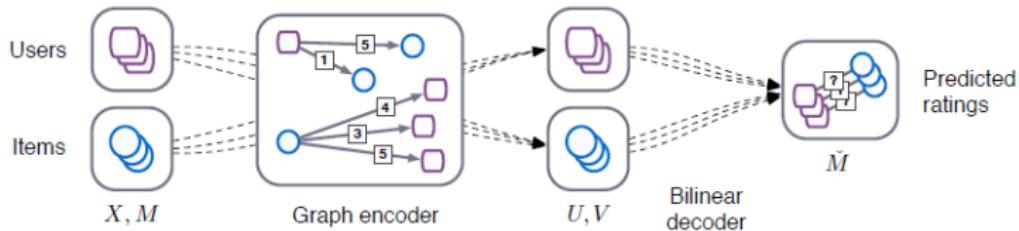


Graph Neural Network for User-Item Recommendation

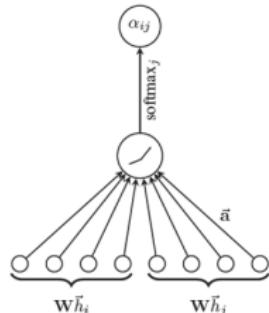
December 25, 2020

Abstract

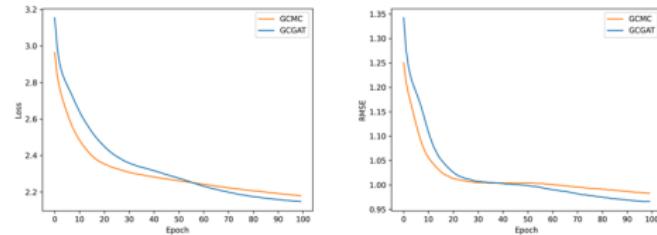
- Graph Convolutional Matrix Completion[KDD'18]



- Graph Attention Network[ICLR'18]



Model	RMSE
GCMC	0.983
GCGAT	0.965



Background

- Recommendation System



This screenshot shows a product page on Amazon.com for the "DualShock 4 Wireless Controller - PlayStation 4 (Magma)".

What other items do customers buy after viewing this item?

- BENODCOL PS4 Controller Charger, DualShock 4 Controller USB Charging Station Dock, \$11.40
- Y Team XWH-PS4-002 Y Team PS4 Controller Charger, Dual USB PS4 Controller Charging Station, \$12.99
- Marvel's Spider-Man: Game of the Year Edition - PlayStation 4, \$53.97
- PlayStation 4 Slim 1TB Console - Fortnite Bundle, \$283.87

Customers who viewed this item also viewed



Background

- Movie Recommendation



WONDER
WOMAN



HARRY
POTTER



GODZILLA



AVATAR



STAR WARS

TOM



BEN

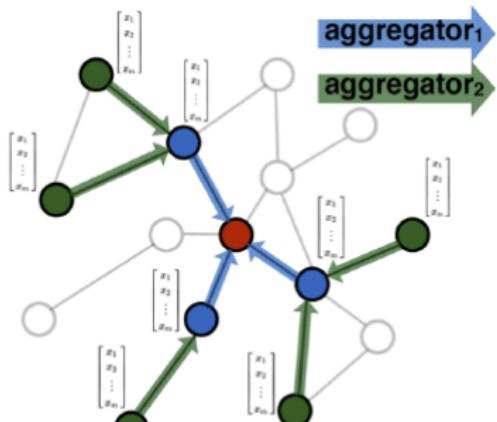
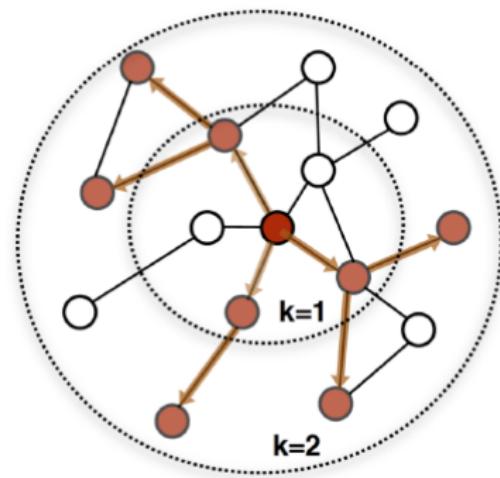


CAITLYN



Background

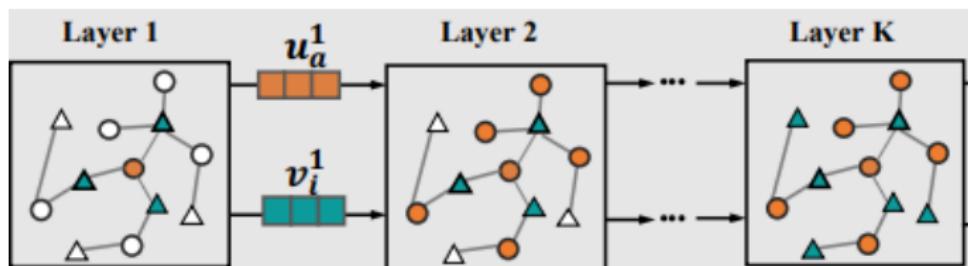
- Graph Neural Network



¹Inductive 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

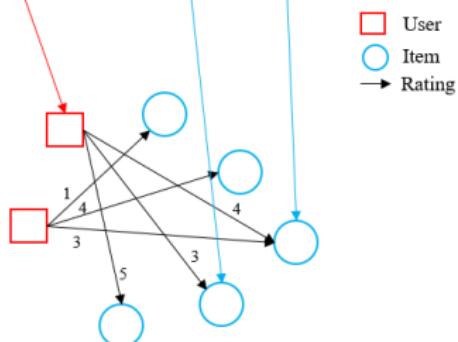
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



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



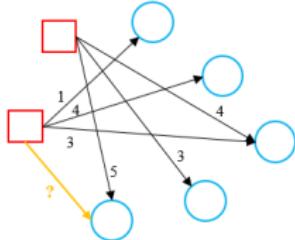
Input Graph

- Rating Matrix → Graph

	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

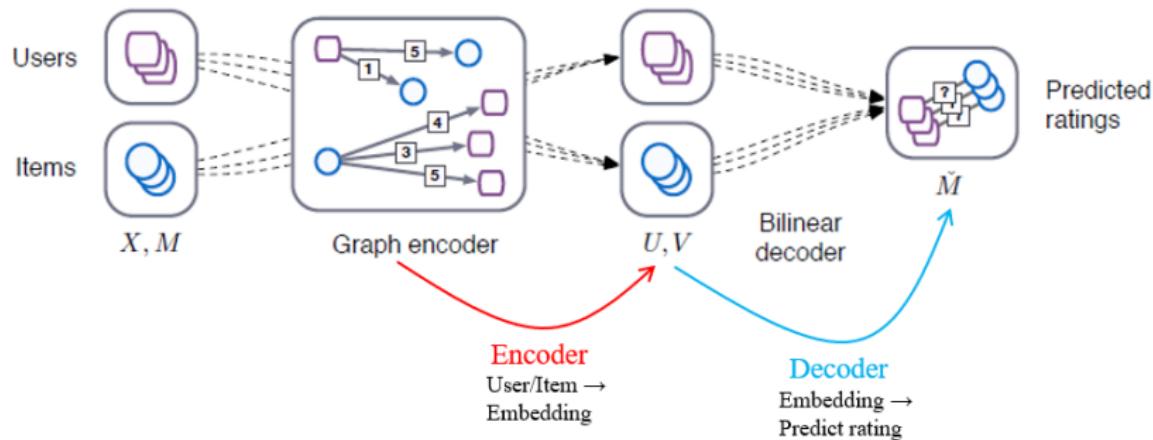


>User
Item
→ Rating



Architecture

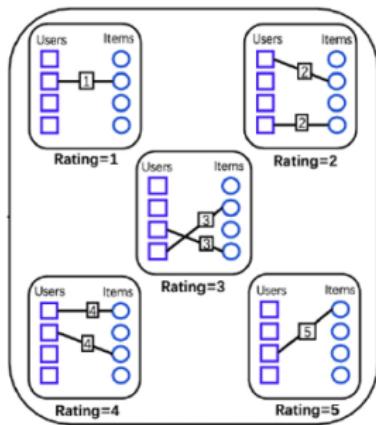
- Encoder-Decoder²



²Graph Convolutional Matrix Completion[KDD'18]

Architecture

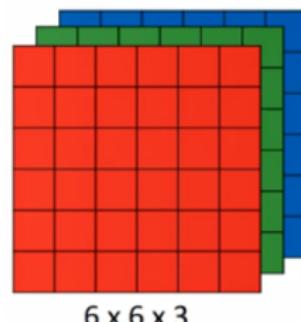
- Encoder



Rating

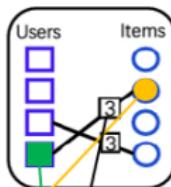


Channel



Architecture

- Encoder



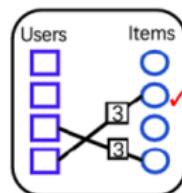
$$m_{v_j \rightarrow u_i, r} = W_r^T x_j, \quad (1)$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \rightarrow u_i, r}, \quad (2)$$

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

Architecture

- Encoder



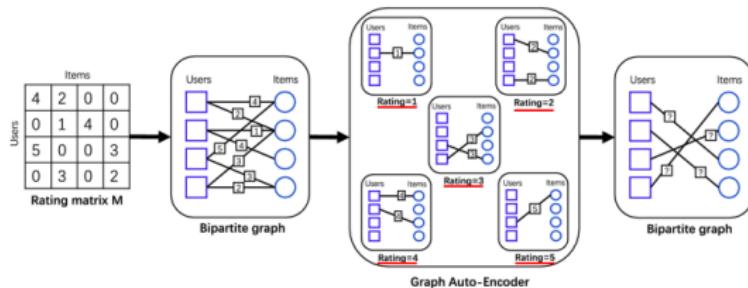
$$m_{v_j \rightarrow u_i, r} = W_r^T x_j, \quad (1)$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \rightarrow u_i, r}, \quad (2)$$

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

Architecture

- Encoder



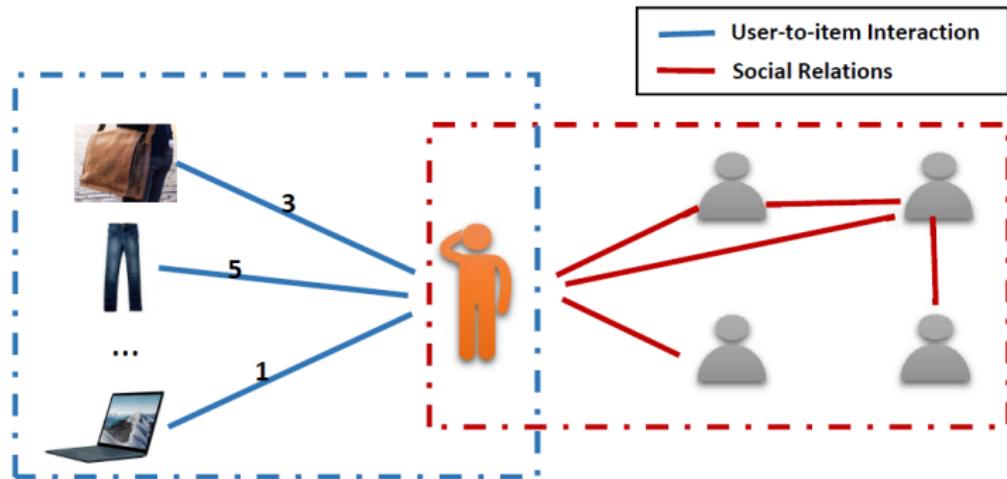
$$m_{v_j \rightarrow u_i, r} = W_r^T x_j, \quad (1)$$

$$m_{u_i, r} = \frac{1}{|N_{i,r}|} \sum_{j \in N_{i,r}} m_{v_j \rightarrow u_i, r}, \quad (2)$$

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

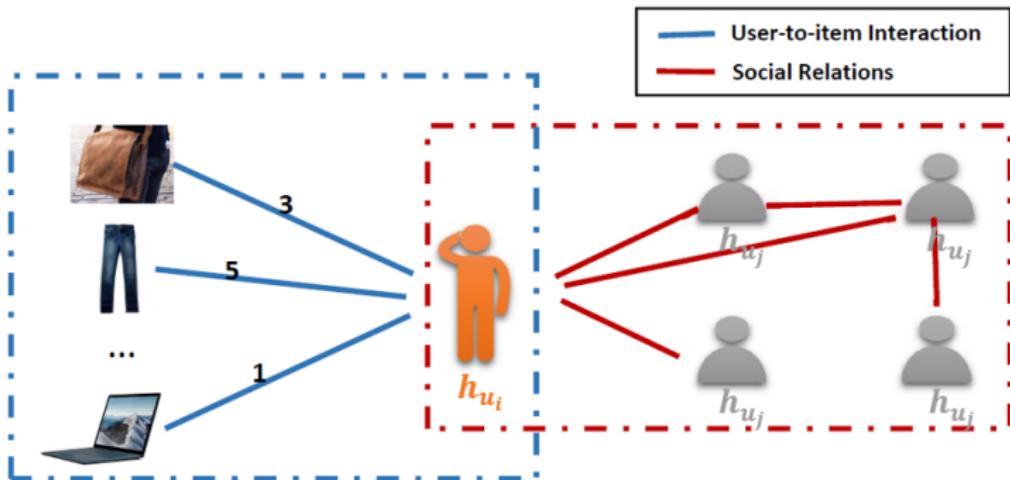
Analysis

- Limitation



Improvement

- Modification

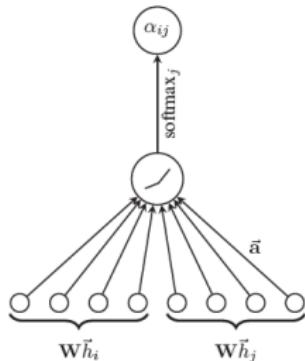


$$h_{u_i} = \sum_{j \in N_i} \alpha_{ij} h_{u_j},$$

- 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 h_v^k of each node in the graph following an **attention network**.

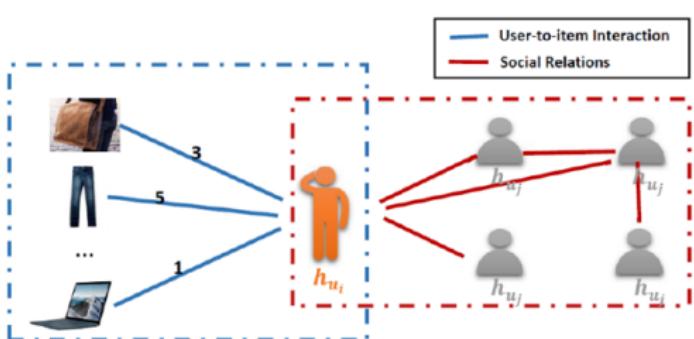
Improvement

- Attention Mechanism



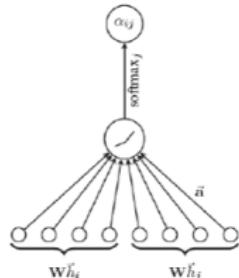
$$\begin{aligned} e_{ij} &= a(\vec{Wh}_{u_i}, \vec{Wh}_{u_j}), \\ &= \text{LeakyReLU}(\vec{a}^T [\vec{Wh}_{u_i} || \vec{Wh}_{u_j}]) \\ \alpha_{ij} &= \text{softmax}_j(e_{ij}) \\ &= \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \end{aligned}$$

Improvement



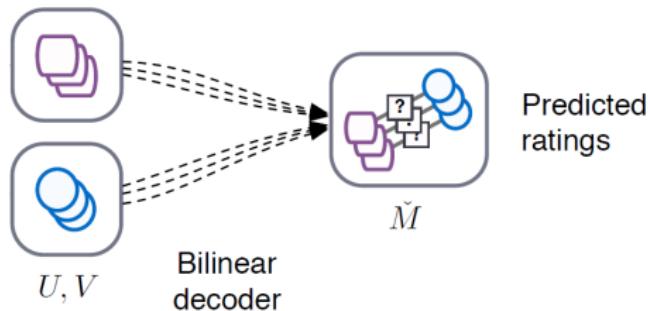
$$h_{u_i} = \sum_{j \in N_i} \alpha_{ij} h_{u_j},$$

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



Architecture

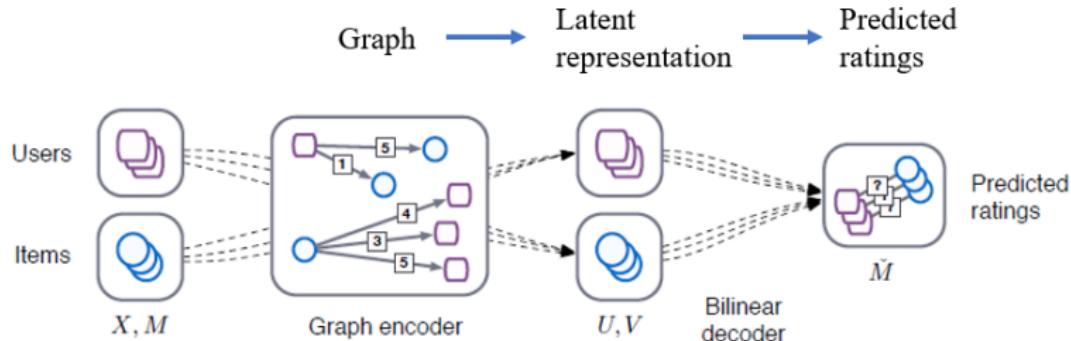
- Decoder



$$(P_r)_{ij} = \frac{\exp(u_i^T Q_r v_j)}{\sum_{s \in R} \exp(u_i^T Q_s v_j)}$$

$$\hat{M} = \sum_{r \in R} r P_r,$$

Architecture



- Loss Function

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

Experiment

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

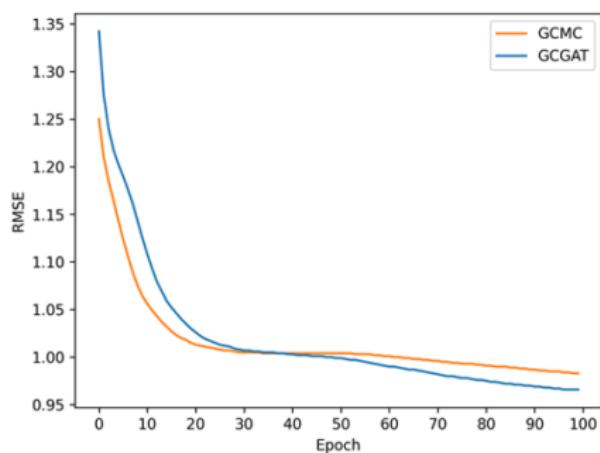
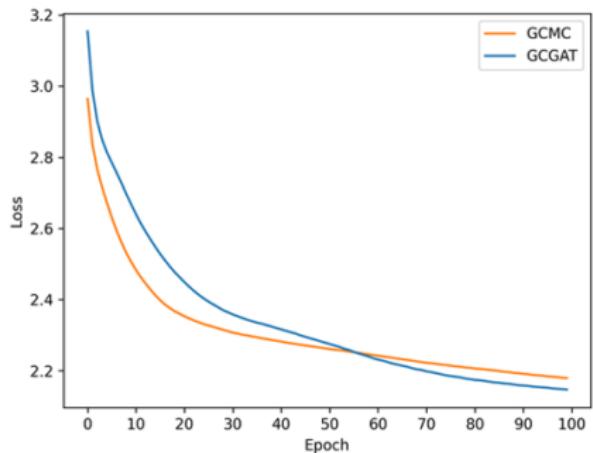
Dataset	Users	Items	Ratings	Rating Levels
ML-100K	943	1682	100,000	1,2,...,5

- Metrics

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

Experiment

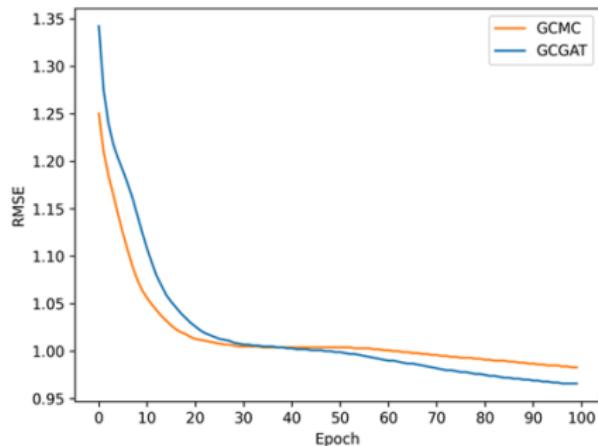
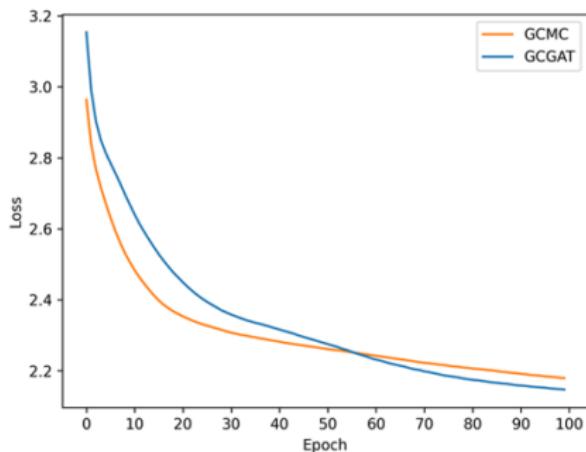
- Result



Model	RMSE
GCMC	0.983
GCGAT	0.965

Experiment

- Result



Model	RMSE
GCMC	0.983
GCGAT	0.965

<https://github.com/BitHub00/GCGAT>

End

Thank you!