Gated Spectral Units: Modeling Co-evolving Patterns for Sequential Recommendation

Anonymous Author(s)

ABSTRACT

Exploiting historical data of users to make future predictions lives at the heart of building effective recommender systems (RS). Recent approaches for sequential recommendations often render past actions of a user into a sequence, seeking to capture the temporal dynamics in the sequence to predict the next item. However, the interests of users evolve over time together due to their mutual influence, and most of existing methods lack the ability to utilize the rich coevolutionary patterns available in underlying data represented by sequential graphs.

In order to capture the co-evolving knowledge for sequential recommendations, we start from introducing an efficient spectral convolution operation to discover complex relationships between users and items from the spectral domain of a graph, where the hidden connectivity information of the graph can be revealed. Then, the spectral convolution is generalized into an recurrent method by utilizing gated mechanisms to model sequential graphs. Experimentally, we demonstrate the advantages of modeling *co-evovling patterns*, and Gated Spectral Units (GSUs) achieve state-of-the-art performance on several benchmark datasets. All code and data are available at *https://github.com/anounymous1234/GSUs*.

KEYWORDS

Sequential Recommendation, Spectral, Graph

ACM Reference Format:
Anonymous Author(s). 1997. Gated Spectral Units: Modeling Co-evolving Patterns for Sequential Recommendation. In *Proceedings of 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

What will a customer buy next? The importance of this question cannot be overstated for building effective recommender systems (RS). RS intersect multiple products and customers, where characteristics of users and perceptions of items not only shift over time but also influence each other. This complex temporal information raises unique challenges.

In order to build a predictive model for users' future purchases, we observe that a user's actions are correlated to not only his or her past activities but also other users' behaviors. Interests of users co-evolve over time and their preferences influence each other

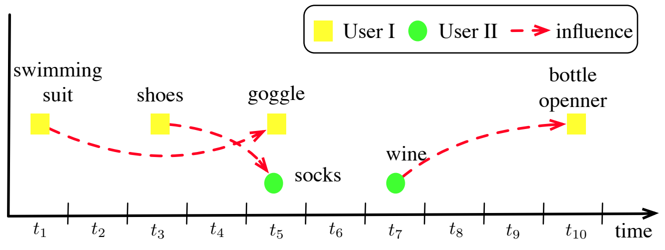

Figure 1: An example illustrates how activities of user co-evovles over time. Yellow and green circles denote purchases of user I and user II, respectively. (Best viewed in color)

dynamically. For example, as shown in Fig. 1, if user u_1 is related to user u_2, when u_1 purchases a pair of shoes at time t_3, u_2 may buy a pair of *socks* for u_1 at a later time, say t_5. Some time later (t_7), when u_2 shops a bottle of *wine*, it is reasonable to expect u_1 to be interested in a *bottle opener*. We term this phenomenon of evolving actions of users and their mutual influence over time as *co-evovling patterns*. It is no doubt that effectively capturing such rich patterns can help reason over complex non-linear user-item interactions.

Existing approaches often lack the ability of learning the co-evovling knowledge, resulting in a limited understanding on behaviors of users and how they influence each other over time. Early works, such as TimeSVD++ [9], focus on modeling the shifting patterns of a user's preferences and the popularity of an item by introducing additional variables changing over time. Recent models first regard activities of a user as a sequence, and then propose Markov Chain (MC) based methods to capture the item dependencies or correlations within the sequence. For instance, in Fig. 1, user u_1 shops a goggle at t_5 time because of the purchase of a *swimming suit* at the time of t_1. Another line of work adopts Recurrent Neural Networks (RNNs) to model the sequence. However, almost all of them fail to capture the rich *co-evovling patterns*.

In this paper, in order to utilize the *co-evovling patterns* and capture dependencies between actions across different users, we first formulate timestamped user-item interactions into Sequential Evolving Graphs (SEGs) (see Definition 2.1), where co-evolutionary knowledge can be revealed. Then, we generalize a spectral unit into a recurrent model by introducing gated mechanisms [2] to model the *co-evovling patterns* from spectral domains. The proposed model, Gated Spectral Units (GSUs), recurrently takes a sequence of graphs as input, and learns state vectors of users and items to summarize *co-evovling patterns* within the sequence. Our work makes the following contributions:

- **Novelty**: To our knowledge, it is the very first recommendation method to model sequential graphs from spectral domains.
- **Demonstrated Effectiveness**: It is demonstrated that the *co-evovling patterns* can be effectively captured from spectral domains of temporal graphs.

Table 1: Notations

Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{I}_i^+, \mathcal{I}_i^-$	a set of items liked by user i, and all other items without interactions with user i
n_u, n_i	number of users and items
$\mathcal{G}_t = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_t\}$	the t_{th} graph consisting of user set $\mathcal{U}$, item set $\mathcal{I}$ and edge set $\mathcal{E}_t$
$\mathbf{L}_t$	the laplacian matrix of the t_{th} graph
$\mathbf{U}_t^{(l)}, \mathbf{\Lambda}_t^{(l)}$	the top-l eigenvectors and eigenvalues of $\mathbf{L}_t$
$\mathbf{H}_t^u, \mathbf{H}_t^i$	hidden state matrices of users and items at timestamp t
$\mathbf{W}_z, \mathbf{W}_h, \mathbf{W}_r$	convolutional filters of update, candidate and reset gate
$\mathbf{b}_z, \mathbf{b}_h, \mathbf{b}_r$	bias vectors of update, candidate and reset gate
Δ	time interval

- **High Performance**: Benefiting from the *co-evovling* knowledge being effectively captured, GSU significantly outperforms state-of-the-art methods on three real-world datasets.

2 BACKGROUND AND PRELIMINARIES

This paper focuses on the recommendation problem with implicit feedbacks (e.g. clicks, purchases, likes), where we only observe whether a person has interacted with an item and do not observe explicit ratings. Let us denote a set of users as $\mathcal{U}$ and an item set $\mathcal{I}$. $\mathcal{I}_i^+$ represents the set of all items liked by user i and $\mathcal{I}_i^-$ stands for the remaining items. Each user-item interaction is represented as a tuple $(i, j, \hat{t})$, denoting that user i has interacted with item j at timestamp $\hat{t}$. We define *Sequential Evolving Graphs (SEGs)* as:

Definition 2.1. *(Sequential Evolving Graphs). Sequential Evovling Graphs (SEGs) are represented as a sequence of bipartite graphs $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t, ...\}$. The t_{th} bipartite graph $\mathcal{G}_t$ is defined as $\mathcal{G}_t = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_t\}$, where $\mathcal{U}$ and $\mathcal{I}$ are the user set and item set, respectively and $\mathcal{E}_t$ denotes an edge set connecting users in $\mathcal{U}$ and items in $\mathcal{I}$. For an edge $(i, j, \hat{t}) \in \mathcal{E}_t$, it denotes user i has interacted with item j at timestamp $\hat{t}$ when $\Delta(t-1) < \hat{t} \leq \Delta t$.*[1]

Given SEGs of length T, we aim to predict edges (user-item interactions) to be formed in $\mathcal{G}_{T+1}$. Throughout the paper, we denote scalars by either lowercase or uppercase letters, vectors by boldfaced lowercase letters, and matrices by boldfaced uppercase letters. Important notations are summarized in Table 1.

3 PROPOSED MODEL

3.1 Spectral Convolution

Inspired by the recent success of graph convolution methods [8], a recently proposed method [15], SpectralCF, extends the idea of spectral convolutions to the task of collaborative filtering. SpectralCF shows a great ability to capture preference patterns of users from the spectral domain of a user-item bipartite graph, and therefore achieves state-of-the-art performance.

Specifically, given a user-item bipartite graph $\mathcal{G}$ and its graph laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}$ and $\mathbf{A}$ denote the degree matrix and adjacent matrix of $\mathcal{G}$, respectively, the spectral convolution operation is defined as: $\begin{bmatrix} \tilde{\mathbf{h}}^u \\ \tilde{\mathbf{h}}^i \end{bmatrix} = \mathbf{U} f_\theta(\mathbf{\Lambda}) \mathbf{U}^T \begin{bmatrix} \mathbf{h}^u \\ \mathbf{h}^i \end{bmatrix}$, where $\mathbf{U} \in \mathbb{R}^{(n_u+n_i)\times(n_u+n_i)}$ and $\mathbf{\Lambda} \in \mathbb{R}^{(n_u+n_i)\times(n_u+n_i)}$ are eigenvectors and eigenvalues of $\mathbf{L}$, respectively; $f_\theta(\mathbf{\Lambda})$ is a convolutional filtering function placed on eigenvalues; $\mathbf{h}^u \in \mathbb{R}^{(n_u\times 1)}$ and $\mathbf{h}^i \in \mathbb{R}^{(n_i\times 1)}$ respectively denote state vectors of users and items, and $\tilde{\mathbf{h}}^u \in \mathbb{R}^{(n_u\times 1)}$ and $\tilde{\mathbf{h}}^i \in \mathbb{R}^{(n_i\times 1)}$ represent new state vectors of users and items, respectively, learned from the spectral domain.

Nonetheless, the number of parameters in $f_\theta(\mathbf{\Lambda})$ is linear to the dimensionality of data, resulting in an unscalable model. To circumvent this issue, [15] utilizes a polynomial approximation to approximate $g_\Theta(\Lambda)$ as $g_\theta(\Lambda) \approx \sum_{p=0}^{P} \theta_p' \Lambda^p$. As a result, the spectral convolution is reformulated as $\begin{bmatrix} \tilde{\mathbf{H}}^u \\ \tilde{\mathbf{H}}^i \end{bmatrix} = (\mathbf{U}\mathbf{U}^\top + \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top) \begin{bmatrix} \mathbf{H}^u \\ \mathbf{H}^i \end{bmatrix} \Theta$, where $\mathbf{H}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{H}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ are F-dimensional row-vectors for users and items, respectively; and $\Theta \in \mathbb{R}^{(F\times F)}$ is a generalized convolutional filter with F channels and F filters.

However, as we aim to model *co-evovling patterns* from sequential graphs other than one static graph, computing the eigendecomposition of laplacians of multiple graphs would be prohibitively expensive. In order to adopt the aforementioned spectral convolution operation for modeling sequential graphs, we notice that, rather than the full eigen-decomposition, top-l eigenvectors and eigenvalues are sufficient to approximate $\mathbf{L}$ [1]. Thus, we adopt ARPACK [10], a most popular iterative eigensolver. Its complexity is $O((n_u + n_i)l^2 + el)$, where e stands for the number of edges, and linear $w.r.t.$ the graph size $(n_u + n_i)$. Due to the sparsity of our graphs, we have $e \ll n_u + n_i$. Given the top-l eigenvectors $\mathbf{U}_t^{(l)}$ and eigenvalues $\mathbf{\Lambda}_t^{(l)}$ of the t_{th} graph, we have:

$$\begin{bmatrix} \tilde{\mathbf{H}}_t^u \\ \tilde{\mathbf{H}}_t^i \end{bmatrix} = (\mathbf{U}_t^{(l)}\mathbf{U}_t^{(l)\top} + \mathbf{U}_t^{(l)}\mathbf{\Lambda}_t^{(l)}\mathbf{U}_t^{(l)\top}) \begin{bmatrix} \mathbf{H}_{t-1}^u \\ \mathbf{H}_{t-1}^i \end{bmatrix} \Theta, \qquad (1)$$

where $\mathbf{H}_{t-1}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{H}_{t-1}^i \in \mathbb{R}^{(n_u+n_i)\times F}$ are state matrices of users and items from the previous time step $t-1$; $\tilde{\mathbf{H}}_t^u$ and $\tilde{\mathbf{H}}_t^i$ are learned by convolving $\begin{bmatrix} \mathbf{H}_{t-1}^u \\ \mathbf{H}_{t-1}^i \end{bmatrix}$ on the current graph $\mathcal{G}_t$. As such, $\tilde{\mathbf{H}}_t^u$ and $\tilde{\mathbf{H}}_t^i$ captures the evolving patterns by integrating information from the previous step $(t-1)$ with newly formed connections of the current step t. Hereafter, we denote the spectral convolution operation in Eq. 1 as a function: $Conv(\begin{bmatrix} \mathbf{H}_{t-1}^u \\ \mathbf{H}_{t-1}^i \end{bmatrix}, \mathcal{G}_t; \Theta)$, parameterized by a convolutional filter Θ.

3.2 Gated Spectral Units

Recall that our spectral convolution $Conv(\begin{bmatrix} \mathbf{H}_{t-1}^u \\ \mathbf{H}_{t-1}^i \end{bmatrix}, \mathcal{G}_t; \Theta)$ is capable of capturing the patterns co-evolving from the previous graph to the current graph. It is an natural idea to introduce gated mechanisms [2] into our spectral convolution to capture co-evolving patterns from a sequence of graphs. Therefore, we present Gated Spectral Units (GSUs), which are capable of learning the *co-evovling patterns* from sequential graphs.

In GSUs, the update gate $\mathbf{Z}_t \in \mathbb{R}^{(n_u+n_i)\times F}$ convolves the historical state matrices on the current graph to decide how much the

[1]In the case that $\mathcal{E}_t$ is empty, we remove $\mathcal{G}_t$ from $\mathcal{G}$.

unit updates its state matrices. It is computed by:

$$\mathbf{Z}_t = \sigma(Conv(\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{W}_z) + \mathbf{b}_z), \tag{2}$$

where $\mathbf{W}_z \in \mathbb{R}^{(n_u+n_i)\times F}$, $\mathbf{b}_z \in \mathbb{R}^{(n_u+n_i)\times 1}$, and σ denotes the sigmoid function. A candidate gate generates a candidate state matrices by resetting the previous $\mathbf{H}^u_{t-1}$ and $\mathbf{H}^i_{t-1}$, and convolving them on $\mathcal{G}_t$ as:

$$\begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \hat{\mathbf{H}}^i_t \end{bmatrix} = \tanh(Conv(\mathbf{R}_t \odot \begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{W}_h) + \mathbf{b}_h), \tag{3}$$

where $\mathbf{W}_h \in \mathbb{R}^{(n_u+n_i)\times F}$, $\mathbf{b}_h \in \mathbb{R}^{(n_u+n_i)\times 1}$, and the reset gate $\mathbf{R}_t \in \mathbb{R}^{(n_u+n_i)\times F}$ is similar to the update gate as below:

$$\mathbf{R}_t = \sigma(Conv(\mathbf{H}^u_{t-1}, \mathbf{H}^i_{t-1}, \mathcal{G}_t; \mathbf{W}_r) + \mathbf{b}_r), \tag{4}$$

where $\mathbf{W}_r \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{b}_r \in \mathbb{R}^{(n_u+n_i)\times 1}$. Finally, the output of GSUs at time t is a linear interpolation between the previous state matrices $\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix} \in \mathbb{R}^{(n_u+n_i)\times F}$ and the candidate $\begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \hat{\mathbf{H}}^i_t \end{bmatrix} \in \mathbb{R}^{(n_u+n_i)\times F}$ as below:

$$\begin{bmatrix} \mathbf{H}^u_t \\ \mathbf{H}^i_t \end{bmatrix} = \mathbf{Z}_t \odot \begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix} + (1 - \mathbf{Z}_t) \odot \begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \hat{\mathbf{H}}^i_t \end{bmatrix}), \tag{5}$$

where $\odot$ denotes the element-wise multiplication.

Overall, GSUs take the current graph $\mathcal{G}_t$ and previous $\mathbf{H}^u_{t-1}$ and $\mathbf{H}^i_{t-1}$ as inputs, and output $\mathbf{H}^u_t$ and $\mathbf{H}^i_t$ for the current time step t. Thus, given the initial state matrices, $\mathbf{H}^u_0$ and $\mathbf{H}^i_0$, which are randomly initialized as trainable parameters, GSUs are able to recurrently process a sequence of graphs, and output state matrices of users and items of the last step, which summarize the *co-evovling patterns* within the sequence.

3.3 Optimization and Prediction

Given SEGs of length K generated from the training data, we randomly sample a batch of SEGs of length $T+1$ ($T+1 \ll K$) for training. For each SEGs of length $T+1$, we feed the first T graphs into GSUs to obtain $\mathbf{H}^u_T$ and $\mathbf{H}^i_T$. And, the score of an edge $(i,j) \in \mathcal{E}_{T+1}$ at step $T+1$ can be calculated as $\mathbf{H}^u_T(i,:)^\top \mathbf{H}^i_T(j,:)$, where $\mathbf{H}^u_T(i,:)$ and $\mathbf{H}^i_T(j,:)$ denote the i_{th} and j_{th} row of $\mathbf{H}^u_T$ and $\mathbf{H}^i_T$, respectively. We optimize the parameters of GSUs by minimizing the loss as:

$$\begin{aligned}\mathcal{L} = \quad & -\sum_{\substack{(i,j)\in\mathcal{E}_{T+1} \\ j'\in\mathcal{I}^-_i}} \ln \sigma(\mathbf{H}^u_T(i,:)^\top \mathbf{H}^i_T(j,:) - \mathbf{H}^u_T(i,:)^\top \mathbf{H}^i_T(j',:)) \\ & + \lambda(||\mathbf{H}^u_T||^2_F + ||\mathbf{H}^i_T||^2_F),\end{aligned} \tag{6}$$

where λ is an regularization term. Eq. 6 seeks to maximize the difference between the scores of an existing edge $(i,j) \in \mathcal{E}_{T+1}$ and a non-existing edge (i,j'), where j' is sampled from $\mathcal{I}^-_i$.

For evaluation, the last T graphs of SEGs of length K are taken into GSUs to attain $\mathbf{H}^u_T$ and $\mathbf{H}^i_T$. The final item recommendation for a user i is given by ranking the score $\mathbf{H}^u_T(i,:)^\top \mathbf{H}^i_T(j,:)$ in a descending order.

4 EXPERIMENTS

In this section we conduct experiments to answer the following research questions:

- **RQ1:** Are the *co-evolving patterns* being effectively captured?
- **RQ2:** How do the *co-evovling patterns* work for handling the *cold-start* problem?

4.1 Datasets

In our experiments, we use three publicly available timestamped datasets: (1) **ML-1M**: [3] MovieLens-1M contains 1,000,209 ratings, 6,014 users and 3,706 movies; (2) **ADM** [11]: Amazon Digital Music 5-core includes 4,731 users, 2,420 video games; (3) **AIV**: Amazon Instant Videos 5-core is collected by [11] and consists of 4,818 users and 1,685 items.

As in [5], we transform datasets with explicit ratings into implicit data by regarding rating of 5 as positive feedback and all others as negative ones. For each dataset, we select the most recent item of each user for testing and the second most recent one for validation. All remaining items will be used for training. To create SEGs to capture *co-evovling patterns*, we set the time interval Δ as 1 day for *ML-1M* and *AIV*, and 7 for *ADM* to reduce the numder of graphs. As a result, we attain the SEGs of length (K) 977, 754, and 1,472 for the dataset of *ML-1M*, *ADM*, and *AIV*, respectively.

4.2 Experimental Settings

Evaluation Protocols. We evaluate all models in two metrics: Hit Ratio@10 (HR@10) and NDCG@10. HR@10 measures the fraction of relevant items at top-10 recommendations out of all relevant items, while NDCG@10 evaluates their ranking performance. We follow a similar strategy as in [5] to avoid heavy computation on evaluating all user-item pairs. For each user i, we randomly sample 999 negative items, and rank these items with the ground-truth item. Based on the rankings of these 1,000 items, HR@10 and NDCG@10 can be evaluated.

Comparative Models. We compare GSUs with six state-of-the-art algorithms. They can be categorized into two groups: (1) Non-sequential Models: **BPR** [12] and **SpectralCF** [15][2]; (2) Sequential Models: **FPMC** [13], **TransRec** [4], **GRU4Rec** [6][3] and **Caser** [14][4]. The first group is added to validate the usefulness of sequential recommendation models, and the second group is for demonstrating the advantage of modeling *co-evovling patterns*.

Parameter Settings. For all methods, we search the latent dimensions from $\{8, 16, 32, 64\}$. The $\mathcal{L}_2$ regularization term is selected from $\{0.0001, 0.001, 0.01, 0.1\}$ for BPR, SpectralCF, FPMC, TransRec and GSUs. We tune all hyper-parameters using the validation set. For GRU4Rec and Caser, we use the parameter settings as suggested in the original papers. The *Adam* optimizer [7] with the learning rate of 0.001 is adopted, and l in Eq. 1 and T are empirically set to 6 and 10, respectively, for GSUs.

4.3 Performance Comparison (RQ1)

In this section we compare GSUs with six state-of-the art methods to answer **RQ1**. Table 2 shows the performance comparison in terms of HR@10 and NDCG@10. Overall, GSUs improves the best comparative method by **27.9%** and **53.4%** in terms of HR@10 and NDCG@10, respectively, averaging on all three datasets. This experiment reveals two interesting observations:

[2]https://github.com/lzheng21/SpectralCF
[3]https://github.com/hidasib/GRU4Rec
[4]https://github.com/graytowne/caser_pytorch

Table 2: Performance comparison in HR@10 and NDCG@10. The best and second best method are boldfaced and underlined, respectively. * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, compared to the best competitor.

Dataset	Metric	BPR	SpectralCF	FPMC	TransRec	GRU4Rec	Caser	GSUs	*GSUs vs. best*
ML-1M	HR@10	0.061	0.081	0.092	0.099	0.102	<u>0.103</u>	**0.131****	27.2%
	NDCG@10	0.025	0.031	0.039	0.041	0.046	<u>0.045</u>	**0.061****	35.6%
ADM	HR@10	0.022	0.031	0.041	0.043	<u>0.051</u>	0.048	**0.065****	27.4%
	NDCG@10	0.011	0.018	0.019	0.021	<u>0.024</u>	0.022	**0.034***	41.7%
AIV	HR@10	0.072	0.088	0.096	0.010	0.111	<u>0.117</u>	**0.151***	29.1%
	NDCG@10	0.022	0.031	0.034	0.037	0.039	<u>0.041</u>	**0.075***	82.9%

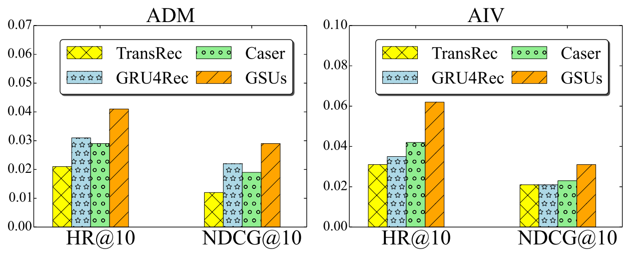

Figure 2: Performance comparison in HR@10 and NDCG@10 under a sparse setting, where each user is associated with only one user-item interaction for training.

- Non-sequential methods underperform sequential methods, indicating the benefits of modeling the short- and long- term dynamics in users' actions.
- Regardless of the data sets and the evaluation metrics, the proposed GSUs always achieve the best performance. This shows that by leveraging the power of *co-evovling patterns*, GSUs can better predict users' future actions.

4.4 Recommending for Cold-start Users (RQ2)

The *cold-start* problem is one of the most challenging issues for RS. It happens when a user interacted with very few number of items, causing a difficulty to understand the user's preferences. We are interested in if *co-evovling patterns* are helpful for alleviating the *cold-start* problem. As such, we conduct experiments under an extremely sparse setting, where we only use the first interaction of each user for training, and the second and third one for validation and test, respectively. All others are discarded. Consequently, we obtain SEGs of length 458 and 1, 158 for the datasets of *ADM* and *AIV*, respectively. Fig. 2 illustrates the performance comparison under the sparse setting. In *ADM*, GSUs outperform the best comparative method, GRU4Rec, by **32.3%** and **31.8%** in HR@10 and NDCG@10, respectively. In *AIV*, GSUs beat the best performing competitor, Caser, by **47.6%** and **34.8%** in HR@10 and NDCG@10, respectively. It is validated that, benefiting from the ability of capturing *co-evolving patterns*, GSUs can better handle cold-start users than state-of-the-art comparative methods.

5 CONCLUSIONS

Despite the promising results achieved by recent sequential methods, most of them fail to leverage the *co-evovling patterns*, and such patterns are affluent in users actions and beneficial for reasoning over intricate user-item relationships.

In this work, in order to power RS with the ability to capture *co-evolving patterns*, we first formulate the dynamic user-item bipartite graph into *Sequential Evovling Graphs* (SEGs) (see Definition 2.1). Then, in order to utilize co-evolutionary patterns from SEGs, we propose Gated Spectral Units (GSUs). GSUs incorporate gated mechanisms into a spectral convolution. In this way, GSUs are able to learn from sequential graphs and capture the *co-evovling patterns* from spectral domains. In experiments, we demonstrate the usefulness of leveraging *co-evolving patterns* by comparing GSUs with six state-of-the-art comparative methods. Overall, averaging on all three datasets, GSUs achieve **27.9%** and **53.4%** improvements over the best performing competitor in terms of HR@10 and NDCG@10, respectively. Additionally, we evaluate GSUs and three comparative methods in an extremely sparse setting, where each user is associated with only one user-item interaction. In the sparse setting, GSUs show its superior ability for handling cold-start users.

REFERENCES

[1] Xinlei Chen and Deng Cai. 2011. Large scale spectral clustering with landmark-based representation.. In *AAAI*, Vol. 5. 14.
[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[3] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.
[4] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 161–169.
[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 173–182. https://doi.org/10.1145/3038912.3052569
[6] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[7] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[8] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[9] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 447–456.
[10] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Vol. 6. Siam.
[11] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. *arXiv preprint arXiv:1506.04757* (2015).
[12] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
[13] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.
[14] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 565–573.
[15] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 311–319.