\top$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_3} = \sum_{u=1}^{m} \sum_{k} \left(\frac{\partial \ell}{\partial U_u(t_k^u)} \odot (1 - U_u(t_k^u)) \odot U_u(t_k^u) \right) I_{i_k}(t_k^u -)^\top$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_4} = \sum_{u=1}^{m} \sum_{k} \left(\frac{\partial \ell}{\partial U_u(t_k^u)} \odot (1 - U_u(t_k^u)) \odot U_u(t_k^u) \right) q_k^{u,i_k}$$

Since the items are treated symmetrically as users, the corresponding derivatives can be obtained in a similar way.

6. EXPERIMENTS

We evaluate our model on real-world datasets. For each sequence of user activities, we use all the events up to time $T \cdot p$ as the training data, and the rest events as the testing data, where T is the observation window. We report the results on two tasks:

- *Item prediction.* At each test time, we predict the item that the user will interact with. We rank all the items in the descending order of the conditional density $f^{u,i}(t) = \lambda^{u,i}(t) S^{u,i}(t)$ to produce a recommendation list. We report the Mean Average Rank (MAR). The smaller the value, the better performance.
- *Time prediction.* We predict the time when a testing event will occur between a given user-item pair. We predict the expectation of next event time on current pair. Using Rayleigh distribution, this number is given by $\mathbb{E}^{u,i}(t|t0) = \sqrt{\frac{\pi}{2 \exp\left(U_u(t-)^\top I_i(t-)\right)}}$. We report the Mean Absolute Error (MAE) between the predicted and true time.

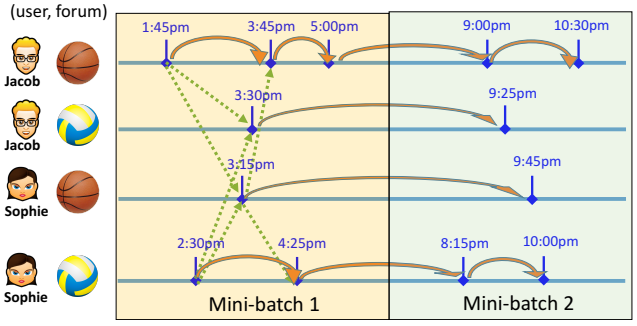


Figure 2: Illustration of BPTT with budget. Here shows the process for two users and two forums. The dependency within each dimension is represented by orange arrow, while the across dimension dependency for the first three events is noted by green dash line. The hidden states computed by mini-batch 1 are inherited by next mini-batch.

6.1 Competitors

We compared our method to the following algorithms:

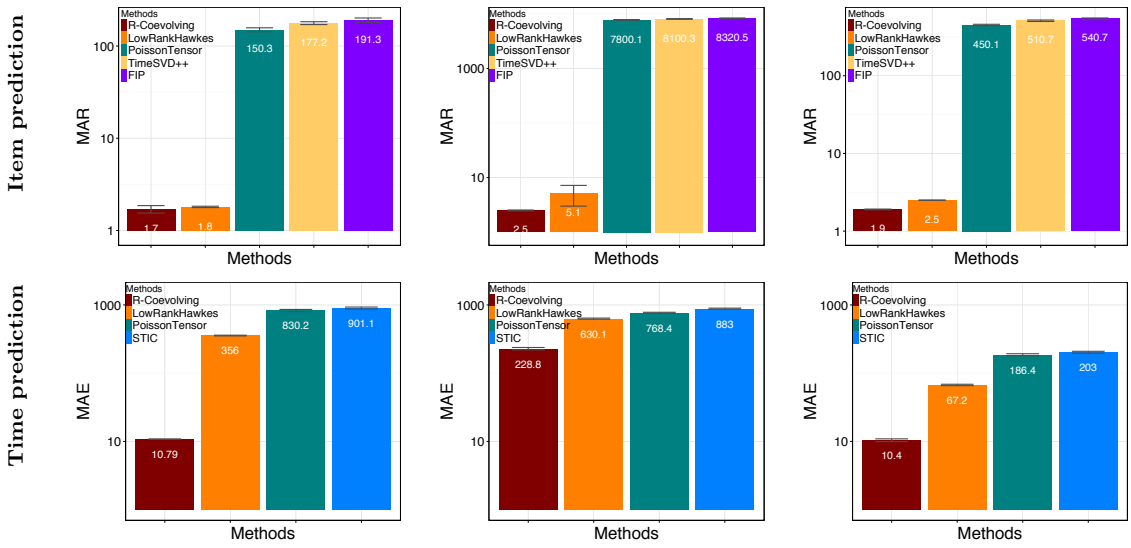
- **PoissonTensor** [14]: Poisson Tensor Factorization has been shown to perform better than factorization methods based on squared loss [12, 13, 33] on recommendation tasks. The performance for this baseline is reported using the average of the parameters fitted over all time intervals.
- **LowRankHawkes** [1]: This is a low rank point process based model which assumes user-item interactions to be independent of each other and does not capture the co-evolution of user and item features.
- **STIC** [34]: it fits a semi-hidden markov model to each observed user-item pair and is only designed for time prediction.
- **TimeSVD++ [11] and FIP [8]**: These two methods are only designed for explicit ratings, the implicit user feedbacks (in the form of a series of interaction events) are converted into the explicit ratings by the respective frequency of interactions with users.

6.2 Datasets

We use three real world datasets.

IPTV. It contains 7,100 users' watching history of 385 TV programs in 11 months (Jan 1 - Nov 30 2012), with around 2M events, and 1,420 movie features (including 1,073 actors, 312 directors, 22 278 genres, 8 countries and 5 years).

Yelp. This data was available in Yelp Dataset challenge Round 7. It contains reviews for various businesses from October, 2004 to December, 2015. Out of available 552K users, we used users with more than 100 posts for our experiments. We cleaned the review text by removing stop words



(a) IPTV (b) Yelp (c) Reddit

Figure 3: Prediction results on three real world datasets.

and punctuation marks and only included words of length $>$ 3 and frequency $>$ 10. After this pre-processing, the dataset comprised of 1,503 users, 47,924 groups (businesses) and 34,508 text features with a total of 2,92,000 reviews. To be able to compare with baselines, we further decreased the size of this processed dataset and used a total of 95,000 reviews between randomly selected 95 users and 17,205 businesses.

Reddit. We collected discussion related data on different subreddits (groups) for the month of January 2014. We filtered all bot users' and their posts from this dataset. Similar to Yelp dataset, we cleaned the text of posts to remove stop words and punctuation marks and only include words of length $>$ 3 and frequency $>$ 10. Furthermore, we only considered top 10,000 users sorted according to the frequency of posts and randomly selected 1,000 users out of it to create smaller dataset. After all pre-processing, the dataset consists of 1,000 users, 1,403 groups and 82,389 text features. This dataset contains a total of 10,000 discussion events.

6.3 Results

Item Recommendation From Figure 3 we can see, our method significantly outperforms epoch-based baselines in terms of item prediction on all the datasets. While the best possible MAR one can achieve is 1, both our method and LowRankHawkes got quite accurate results. Regarding the MAR metric, the performance is also slightly better compared with LowRankHawkes. Since one only need the rank of conditional density f to conduct item prediction, LowRankHawkes may still be good at differentiating f, but could not learn the actual value of f accurately, as shown in the time prediction task where the value of f is needed for precise prediction.

Time Prediction On time prediction, R-coevolve significantly outperforms other methods. For example, compared with LowRankHawkes, it has 2$\times$ time improvement on Yelp, 6$\times$ improvement on Reddit, and 30$\times$ improvement on IPTV. The time unit is hour. Hence it has 2 weeks accuracy improvement on IPTV and 2 days on Reddit. This is important for online merchants to make time sensitive recommendations. An intuitive explanation is that our method accurately captures the *nonlinear* pattern between user and item interactions. The competitor LowRankHawkes assumes specific parametric forms of the user-item interaction process, hence may not be accurate or expressive enough to capture real world temporal patterns. Furthermore, the LowRankHawkes modeled each user-item interaction dimension independently, which may lose the important affection from user's interaction with other items while predicting the current item's reoccurance time.

7. CONCLUSION

We have proposed an efficient framework for modeling the co-evolution nature of users' and items' latent features. It is a generative model designed for modeling and understanding user's online behaviors, which is different from prior work that only focuses on the prediction task in the recommender system. Moreover, the user and item's evolving and co-evolving processes are captured by the RNN. We demonstrate the superior performance of our method on the time prediction task, which is not possible by most prior work. Future work includes extending to other applications such as modeling dynamics of social message groups, and understanding peoples' behaviors on Q&A sites.

Acknowledge

This project was supported in part by NSF/NIH BIGDATA 1R01GM108341, ONR N00014-15-1- 2340, NSF IIS-1639792, NSF IIS-1218749, NSF CAREER IIS-1350983, Intel and NVIDIA.

8. REFERENCES

[1] Nan Du, Yichen Wang, Niao He, and Le Song. Time sensitive recommendation from recurrent user activities. In *NIPS*, 2015.

[2] Thomas Josef Liniger. *Multivariate Hawkes Processes*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2009.

[3] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo.

In W.W. Cohen, A. McCallum, and S.T. Roweis, editors, *ICML*, volume 307, pages 880–887. ACM, 2008.

[4] Y. Chen, D. Pavlov, and J.F. Canny. Large-scale behavioral targeting. In J.F. Elder, F. Fogelman-Soulié, P.A. Flach, and M. J. Zaki, editors, *KDD*, pages 209–218. ACM, 2009.

[5] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In J.F. Elder, F. Fogelman-Soulié, P.A. Flach, and M.J. Zaki, editors, *KDD*, pages 19–28. ACM, 2009.

[6] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[7] Yehuda Koren and Joe Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *RecSys*, 2011.

[8] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. Like like alike: joint friendship and interest propagation in social networks. In *WWW*, 2011.

[9] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. Beyond clicks: Dwell time for personalization. In *RecSys*, 2014.

[10] Yichen Wang and Aditya Pal. Detecting emotions in social media: A constrained optimization approach. In *IJCAI*, 2015.

[11] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, 2009.

[12] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Recsys*, pages 79–86. ACM, 2010.

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

[14] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[15] San Gultekin and John Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *ICDM*, pages 140–149, 2014.

[16] Laurent Charlin, Rajesh Ranganath, James McInerney, and David M Blei. Dynamic poisson factorization. In *RecSys*, 2015.

[17] Jiayu Zhou Juhan Lee Preeti Bhargava, Thomas Phan. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *WWW*, 2015.

[18] Prem Gopalan, Jake M Hofman, and David M Blei. Scalable recommendation with hierarchical poisson factorization. *UAI*, 2015.

[19] Balázs Hidasi and Domonkos Tikk. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, pages 1–30, 2015.

[20] Xin Wang, Roger Donaldson, Christopher Nell, Peter Gorniak, Martin Ester, and Jiajun Bu. Recommending groups to users using user-group engagement and time-dependent matrix factorization. In *AAAI*, 2016.

[21] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *KDD*. ACM, 2015.

[22] Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[23] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 2016.

[24] D.R. Cox and V. Isham. *Point processes*, volume 12. Chapman & Hall/CRC, 1980.

[25] D.R. Cox and P.A.W. Lewis. Multivariate point processes. *Selected Statistical Papers of Sir David Cox: Volume 1, Design of Investigations, Statistical Methods and Applications*, 1:159, 2006.

[26] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. *Survival and event history analysis: a process point of view*. Springer, 2008.

[27] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[28] Yichen Wang, Bo Xie, Nan Du, and Le Song. Isotonic hawkes processes. In *ICML*, 2016.

[29] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the International Conference on Machine Learning*, 2011.

[30] D.J. Daley and D. Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*, volume 2. Springer, 2007.

[31] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.

[33] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *KDD*, 2015.

[34] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. Just in time recommendations: Modeling the dynamics of boredom in activity streams. In *WSDM*, 2015.