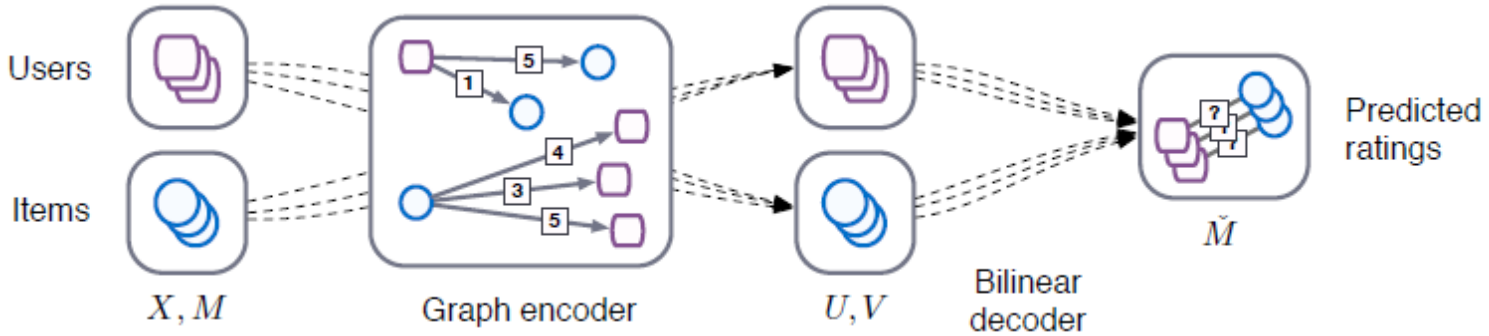
Graph Neural Network for User-Item Recommendation

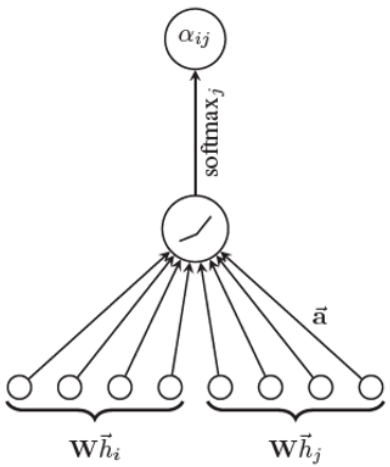
December 25, 2020

Abstract

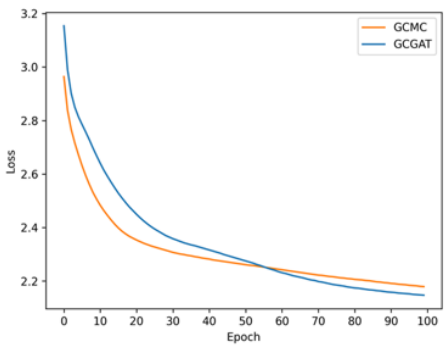
- Graph Convolutional Matrix Completion[KDD'18]

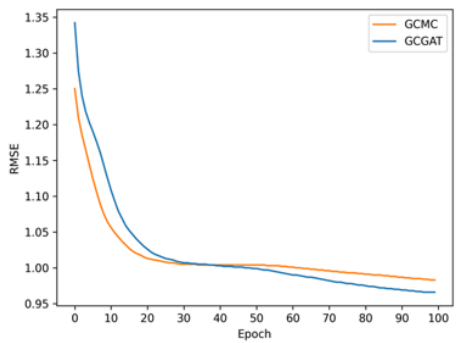


- Graph Attention Network[ICLR'18]



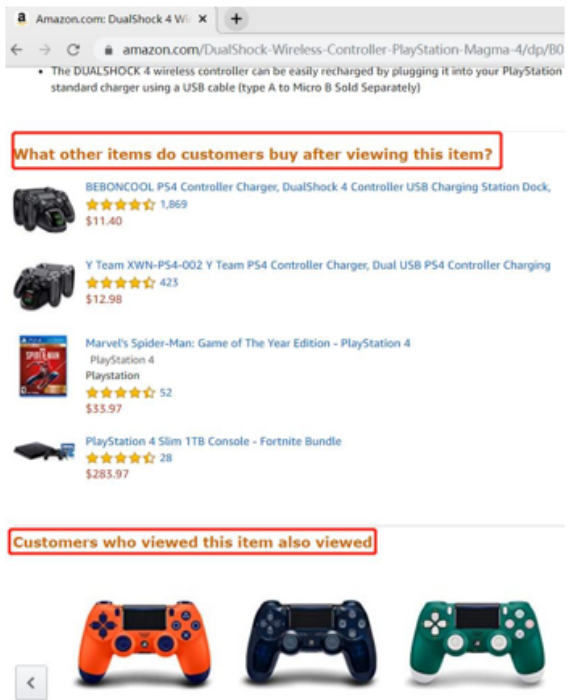
Model	RMSE
GCMC	0.983
GCGAT	**0.965**





Background

- Recommendation System

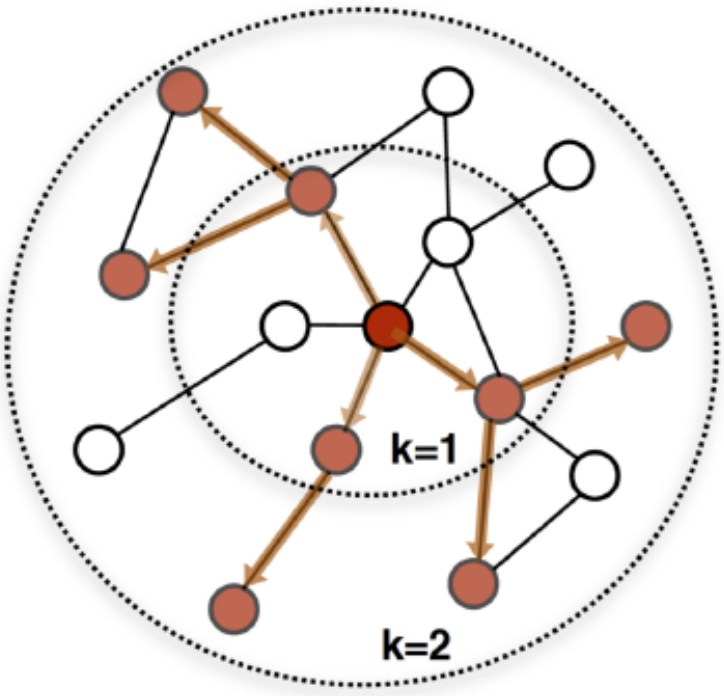


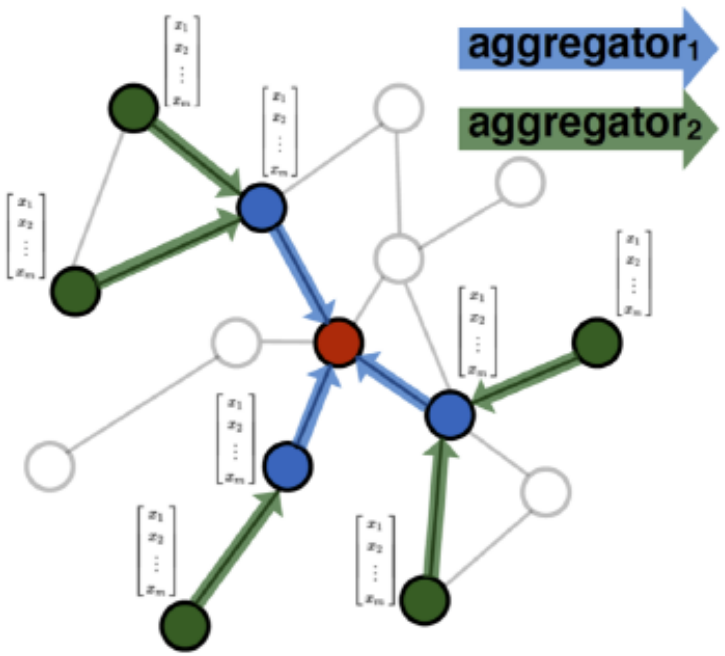
Background

- Movie Recommendation

Background

- Graph Neural Network



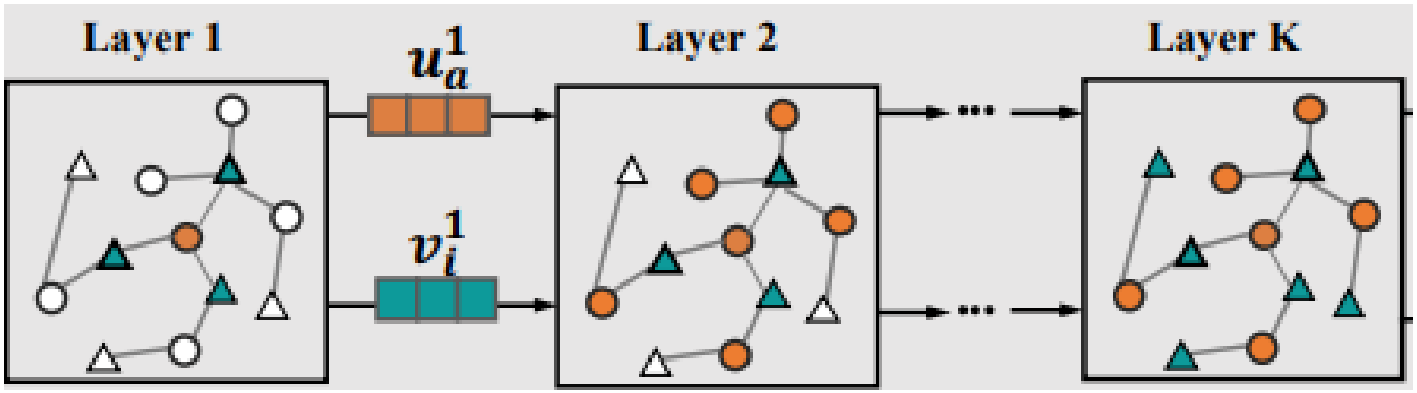


1

[1] Inductive Representation Learning on Large Graphs[NIPS'17]

Motivation

- Graph type data is common
- Cold-start challenge
  - how to recommend item for new user?



- Social influence

Input Graph

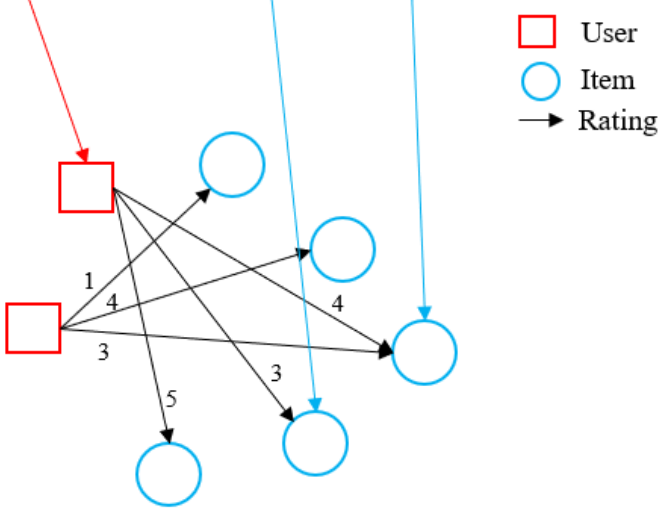
- Rating Matrix

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Input Graph

- user/item → nodes, rating → edges

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



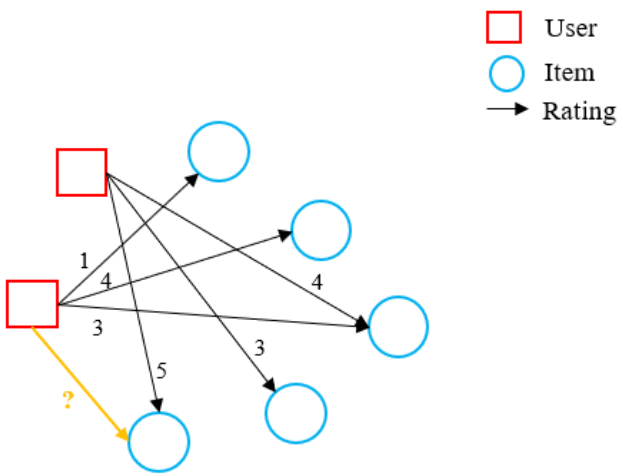
Input Graph

- Rating Matrix → Graph

Rating Matrix

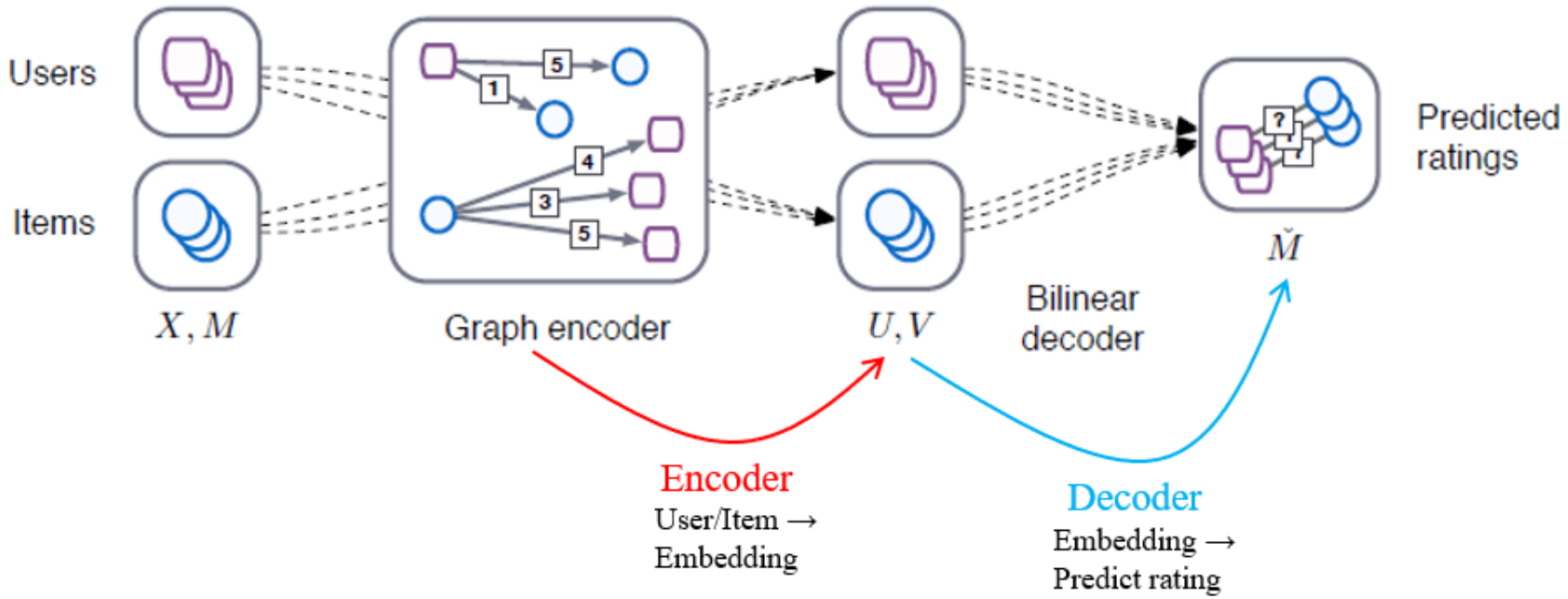
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Graph



Architecture

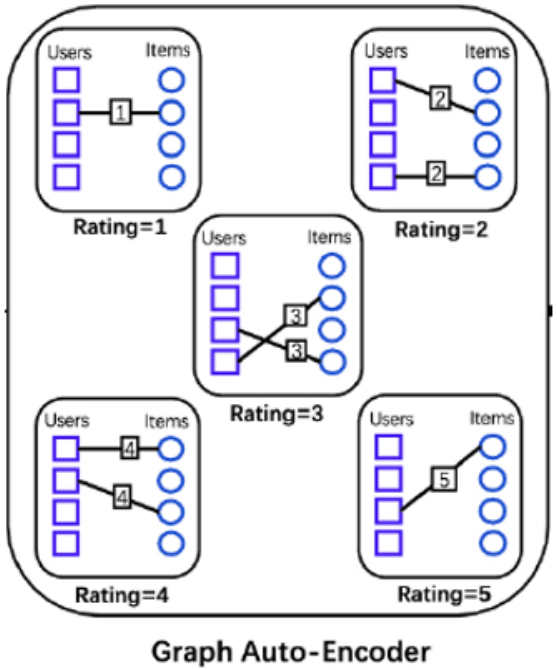
- Encoder-Decoder[2]

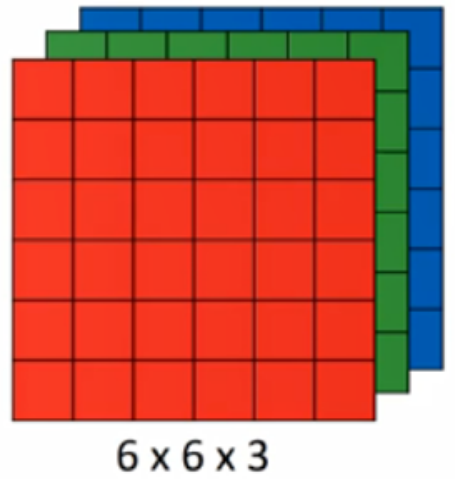


[2]Graph Convolutional Matrix Completion[KDD'18]

Architecture

- Encoder

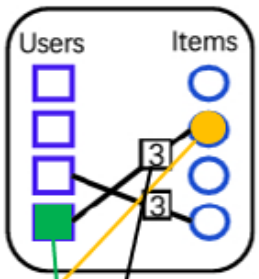




Rating ⟷ Channel

Architecture

- Encoder



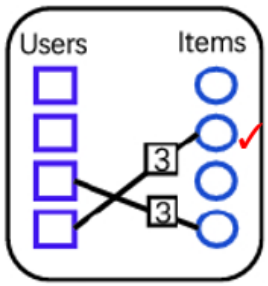
$$m_{v_j \to u_i, r} = W_r^T x_j, \tag{1}$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \to u_i, r}, \tag{2}$$

$$h_{u_i} = \sigma(\text{concat}(\{m_{u_i, r}\}_{r \in R})), \tag{3}$$

Architecture

- Encoder



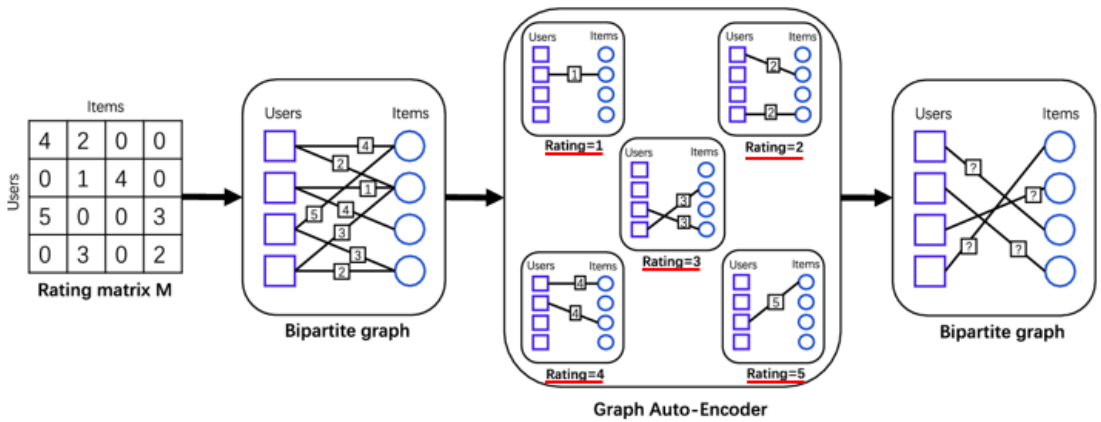
$$m_{v_j \to u_i, r} = W_r^T x_j, \tag{1}$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \to u_i, r}, \tag{2}$$

$$h_{u_i} = \sigma(\text{concat}(\{m_{u_i, r}\}_{r \in R})), \tag{3}$$

Architecture

- Encoder



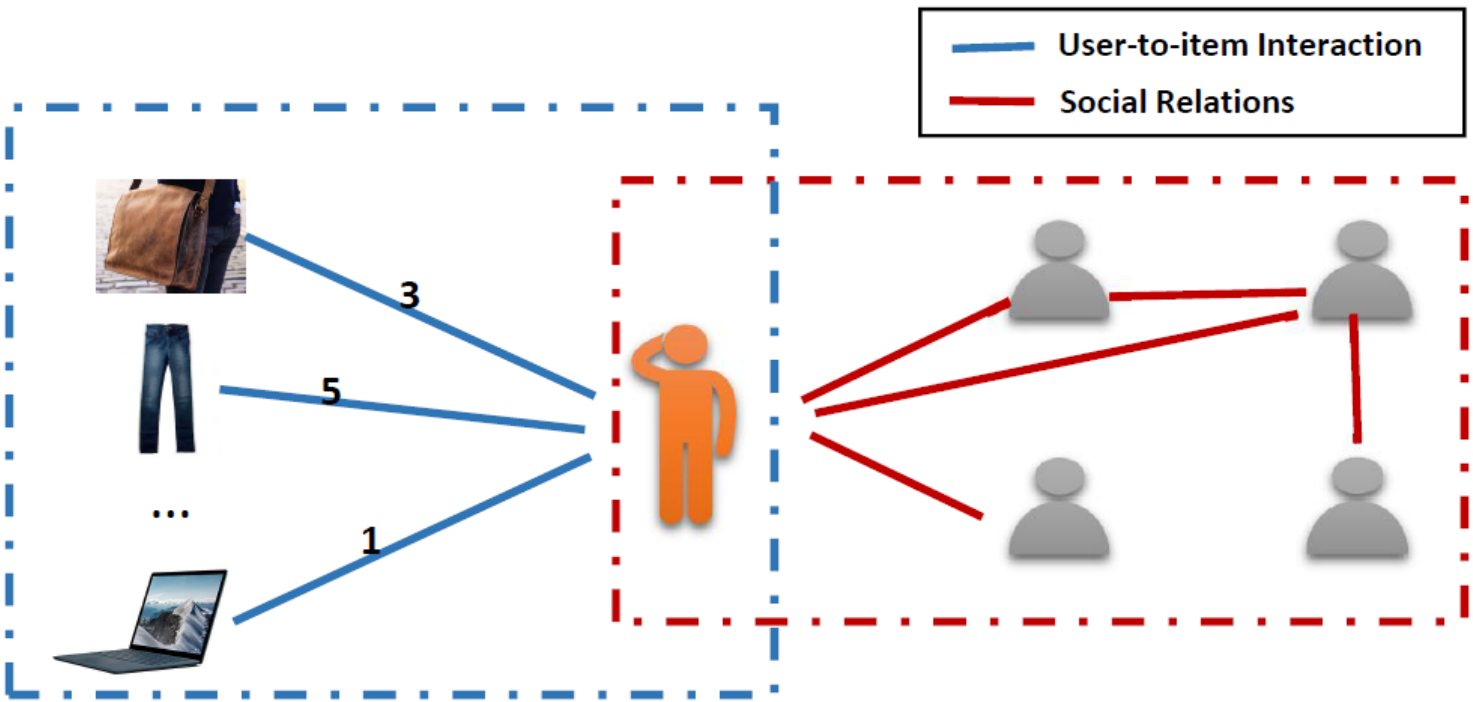
$$m_{v_j \to u_i, r} = \mathrm{W}_r^T x_j, \tag{1}$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \to u_i, r}, \tag{2}$$

$$h_{u_i} = \sigma(\mathrm{concat}(\{m_{u_i, r}\}_{r \in R})), \tag{3}$$

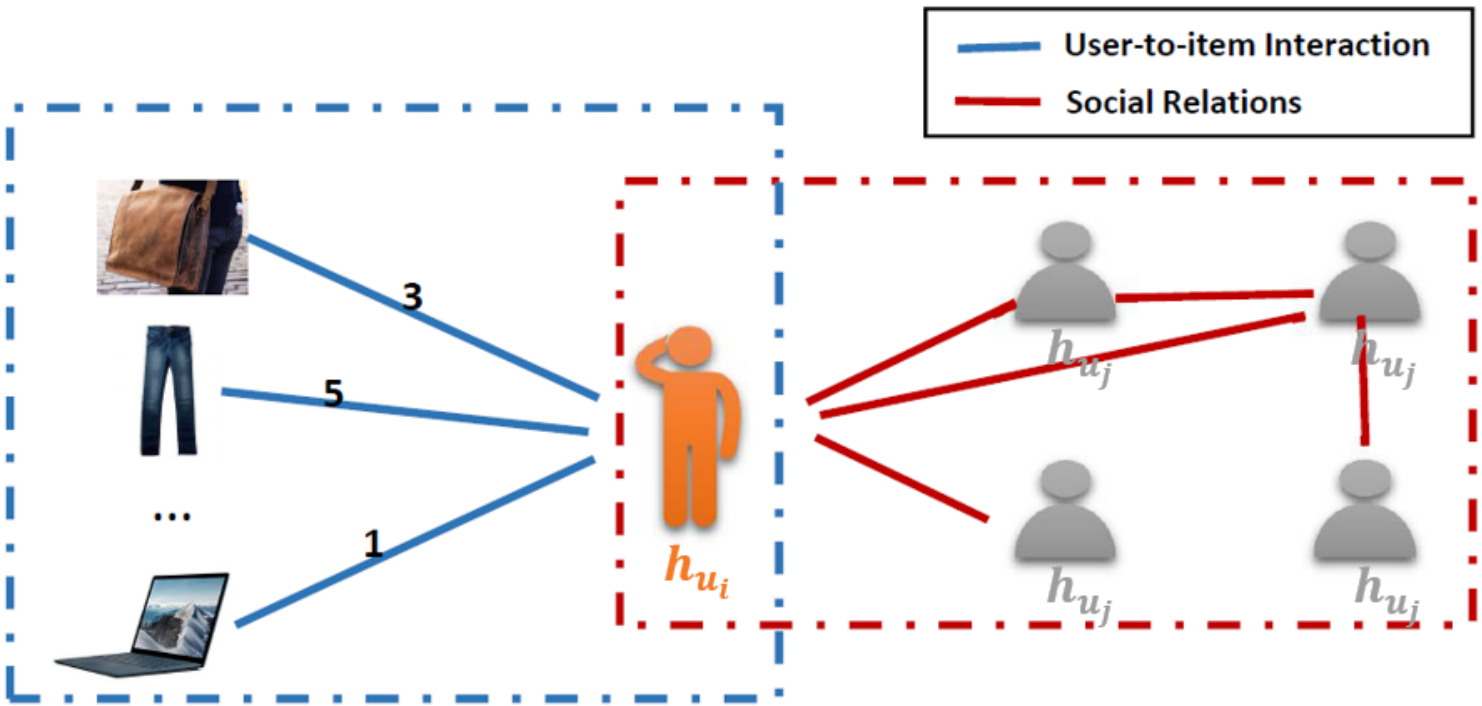
Analysis

- Limitation



Improvement

- Modification



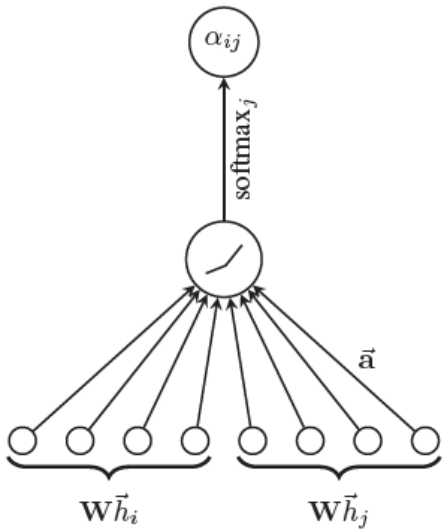
$$h_{u_i} = \sum_{j \in N_i} \alpha_{ij} h_{u_j},$$

GAT

- Can we simply learn a weight for each node in the graph?
  - Important node (e.g. with large degree) deserves large weight.
- Probably not.
- The importance of each node to each neighbor should be different.
- Goal: Specify arbitrary importance to different neighbors of each node in the graph.
- Idea: Compute embedding $\boldsymbol{h}_v^k$ of each node in the graph following an attention network.

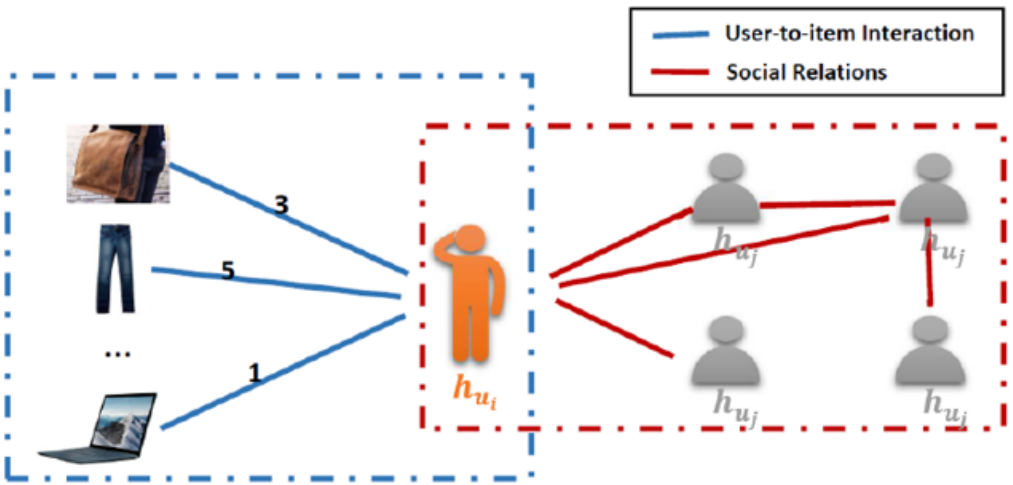
Improvement

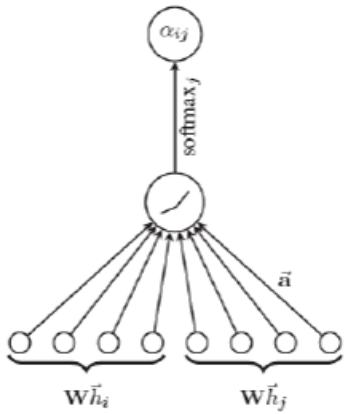
- Attention Mechanism



$$
\begin{aligned}
e_{ij} &= a(\mathrm{W}h_{u_i}, \mathrm{W}h_{u_j}), \\
&= \mathrm{LeakyReLU}(\overrightarrow{a}^T[\mathrm{W}h_{u_i}||\mathrm{W}h_{u_j}]) \\
\alpha_{ij} &= \mathrm{softmax}_j(e_{ij}) \\
&= \frac{\exp(e_{ij})}{\sum_{k\in N_i}\exp(e_{ik})}
\end{aligned}
$$

Improvement



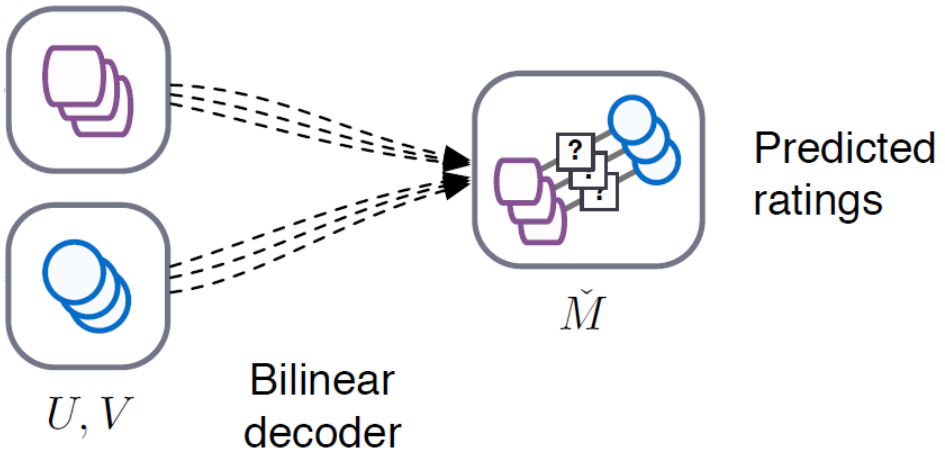


$$h_{u_i} = \sum_{j \in N_i} \alpha_{ij} h_{u_j},$$

$$\begin{aligned} e_{ij} &= a(\mathbf{W}h_{u_i}, \mathbf{W}h_{u_j}), \\ &= \text{LeakyReLU}(\overrightarrow{a}^T [\mathbf{W}h_{u_i} || \mathbf{W}h_{u_j}]) \\ \alpha_{ij} &= \text{softmax}_j(e_{ij}) \\ &= \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \end{aligned}$$

Architecture

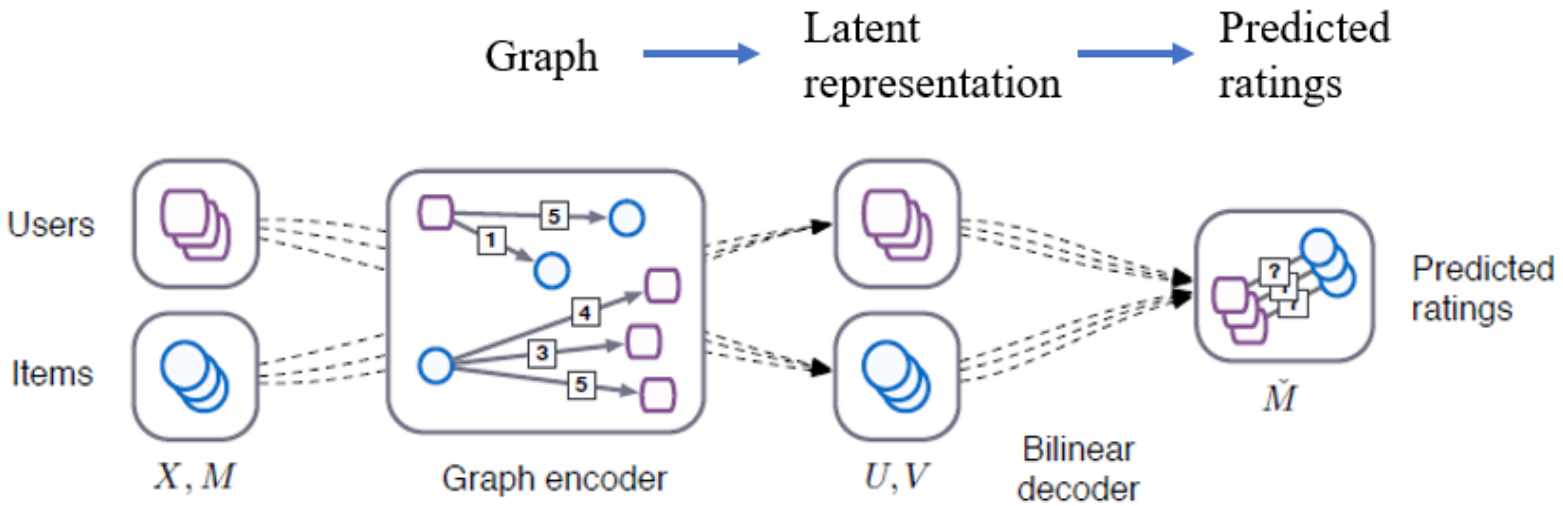
- Decoder



$$(\mathrm{P}_r)_{ij} = \frac{\exp(u_i^T \mathrm{Q}_r v_j)}{\sum_{s \in R} \exp(u_i^T \mathrm{Q}_s v_j)}$$

$$\hat{\mathrm{M}} = \sum_{r \in R} r \mathrm{P}_r,$$

Architecture



- Loss Function

$$L = -\sum_{(i,j)\in\Omega}\sum_{r=1}^{R} I[\mathrm{M}_{ij} = r]\log p(\hat{\mathrm{M}} = r),$$

Experiment

- Model
  - GCMC(original)
  - GCGAT(ours with attention mechanism)
- Dataset

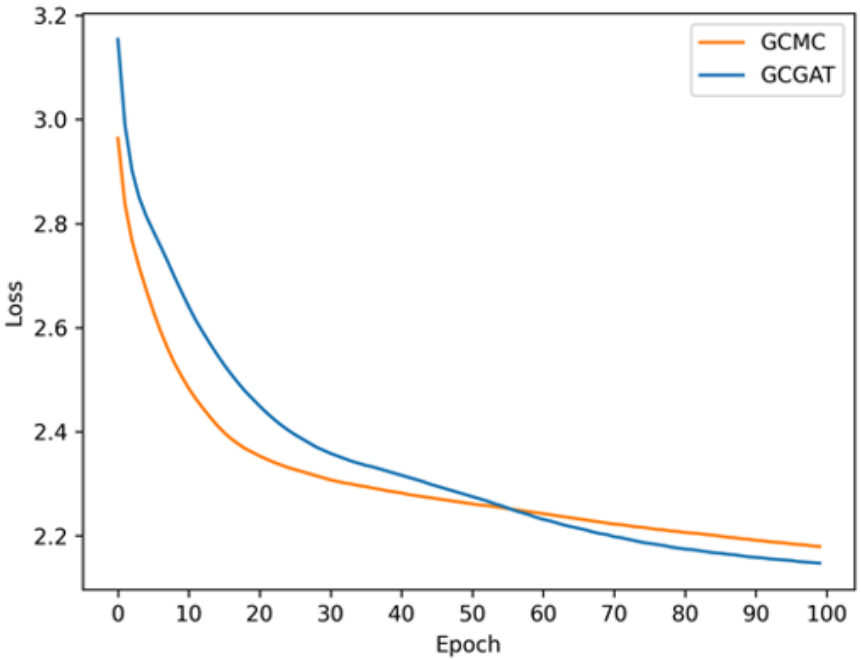
Dataset	Users	Items	Ratings	Rating Levels
ML-100K	943	1682	100,000	1,2,...,5

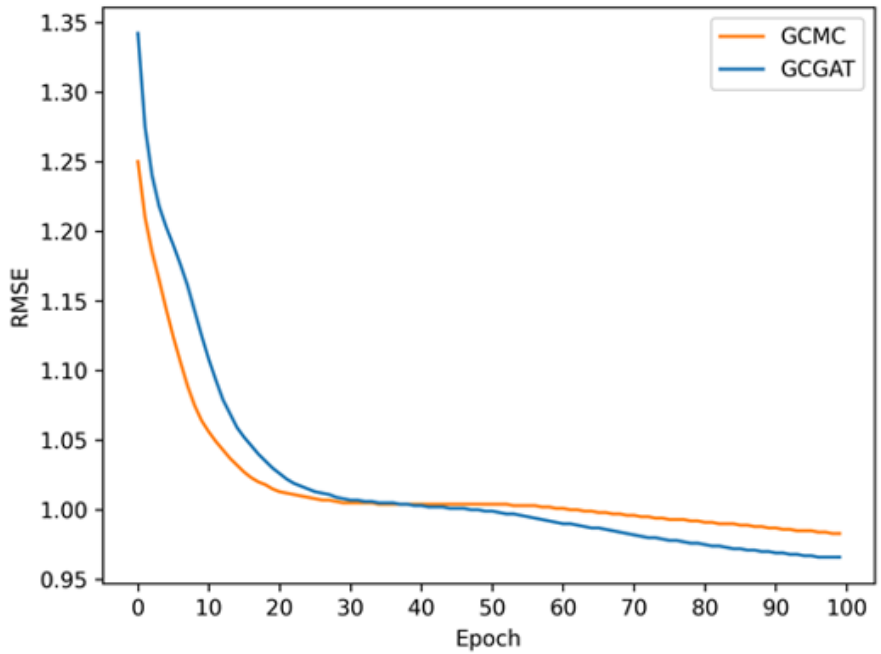
- Metrics

$$L_{\text{RMSE}} = \sqrt{\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \left((\hat{\mathrm{M}}_r)_{ij} - (\mathrm{M})_{ij} \right)}$$

Experiment

- Result

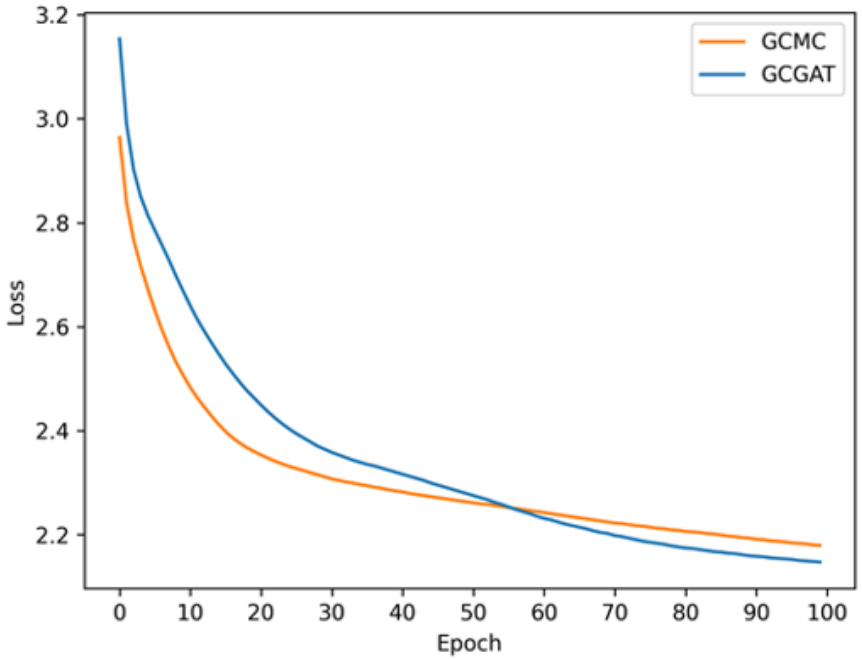


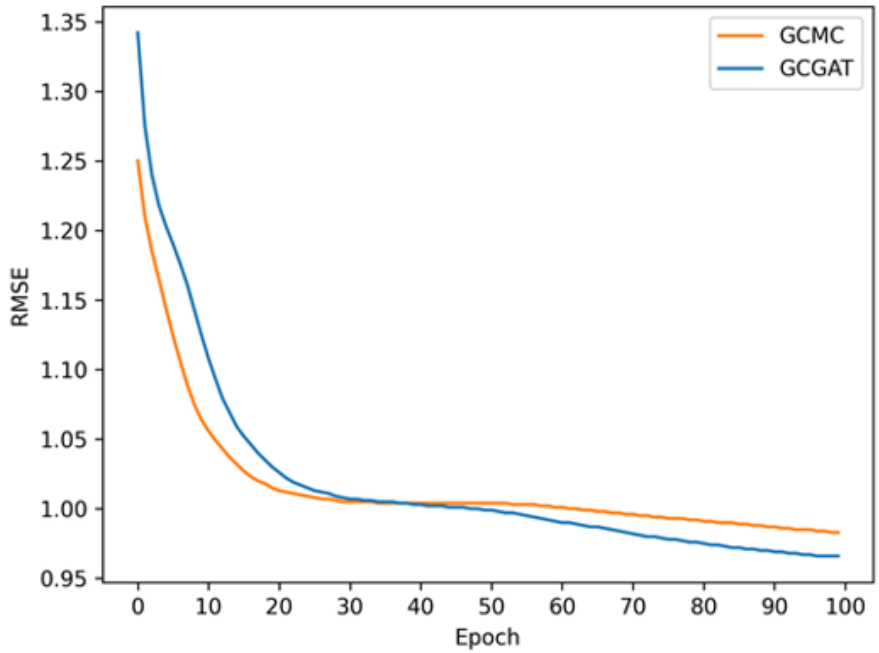


Model	RMSE
GCMC	0.983
GCGAT	**0.965**

Experiment

- Result





Model	RMSE
GCMC	0.983
GCGAT	**0.965**

https://github.com/BitHub00/GCGAT

Thank you!