SIGformer: Sign-aware Graph Transformer for Recommendation

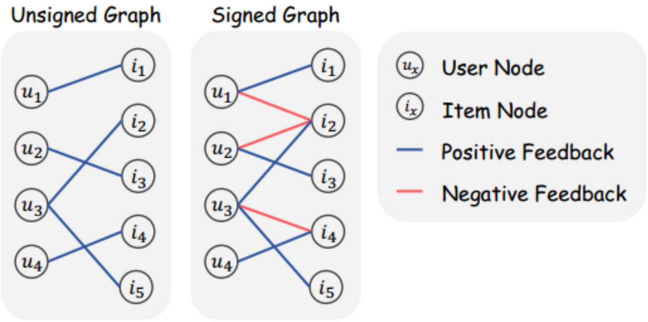
Lu Li

June 7, 2024

Introduction

Challenge

- They process positive and negative feedback separately, which fails to holistically leverage the collaborative information within the signed graph.
- They rely on MLPs or GNNs for information extraction from negative feedback, which may not be effective.



Contribution

Method Contribution

- introduce SIGformer, a new method that employs the **transformer architecture** to sign-aware graph-based recommendation.
- propose **two innovative sign-aware positional encodings**, derived from the perspectives of signed graph spectrum and paths, which fully exploit the sign-aware collaborative information.

Sign-aware Spectral Encoding (SSE)

To integrate the structure of the entire signed graph, we propose to utilize the node spectral representation on the signed graph.

Sign-aware Path Encoding (SPE)

To further capture collaborative relations among users and items, we focus on the patterns of paths within the signed graph.

Transformer

In the self-attention module, the input features $\mathbf{X} \in \mathbb{R}^{n \times d}$ are projected to the corresponding **query Q, key K, and value V**, and then calculated via attention with:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$
$$\mathrm{Attn}(\mathbf{X}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right)\mathbf{V} \tag{1}$$

where $\mathbf{W}_Q \in \mathbb{R}^{d \times d_K}, \mathbf{W}_K \in \mathbb{R}^{d \times d_K}, \mathbf{W}_V \in \mathbb{R}^{d \times d_V}$ denote the projected matrices of query, key and value respectively.

Methodology

Embedding Module:

$$\mathbf{E}^{(0)} = [\underbrace{\mathbf{e}_{u_1}^{(0)}, \cdots, \mathbf{e}_{u_n}^{(0)}}_{\text{user embeddings}}, \underbrace{\mathbf{e}_{i_1}^{(0)}, \cdots, \mathbf{e}_{i_m}^{(0)}}_{\text{item embeddings}}]^T.$$

Sign-aware Transformer Module:

$$\mathbf{Q}^{(l)} = \mathbf{K}^{(l)} = \mathbf{V}^{(l)} = \mathbf{E}^{(l-1)}$$

$$\mathbf{E}^{(l)} = \frac{1}{2}(\mathrm{softmax}(\frac{\mathbf{Q}^{(l)}(\mathbf{K}^{(l)})^T}{\sqrt{d}} + \mathbf{P}_s^{(l)}) + \mathrm{softmax}(\mathbf{P}_p^{(l)}))\mathbf{V}^{(l)}$$

Prediction Module:

$$\mathbf{E} = \frac{1}{L+1} \sum_{0 \le l \le L} \mathbf{E}^{(l)}$$

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i$$

SSE

We begin by combining the Laplacians of the positive and negative graphs as follows:

$$\mathbf{L} = \frac{1}{1-\alpha}(\mathbf{L}^{+} - \alpha \mathbf{L}^{-})$$

The Laplacian eigenvectors of the signed graph are:

$$\mathbf{L} = \mathbf{H}^{T} \mathbf{\Lambda} \mathbf{H}, \quad \mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{n+m}]^{T}$$

The eigenvectors of the d_h smallest eigenvalues denoted $\tilde{\mathrm{H}}$,are used for encoding node relations in the signed graph:

$$\mathbf{P}_{s}^{(l)} = \theta^{(l)} \tilde{\mathbf{H}}^{T} \tilde{\mathbf{H}}, \quad \tilde{\mathbf{H}} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{d_h}]^{T}$$

SSE

Connecting with Low-pass Filtering The vector v can be expressed by a combination of the basis H:

$$\mathbf{v} = \sum_{1 \le k \le n+m} \varepsilon_k \mathbf{h}_k$$

$$\mathbf{P}_s^{(l)} \mathbf{v} = (\sum_{1 \le k \le d_h} \theta^{(l)} \mathbf{h}_k \mathbf{h}_k{}^T)(\sum_{1 \le k \le n+m} \varepsilon_k \mathbf{h}_k) = \theta^{(l)} \sum_{1 \le k \le d_h} \varepsilon_k \mathbf{h}_k$$

lemma1

The low-frequency components $(h_1, h_2, \cdots, h_{d_h})$ optimizes the following objective function:

$$[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{d_h}] = \arg \min_{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{d_h}} \sum_{1 \le k \le d_h} \Big(\sum_{(u,i) \in \mathcal{E}^+} (\frac{\mathbf{z}_{ku}}{\sqrt{d_u^+}} - \frac{\mathbf{z}_{ki}}{\sqrt{d_i^+}})^2 -$$

$$\alpha \sum_{(u,i) \in \mathcal{E}^-} (\frac{\mathbf{z}_{ku}}{\sqrt{d_u^-}} - \frac{\mathbf{z}_{ki}}{\sqrt{d_i^-}})^2 \Big)$$

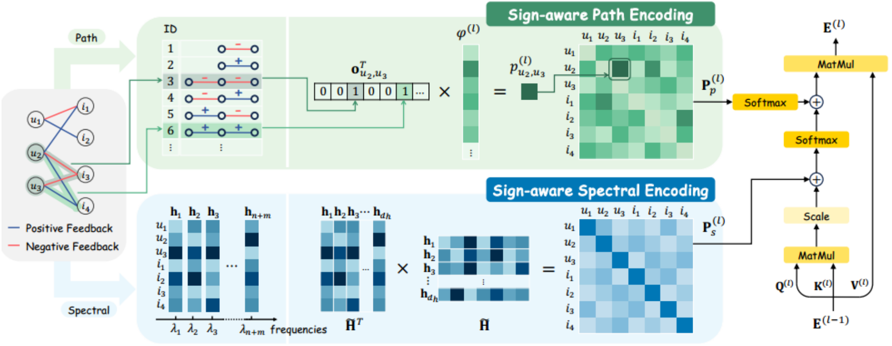
SPE

Our fundamental intuition is that **different path types indicate varying levels of affinity between the nodes they connect.**
We integrate this rich path information into the transformer architecture to capture nodes' affinity:

$$p_{vw}^{(l)} = \mathbf{o}_{vw}^{T} \varphi^{(l)}$$

For convenience, we aggregate $p_{vw}^{(l)}$ for all node pairs into a matrix, termed as sign-aware path encoding $\mathbf{P}_p^{(l)}$.

illustration



Implementation details

Specifically, we utilize a **random walk** strategy on the signed graph to pick up nodes for aggregation and concurrently record the walked path for computing $\mathbf{P}_p^{(l)}$.
For each node $v \in V$, we perform a non-cyclic random walk of length L_p starting from each neighbor of v to sample a set of nodes S_v associated with the trajectory type.

$$\mathbf{e}_v^{(l)} = \frac{1}{2} \sum_{w \in S_v} \left(\mathrm{softmax}\Big(\frac{(\mathbf{e}_v^{(l-1)})^T \mathbf{e}_w^{(l-1)}}{\sqrt{d}} + \theta^{(l)} m_{vw} \Big) \right.$$

$$\left. + \mathrm{softmax}(\varphi_{t_{vw}}) \mathbf{e}_w^{(l-1)} \right)$$

Implementation details

BPR loss is adopted for optimizing our SIGformer:

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{E}^+} \ln\sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \sum_{(u,i)\in\mathcal{E}^-} \ln\sigma\left(\beta(\hat{y}_{ui} - \hat{y}_{uj})\right)$$

β is a hyperparameter that balances the influence from the negative feedback.

Experiment

- RQ1: How does SIGformer perform compared with existing methods?
- RQ2: What are the impacts of the important components (e.g., , two positional encodings, negative interactions) on SIGformer?
- RQ3: How do the hyperparameters affect the model performance?
- RQ4: How do different path types capture node similarity?
- RQ5: How does the runtime of SIGformer compare with existing methods?

Experiment

RQ1: How does SIGformer perform compared with existing methods?

Table 2: Performance comparison between SIGformer and baselines. The best result is bolded and the runner-up is underlined. The mark '*' suggests the improvement is statistically significant with $p < 0.05$.

		Amazon-CDs		Amazon-Music		Epinions		KuaiRec		KuaiRand	
		Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
Unsigned Graph-based RS	LightGCN	0.1325	0.0781	0.2725	0.1601	0.0854	0.0510	0.0826	0.0499	0.1197	0.0588
	LightGCL	0.1040	0.0591	0.2921	0.1648	0.0864	0.0516	0.0848	0.0520	0.1291	0.0628
	XSimGCL	0.1346	0.0796	0.2848	0.1683	0.0887	0.0558	0.0863	0.0522	0.1293	0.0641
	GFormer	0.1366	0.0812	0.2807	0.1648	**0.0978**	**0.0602**	0.0864	0.0520	0.1083	0.0532
Sign-aware Graph-based RS	SiReN	0.1369	0.0801	0.2880	0.1725	0.0804	0.0492	0.0826	0.0473	0.1167	0.0571
	SiGRec	0.1092	0.0648	0.1591	0.0896	0.0738	0.0475	0.0497	0.0314	0.1266	0.0699
	PANE-GNN	0.1361	0.0810	0.2691	0.1605	0.0532	0.0301	0.0806	0.0514	0.1066	0.0522
Signed Graph Embedding Methods	SBGNN	0.0183	0.0100	0.0641	0.0325	0.0249	0.0143	0.0797	0.0469	0.0750	0.0361
	SLGNN	0.0283	0.0148	0.1498	0.0788	0.0585	0.0336	0.0865	0.0508	0.1082	0.0520
Graph Transformer	SGFormer	0.0492	0.0275	0.2402	0.1373	0.0588	0.0343	0.0840	0.0504	0.0883	0.0423
	SignGT	0.0231	0.0121	0.1283	0.0666	0.0521	0.0300	0.0861	0.0515	0.0927	0.0439
Our Method	SIGformer	**0.1412***	**0.0828***	**0.3091***	**0.1827***	0.0974	0.0585	**0.0908***	**0.0539***	**0.1494***	**0.0722***
		+3.09%	+1.96%	+5.81%	+5.87%	-0.41%	-2.77%	+5.05%	+3.32%	+15.61%	+3.33%

Experiment

RQ2: What are the impacts of the important components on SIGformer?

Table 3: The results of the ablation study, where positional encodings or negative interactions are removed respectively.

	Negative Interactions?	Spectral Encoding?	Path Encoding?	Amazon-CDs Recall	Amazon-CDs NDCG	Amazon-Music Recall	Amazon-Music NDCG	Epinions Recall	Epinions NDCG	KuaiRec Recall	KuaiRec NDCG	KuaiRand Recall	KuaiRand NDCG
SIGformer-w/o-Neg		✓	✓	0.1349	0.0775	0.2937	0.1738	0.0824	0.0477	0.0708	0.0433	0.1173	0.0545
SIGformer-w/o-En	✓			0.1355	0.0779	0.2932	0.1698	0.0894	0.0526	0.0728	0.0448	0.1413	0.0661
SIGformer-w/o-SPE	✓	✓		0.1380	0.0798	0.2988	0.1744	0.0959	0.0574	0.0862	0.0520	0.1471	0.0697
SIGformer-w/o-SSE	✓		✓	0.1381	0.0812	0.2947	0.1758	0.0945	0.0566	0.0866	0.0515	0.1457	0.0703
SIGformer	✓	✓	✓	**0.1412**	**0.0828**	**0.3091**	**0.1827**	**0.0974**	**0.0585**	**0.0908**	**0.0539**	**0.1494**	**0.0722**

Experiment

RQ3: How do the hyperparameters affect the model performance?

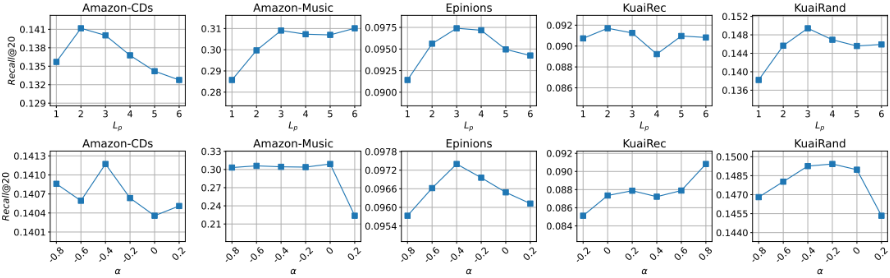


Figure 3: Performance in terms of $Recall@20$ **with different** K **and** α**.**

Experiment

RQ4: How do different path types capture node similarity?

(a) Top-5 values of φ

Path patterns	Values
+ + − −	1.704
+	1.661
− − +	1.658
+ − +	1.654
+ − −	1.654

(b) Bottom-5 values of φ

Path patterns	Values
− − − +	-1.303
− − + +	-1.226
− +	-1.225
− + − +	-1.210
− + + +	-1.116

Figure 4: The top-5 and bottom-5 values of the learned φ from KuaiRec.

Experiment

RQ5: How does the runtime of SIGformer compare with existing methods?

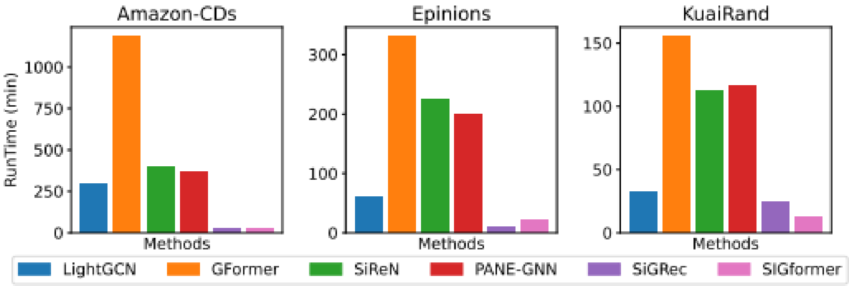


Figure 5: Runtime comparison of SIGformer with baselines.