
GRAPH-BASED RECOMMENDATION

A gentle introduction to recommender systems working with and on graphs

Daniele Malatesta

Postdoc researcher at CentraleSupélec, Inria, Université Paris-Saclay
Gif-sur-Yvette (France)

Email: daniele.malatesta@centralesupelec.fr

ACKNOWLEDGEMENTS

DISCLAIMER: The current slides are based upon an invited lecture held in the Dominante Math Data Science (MDS) – Master in DSBA – M.Sc. in AI programs at CentraleSupélec for the course of “**Machine Learning in Network Science**” (Instructor: Prof. Fragkiskos D. Malliaros) and an invited talk at **Cognism.com**.

Furthermore, this lecture is partially based on the material by:

- Jure Leskovec, Stanford University
- Xavier Bresson, National University of Singapore
- Previous works from my PhD at Politecnico di Bari (Italy)

NOTE: I’m planning to write some lecture notes from the current material. Stay tuned!

CONTENTS

- The recommendation task and traditional models
- Graph-based recommendation
- Background notions on graph neural networks
- Current directions in graph neural networks for recommendation



The recommendation task and traditional models

INFORMATION OVERLOAD ON THE INTERNET

We are assisting to an era of information overload on the Internet, where popular online platforms (e.g., Netflix, Amazon, Spotify, YouTube, X, Meta, Booking.com) store immense catalogues of data customers can interact with.



THE NEED FOR FILTERING ALGORITHMS

Users may feel overwhelmed in front of such an amount of data, as it becomes difficult (if not impossible) to navigate through the whole catalogue. That is why we need some **filtering algorithms** to select only the **relevant** data.



RECOMMENDER SYSTEMS

“Recommender Systems are software tools and techniques providing suggestions for items to be of use of a user. The suggestions provided are aimed at supporting their users in various decision-making processes, such as items to buy, what music to listen , or what news to read.”



THE RECOMMENDATION DATA

In the most basic setting, the recommendation data generally involves a set of users

$$U = \{u_1, u_2, \dots, u_N\}$$

and a set of items

$$I = \{i_1, i_2, \dots, i_M\}$$

and their recorded interactions ($R \in \mathbb{R}^{N \times M}$).





						
	✓					✓
	✓		✓			
		✓	✓		✓	
	✓			✓		
		✓				✓

EXPLICIT VS. IMPLICIT FEEDBACK

User-item interactions can come in the form of **explicit** feedback (i.e., ratings) or **implicit** feedback (e.g., views, clicks, visits). Implicit feedback is the most common (and complex) setting!

						
	3					1
	1		5			
		4	4		3	
	1			2		
		3				5

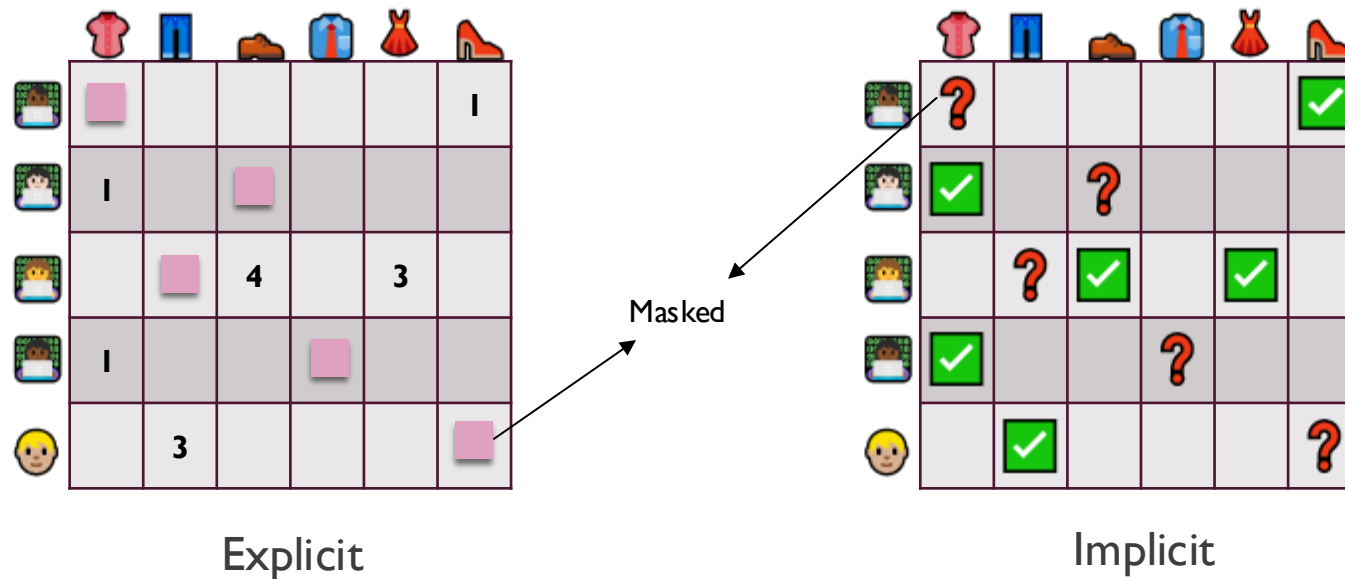
Explicit

Implicit

POSSIBLE RECOMMENDATION TASKS

We mask certain entries of the interaction matrix and retain only a portion of it (training set). Our goal is to predict the masked entries (test set). This kind of reminds **link prediction**!



RATING PREDICTION

Given a set of user-item pairs, we aim to design a **utility function** able to predict the rating each user will give to each item.

► Utility function $f_{\text{rec}} : U \times I \times R \rightarrow \mathbb{R}$

► Evaluation metrics:

$$\text{MSE}(\hat{y}, y) = \frac{1}{|R_{\text{test}}|} \sum_{u,i \in R_{\text{test}}} (f_{\text{rec}}(u, i) - R_{ui})^2$$

$$\text{RMSE}(\hat{y}, y) = \sqrt{\text{MSE}(\hat{y} - y)}$$

$$\text{MAE}(\hat{y}, y) = \frac{1}{|R_{\text{test}}|} \sum_{u,i \in R_{\text{test}}} |f_{\text{rec}}(u, i) - R_{ui}|$$

TOP-K RECOMMENDATION

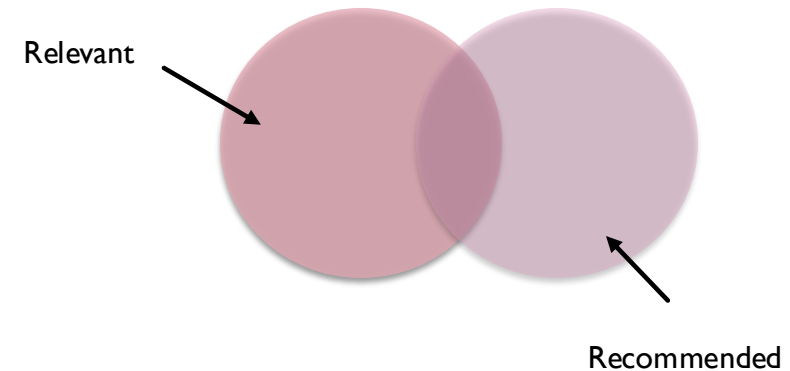
Given a set of users, we aim to design a **utility function** able to predict a list of top-K items each user might like from the catalogue.

► Utility function $f_{\text{rec}} : U \times I \times R \rightarrow \mathbb{R}$

► Evaluation metrics: $\text{Recall@K} = \frac{1}{U} \sum_{u \in U} \frac{|\text{Rec}_u@K \cap \text{Rel}_u|}{|\text{Rel}_u|}$

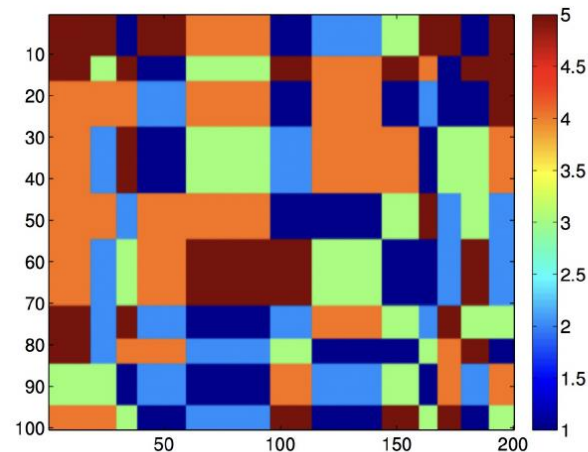
$$\text{Precision@K} = \frac{1}{U} \sum_{u \in U} \frac{|\text{Rec}_u@K \cap \text{Rel}_u|}{K}$$

$$\text{nDCG@K} = \frac{1}{U} \sum_{u \in U} \frac{\text{DCG}_u@K}{\text{IDCG}_u@K} \left\{ \begin{array}{l} \text{DCG}_u@K = \sum_{\hat{R}_{ui} \in \text{Rec}_u@K} \frac{\hat{R}_{ui}}{\log_2(i+1)} \\ \text{IDCG}_u@K = \sum_{R_{ui} \in \text{Rel}_u@K} \frac{R_{ui}}{\log_2(i+1)} \end{array} \right.$$



COLLABORATIVE FILTERING (1/3)

One of the main paradigms to design a recommender system is **collaborative filtering**. It assumes that the user-item interaction data is a low-rank matrix, meaning that several rows and columns (i.e., user and item ratings) are linearly dependent.



Low-rank matrix

$$\mathbf{X} \in \mathbb{R}^{100 \times 200}$$

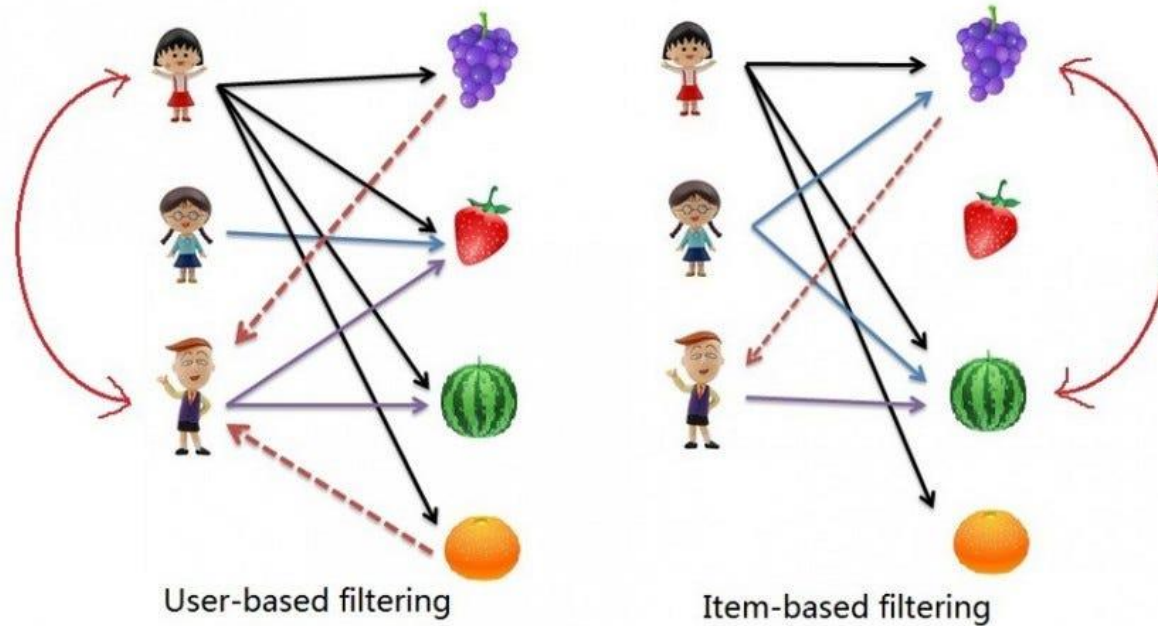
linearly independent rows = 13

linearly independent cols = 15

$$\Rightarrow \text{rank}(\mathbf{X}) = \max(13, 15) = 15$$

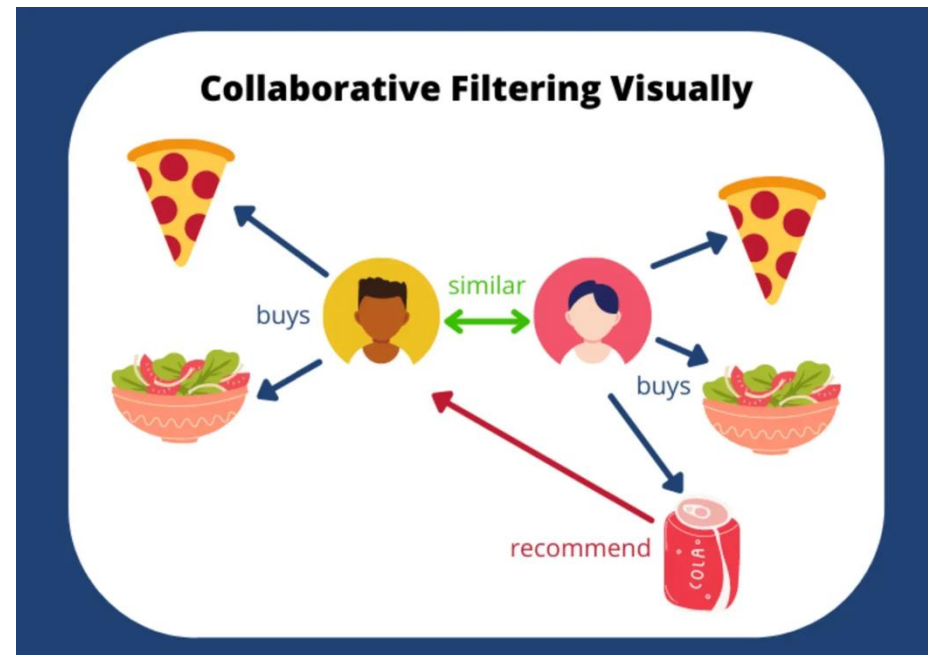
COLLABORATIVE FILTERING (2/3)

Practically, this entails that users (items) who have interacted with the same items (users) share similar characteristics.



COLLABORATIVE FILTERING (3/3)

Eventually, this means that if two users have similar tastes, we can recommend them similar (new) items.



ITEM-TO-ITEM COLLABORATIVE FILTERING

We want to suggest the most similar items from the catalogue to those the user has interacted with. To this end, we need to calculate an item-item similarity matrix as follows:

```
For each item in product catalog,  $I_1$ 
  For each customer  $C$  who purchased  $I_1$ 
    For each item  $I_2$  purchased by
      customer  $C$ 
      Record that a customer purchased  $I_1$ 
        and  $I_2$ 
  For each item  $I_2$ 
    Compute the similarity between  $I_1$  and  $I_2$ 
```

Cosine similarity

Their ratings vectors

	shirt	pants	shoes	bag	dress	boots
User 1	✓					✓
User 2	✓		✓			
User 3		✓	✓		✓	
User 4	✓			✓		
User 5		✓				✓

ITEM-TO-ITEM COLLABORATIVE FILTERING

We want to suggest the most similar items from the catalogue to those the user has interacted with. To this end, we need to calculate an item-item similarity matrix as follows:

$$\hat{R}_{ui} = \sum_{j \in I_u^+} \text{sim}_{ji}$$

Items interacted by the user

Similarity function (e.g., cosine similarity)

BAYESIAN PERSONALIZED RANKING (BPR)

- Let $T = \{(u, i, j) \mid i \in I_u^+ \wedge j \in I_u^-\}$ be a set of triples with user, **positive** item (interacted) and **negative** item (non-interacted).

BAYESIAN PERSONALIZED RANKING (BPR)

- Let $T = \{(u, i, j) \mid i \in I_u^+ \wedge j \in I_u^-\}$ be a set of triples with user, **positive** item (interacted) and **negative** item (non-interacted).
- Bayesian personalized ranking (BPR) seeks to minimize the following loss function:

$$L_{\text{BPR}} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj}),$$

Sigmoid function

Predicted rating for positive item

Predicted rating for negative item

BAYESIAN PERSONALIZED RANKING (BPR)

- Let $T = \{(u, i, j) \mid i \in I_u^+ \wedge j \in I_u^-\}$ be a set of triples with user, **positive** item (interacted) and **negative** item (non-interacted).
- Bayesian personalized ranking (BPR) seeks to minimize the following loss function:

$$L_{\text{BPR}} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj}),$$

Sigmoid function Predicted rating for positive item Predicted rating for negative item

- We aim to maximize the divergence between positive and negative items!

MATRIX FACTORIZATION WITH BPR (BPRMF)

- A very common way of predicting the user-item interaction score is through the matrix factorization (MF) model.

MATRIX FACTORIZATION WITH BPR (BPRMF)

- A very common way of predicting the user-item interaction score is through the matrix factorization (MF) model.
- We map users and items in the recommendation system to embeddings in the latent space $e_u, e_i \in \mathbb{R}^d$, where $d \ll N, M$.

MATRIX FACTORIZATION WITH BPR (BPRMF)

- A very common way of predicting the user-item interaction score is through the matrix factorization (MF) model.
- We map users and items in the recommendation system to embeddings in the latent space $e_u, e_i \in \mathbb{R}^d$, where $d \ll N, M$.
- Then, we predict the user-item interaction score as the dot product between the two embeddings:

$$\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$$

MATRIX FACTORIZATION WITH BPR (BPRMF)

- A very common way of predicting the user-item interaction score is through the matrix factorization (MF) model.
- We map users and items in the recommendation system to embeddings in the latent space $e_u, e_i \in \mathbb{R}^d$, where $d \ll N, M$.
- Then, we predict the user-item interaction score as the dot product between the two embeddings:

$$\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$$

- The user and item embeddings are weights of the recommendation model, trained to optimize the BPR loss.



Graph-based recommendation

RECOMMENDATION DATA IS A GRAPH! (1/5)

User-item interaction matrix

	i_1	i_2	i_3	i_4	i_5	i_6	Items
u_1	1	0	1	1	1	0	
u_2	0	0	0	0	1	0	
u_3	1	1	1	0	0	0	
u_4	0	0	1	1	0	1	

Users

RECOMMENDATION DATA IS A GRAPH! (2/5)

User-item interaction matrix

	i_1	i_2	i_3	i_4	i_5	i_6	→ Items
u_1	1	0	1	1	1	0	
u_2	0	0	0	0	1	0	
u_3	1	1	1	0	0	0	
u_4	0	0	1	1	0	1	
↓ Users							

	u_1	u_2	u_3	u_4	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	0	0	0	1	0	1	1	1	0
u_2	0	0	0	0	0	0	0	0	1	0
u_3	0	0	0	0	1	1	1	0	0	0
u_4	0	0	0	0	0	0	1	1	0	1
i_1	1	0	1	0	0	0	0	0	0	0
i_2	0	0	1	0	0	0	0	0	0	0
i_3	1	0	1	1	0	0	0	0	0	0
i_4	1	0	0	1	0	0	0	0	0	0
i_5	1	1	0	0	0	0	0	0	0	0
i_6	0	0	0	1	0	0	0	0	0	0

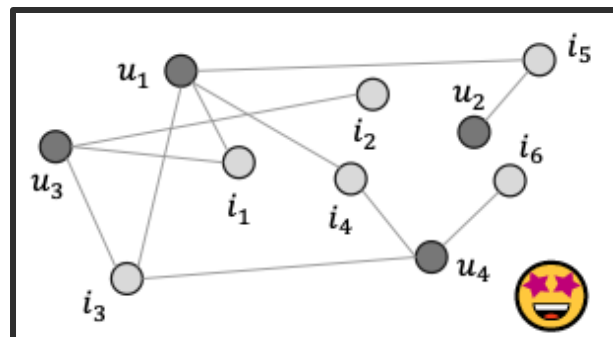
Adjacency matrix

RECOMMENDATION DATA IS A GRAPH! (3/5)

User-item interaction matrix

	i_1	i_2	i_3	i_4	i_5	i_6	Items
u_1	1	0	1	1	1	0	
u_2	0	0	0	0	1	0	
u_3	1	1	1	0	0	0	
u_4	0	0	1	1	0	1	

Users



BIPARTITE AND UNDIRECTED GRAPH

	u_1	u_2	u_3	u_4	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	0	0	0	1	0	1	1	1	0
u_2	0	0	0	0	0	0	0	0	1	0
u_3	0	0	0	0	1	1	1	0	0	0
u_4	0	0	0	0	0	0	1	1	0	1
i_1	1	0	1	0	0	0	0	0	0	0
i_2	0	0	1	0	0	0	0	0	0	0
i_3	1	0	1	1	0	0	0	0	0	0
i_4	1	0	0	1	0	0	0	0	0	0
i_5	1	1	0	0	0	0	0	0	0	0
i_6	0	0	0	1	0	0	0	0	0	0

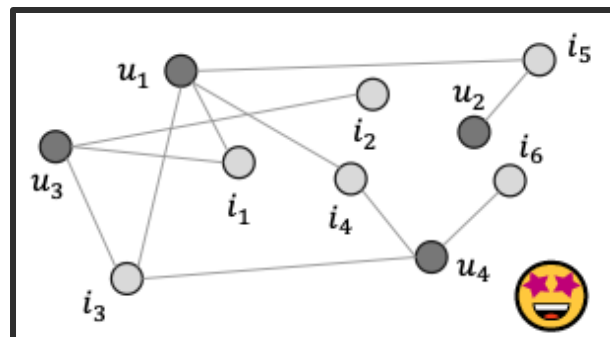
Adjacency matrix

RECOMMENDATION DATA IS A GRAPH! (4/5)

User-item interaction matrix

	i_1	i_2	i_3	i_4	i_5	i_6	Items
u_1	1	0	1	1	1	0	
u_2	0	0	0	0	1	0	
u_3	1	1	1	0	0	0	
u_4	0	0	1	1	0	1	

Users



BIPARTITE AND UNDIRECTED GRAPH

	u_1	u_2	u_3	u_4	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	0	0	0	1	0	1	1	1	0
u_2	0	0	0	0	0	0	0	0	1	0
u_3	0	0	0	0	1	1	1	0	0	0
u_4	0	0	0	0	0	0	1	1	0	1
i_1	1	0	1	0	0	0	0	0	0	0
i_2	0	0	1	0	0	0	0	0	0	0
i_3	1	0	1	1	0	0	0	0	0	0
i_4	1	0	0	1	0	0	0	0	0	0
i_5	1	1	0	0	0	0	0	0	0	0
i_6	0	0	0	1	0	0	0	0	0	0

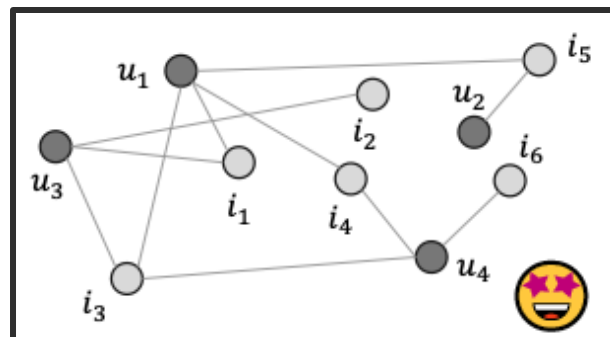
Adjacency matrix

RECOMMENDATION DATA IS A GRAPH! (5/5)

User-item interaction matrix

	i_1	i_2	i_3	i_4	i_5	i_6	Items
u_1	1	0	1	1	1	0	
u_2	0	0	0	0	1	0	
u_3	1	1	1	0	0	0	
u_4	0	0	1	1	0	1	

Users



BIPARTITE AND UNDIRECTED GRAPH

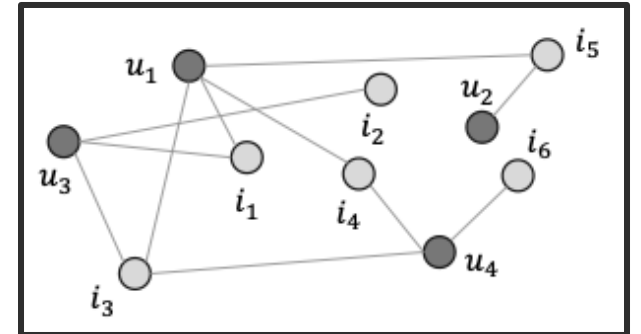
	u_1	u_2	u_3	u_4	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	0	0	0	1	0	1	1	1	0
u_2	0	0	0	0	0	0	0	0	1	0
u_3	0	0	0	0	1	1	1	0	0	0
u_4	0	0	0	0	0	0	1	1	0	1
i_1	1	0	1	0	0	0	0	0	0	0
i_2	0	0	1	0	0	0	0	0	0	0
i_3	1	0	1	1	0	0	0	0	0	0
i_4	1	0	0	1	0	0	0	0	0	0
i_5	1	1	0	0	0	0	0	0	0	0
i_6	0	0	0	1	0	0	0	0	0	0

Adjacency matrix

RECOMMENDATION GRAPH DATA: AN OVERVIEW

- Users' and items' nodes ($|U| = N$ and $|I| = M$)
- The user-item interaction matrix $R \in \mathbb{R}^{N \times M}$
- We build the adjacency matrix $A \in \mathbb{R}^{(N+M) \times (N+M)}$:

$$A = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}.$$



- Then, we can formally define the user-item bipartite and undirected graph $G = \{U \cup I, A\}$

RECOMMENDATION GRAPH DATA: AN OVERVIEW

- User-user and item-item co-purchase graphs:

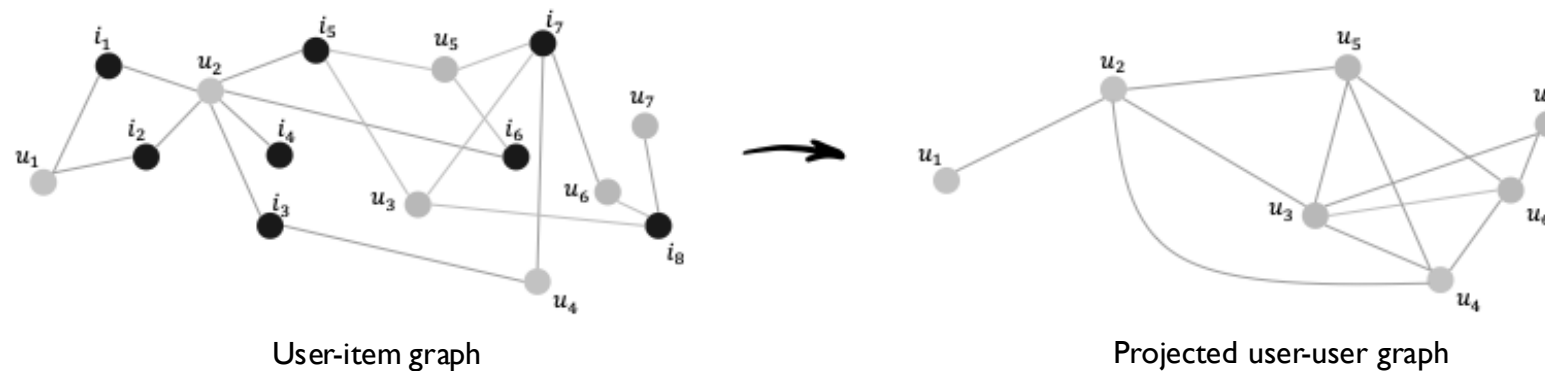
$$R^U = R \cdot R^T, R^I = R^T$$

RECOMMENDATION GRAPH DATA: AN OVERVIEW

- User-user and item-item co-purchase graphs:

$$R^U = R \cdot R^T, R^I = R^T$$

- They represent users interacting with the same items, and items being interacted by the same users



TOPOLOGICAL PROPERTIES (1/10)

Node degree

Let $N_u = \{i \in I \mid R_{ui} = 1\}$ and $N_i = \{u \in U \mid R_{ui} = 1\}$.

TOPOLOGICAL PROPERTIES (2/10)

Node degree

Let $N_u = \{i \in I \mid R_{ui} = 1\}$ and $N_i = \{u \in U \mid R_{ui} = 1\}$.

Then, the node degree user- and item-wise is calculated as:

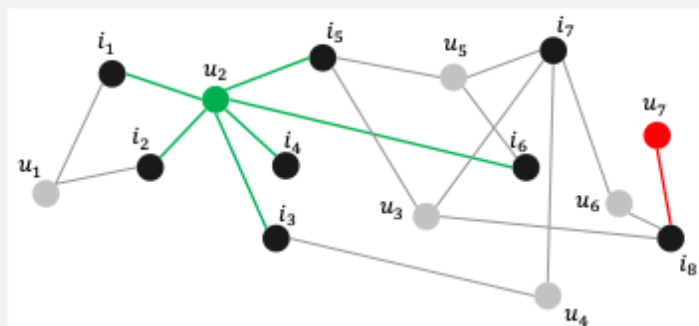
$$\sigma_u = |N_u|, \quad \sigma_i = |N_i|$$

TOPOLOGICAL PROPERTIES (3/10)

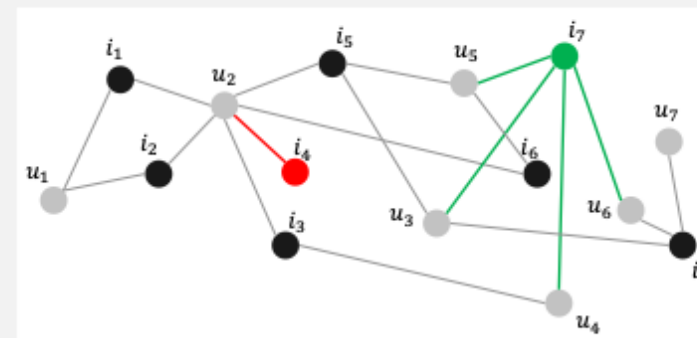
Node degree (re-interpretation)

NODE DEGREE IN RecSys

A graphical representation of the node degree interpretation in the user-item graph. On the left, the user node degree as a measure of user activity. On the right, the item node degree distinguishes popular from niche items.



(a) **active** user vs. **inactive** user



(b) **popular** item vs. **niche** item

TOPOLOGICAL PROPERTIES (4/10)

Clustering coefficient

With a notation abuse, let $N_u^{(2)} = \{v \in U \mid \exists i \in I, i \in N_u \wedge i \in N_v\}$ be the second-order neighbourhood set of u .

TOPOLOGICAL PROPERTIES (5/10)

Clustering coefficient

With a notation abuse, let $N_u^{(2)} = \{v \in U \mid \exists i \in I, i \in N_u \wedge i \in N_v\}$ be the second-order neighbourhood set of u .

Then, the clustering coefficient for u is calculated as:

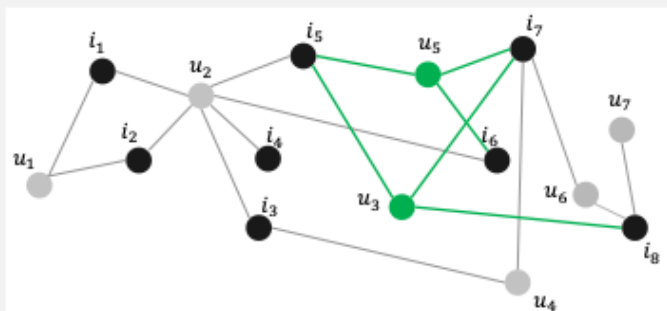
$$\gamma_u = \frac{\sum_{v \in N_u^{(2)}} \gamma_{u,v}}{|N_u^{(2)}|}, \quad \text{where } \gamma_{u,v} = \frac{|N_u \cap N_v|}{|N_u \cup N_v|}$$

TOPOLOGICAL PROPERTIES (6/10)

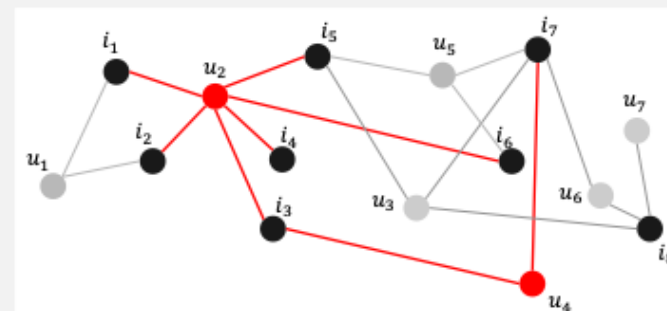
Clustering coefficient (re-interpretation)

CLUSTERING COEFFICIENT IN RecSys

Graphical representation of the node degree interpretation in the user-item graph. On the left, an example illustrates two similar users sharing the 67% of the interacted items. On the right, two neighboring users exhibit different preferences.



(a) **similar** user activity ($\gamma_{u_3, u_5} = 0.667$)



(b) **different** user activity ($\gamma_{u_2, u_4} = 0.167$)

TOPOLOGICAL PROPERTIES (7/10)

Degree assortativity

Let $D = \{d_1, d_2, \dots\}$ be the set of distinct degree values in the user-user (item-item) graph.

TOPOLOGICAL PROPERTIES (8/10)

Degree assortativity

Let $D = \{d_1, d_2, \dots\}$ be the set of distinct degree values in the user-user (item-item) graph.

Then, let $e_{d_h d_k}$ be the fraction of edges linking nodes with degrees d_h and d_k , and q_{d_h} the probability of reaching a node with degree d_h when traversing from another node with the same degree (**excess degree**).

TOPOLOGICAL PROPERTIES (9/10)

Degree assortativity

The degree assortativity in the user-user (item-item) graph is obtained as:

$$\rho = \frac{\sum_{d_h, d_k} d_h d_k (e_{d_h, d_k} - q_{d_h} q_{d_k})}{\text{std}_q^2}$$



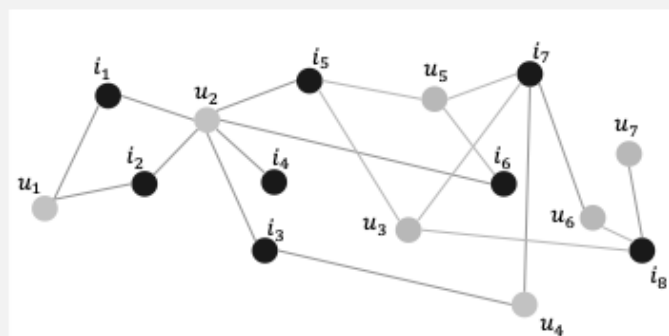
Standard deviation of the probability
distribution

TOPOLOGICAL PROPERTIES (10/10)

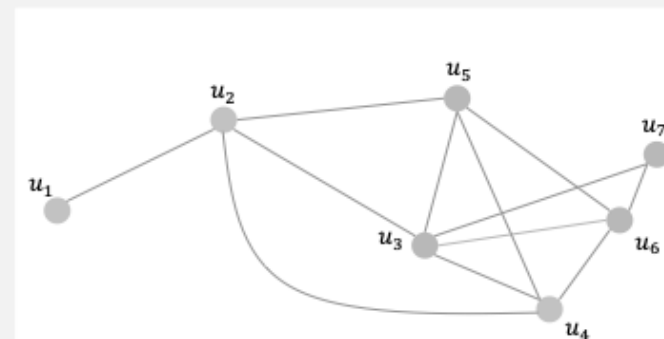
Degree assortativity (re-interpretation)

DEGREE ASSORTATIVITY IN RecSys

Graphical representation of the degree assortativity interpretation in the user-item graph. In this example, user nodes are different in terms of node degree, thus resulting in a low value of degree assortativity.



(a) original user-item graph



(b) degree (**dis**)assortativity ($\rho = -0.191$)

RECOMMENDATION BASED ON RANDOM WALKS (I/4)

Accurate, diverse, and scalable recommendations (RP3 β)

Let $D \in \mathbb{R}^{(N+M) \times (N+M)}$ be the diagonal degree matrix of the user-item graph, where $D_{vv} = |N_v|$.

RECOMMENDATION BASED ON RANDOM WALKS (2/4)

Accurate, diverse, and scalable recommendations (RP3 β)

Let $D \in \mathbb{R}^{(N+M) \times (N+M)}$ be the diagonal degree matrix of the user-item graph, where $D_{vv} = |N_v|$.

We can design a transition probability $P = D^{-1}A$ based upon random walks. This process can be repeated s times:

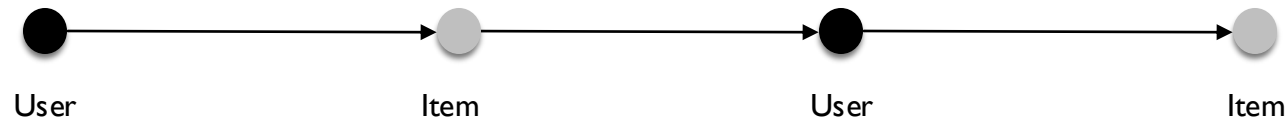
$$P^s = (D^{-1}A)^s$$

RECOMMENDATION BASED ON RANDOM WALKS (3/4)

Accurate, diverse, and scalable recommendations (RP3 β)

For $s = 3$, we have a good estimation of the user experience on online platforms:

$$P^3 = (D^{-1}A)^3$$



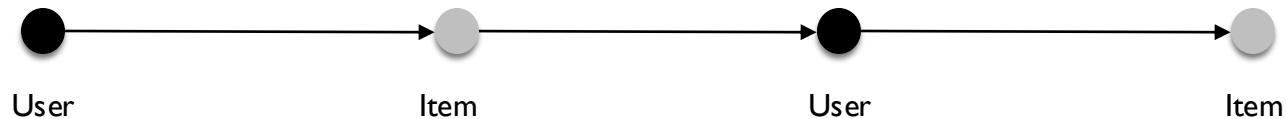
RECOMMENDATION BASED ON RANDOM WALKS (4/4)


Accurate, diverse, and scalable recommendations (RP3 β)

However, such a transition probability matrix is biased towards **popular** items (i.e., high-degree nodes). To address this aspect, we re-formulate the transition probability such that:

$$\tilde{p}_{ui}^3 = \frac{p_{ui}^3}{D_{ii}^\beta}$$

Scale with the item degree





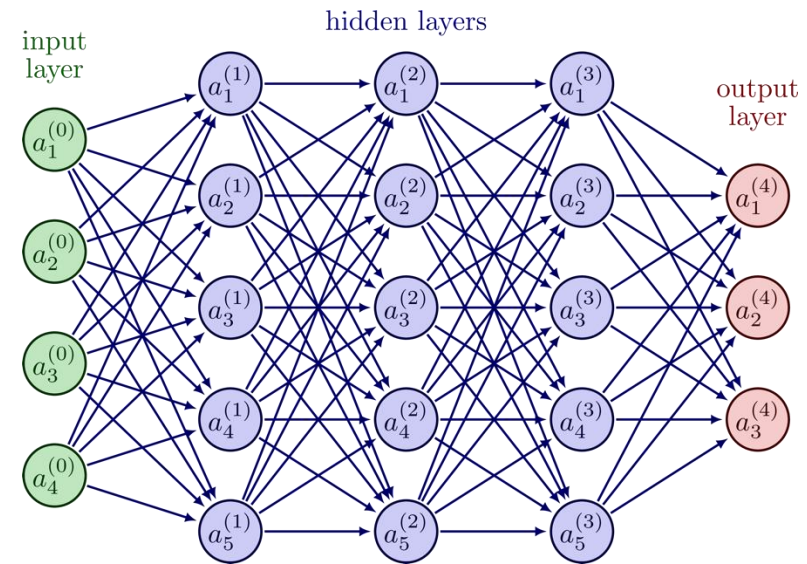
Background notions on graph neural networks

WHILE WITH TABULAR DATA...

	age (n)	job (c)	marital (c)	education (c)	balance (n)	housing (c)
0	30	unemployed	married	primary	1787	no
1	33	services	married	secondary	4789	yes
2	35	management	single	tertiary	1350	yes
3	30	management	married	tertiary	1476	yes
4	59	blue-collar	married	secondary	0	yes
5	35	management	single	tertiary	747	no

Input data

$$X \in \mathbb{R}^{N \times d}$$



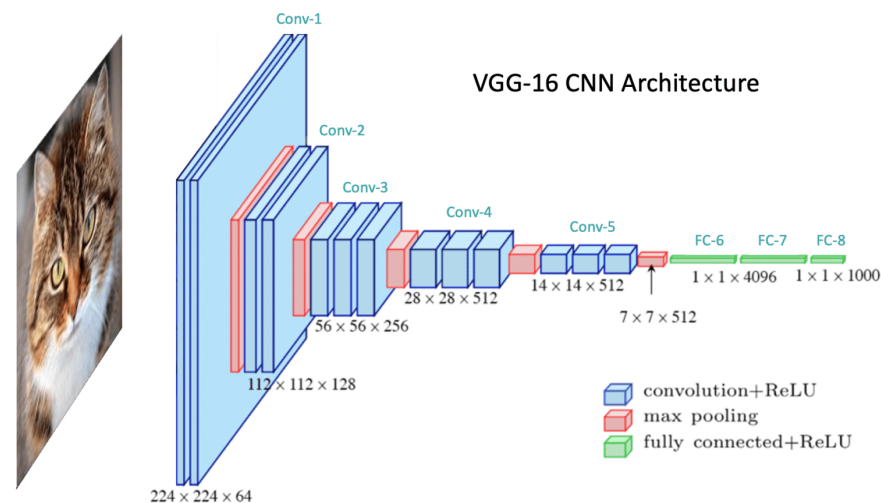
NEURAL NETWORKS

...AND IMAGES...



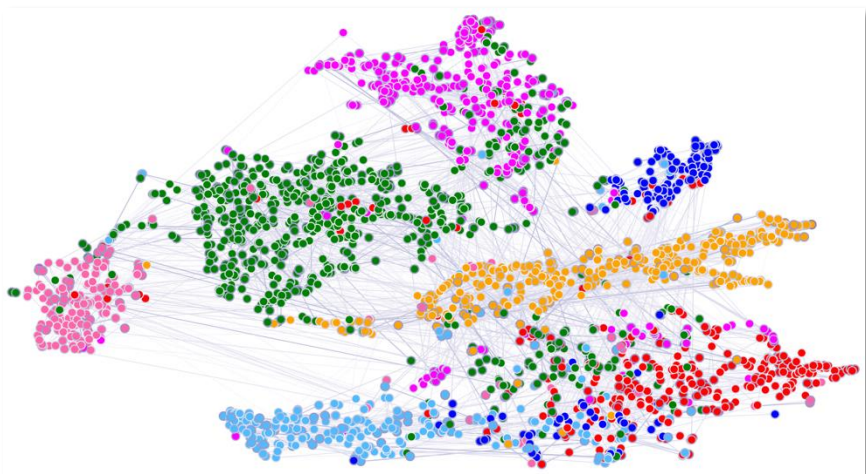
Input data

$$X \in \mathbb{R}^{H \times W}$$



DEEP CONVOLUTIONAL NEURAL NETWORKS

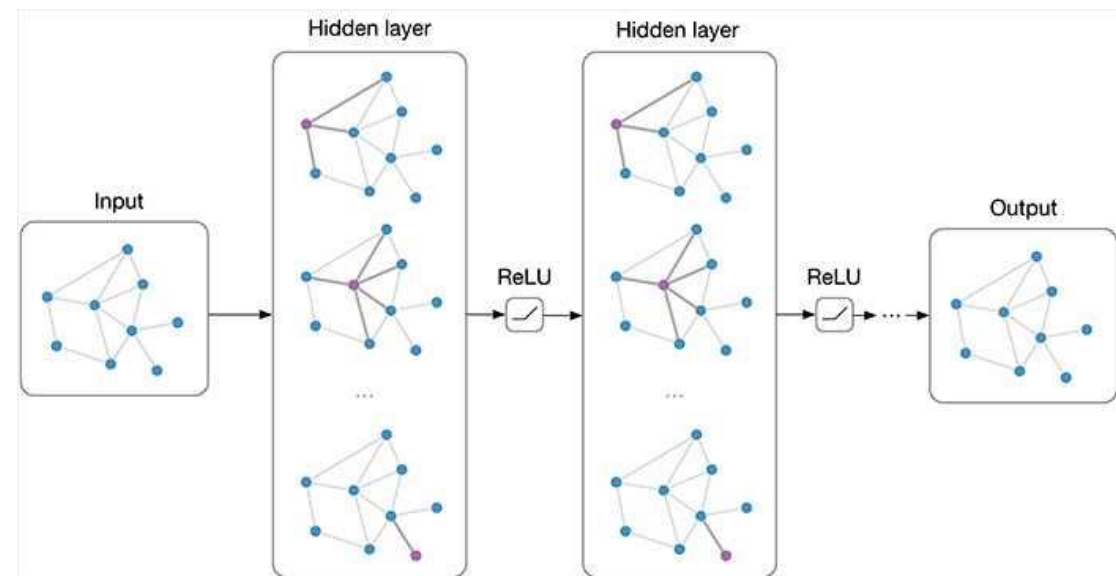
...WHAT ABOUT GRAPHS?



Input data

$$X \in \mathbb{R}^{N \times d}$$

$$A \in \mathbb{R}^{N \times N}$$



MESSAGE PASSING GRAPH
NEURAL NETWORKS

MESSAGE PASSING GRAPH NEURAL NETWORKS (1/4)

Message passing graph neural networks (**GNNs**) work on the **message passing** procedure, that iteratively refines the features of each **ego** node through the features of its **neighbor** nodes, at multiple distance hops

MESSAGE PASSING GRAPH NEURAL NETWORKS (2/4)

Message passing graph neural networks (**GNNs**) work on the **message passing** procedure, that iteratively refines the features of each **ego** node through the features of its **neighbor** nodes, at multiple distance hops

Let $X \in \mathbb{R}^{N \times d}$ be the initial node features, and $A \in \mathbb{R}^{N \times N}$ the adjacency matrix

MESSAGE PASSING GRAPH NEURAL NETWORKS (3/4)

Message passing graph neural networks (**GNNs**) work on the **message passing** procedure, that iteratively refines the features of each **ego** node through the features of its **neighbor** nodes, at multiple distance hops

Let $X \in \mathbb{R}^{N \times d}$ be the initial node features, and $A \in \mathbb{R}^{N \times N}$ the adjacency matrix

The message passing procedure, after l layers, refines the node features as:

$$X^{(l)} = \text{GNN}_l(A, X^{(l-1)}), \quad l \in [1, \dots, L]$$

Message passing layer
at layer l

Node features from
the previous iteration

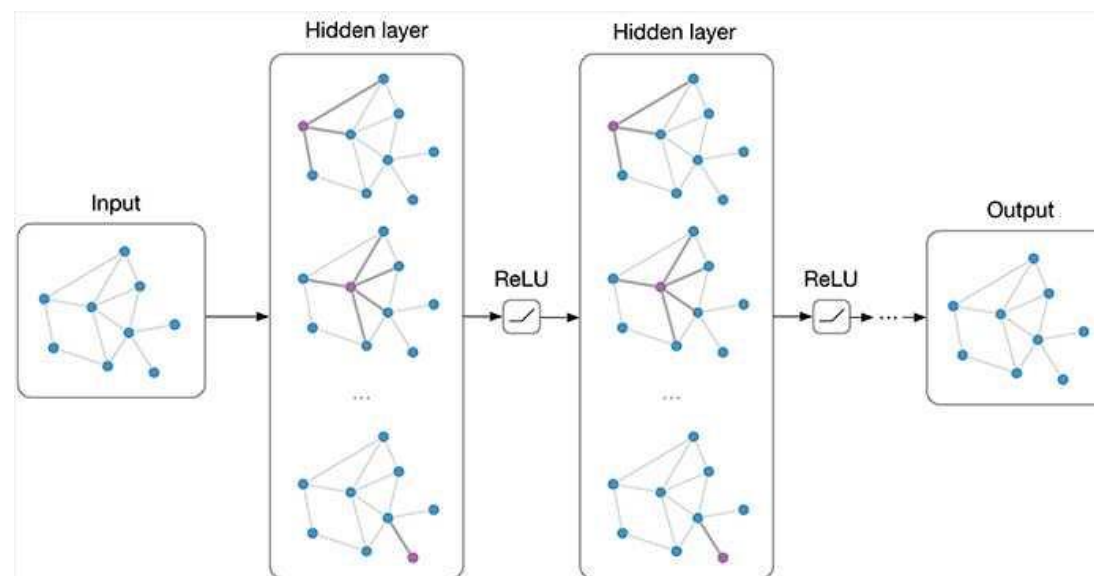
MESSAGE PASSING GRAPH NEURAL NETWORKS (4/4)

Finally, we obtain a unique final representation for each node features, and perform the prediction task (e.g., node classification, link prediction)

$$X^* = \text{AGGREGATE}(X^{(0)}, X^{(1)}, \dots, X^{(L)})$$

For instance, a weighted sum

We use these features for the prediction task

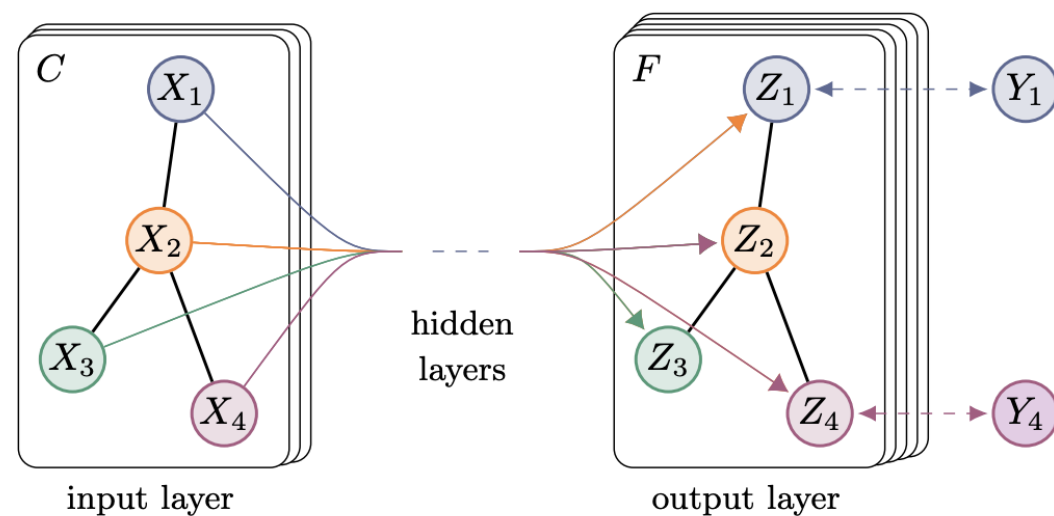


GRAPH CONVOLUTIONAL NETWORK (GCN) (1/2)

Graph convolutional network (GCN) is one of the pioneer approaches in graph representation learning:

$$X^{(l)} = \text{ReLU}(\tilde{A}X^{(l-1)}W^{(l)}), \quad l \in [1, \dots, L]$$

Activation function Normalized adjacency matrix Feature projection

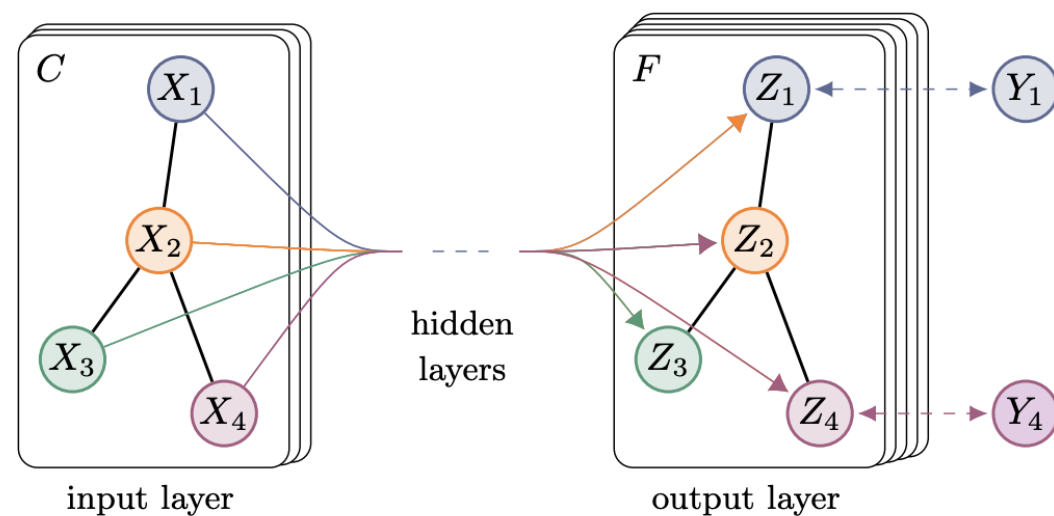


GRAPH CONVOLUTIONAL NETWORK (GCN) (2/2)

With a focus on a single node:

$$x_v^{(l)} = \text{ReLU} \left(\sum_{u \in N_v} \frac{1}{\sqrt{|N_v||N_u|}} x_u^{(l-1)} W^{(l)} \right)$$

Neighbor nodes Normalization Shared across all nodes





Current directions in graph neural networks for recommendation

LET US CONSIDER MATRIX FACTORIZATION ONCE AGAIN (I/4)

Let us go back to the formulation of MF:

- Inference: $\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$
- Loss: $L_{BPR} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj})$

LET US CONSIDER MATRIX FACTORIZATION ONCE AGAIN (2/4)

Let us go back to the formulation of MF:

- Inference: $\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$

No use of the user-item graph 🤖

- Loss: $L_{BPR} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj})$

LET US CONSIDER MATRIX FACTORIZATION ONCE AGAIN (3/4)

Let us go back to the formulation of MF:

- Inference: $\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$

No use of the user-item graph 🙄

- Loss: $L_{BPR} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj})$

User-item interactions in BPR
😍 ...

LET US CONSIDER MATRIX FACTORIZATION ONCE AGAIN (4/4)

Let us go back to the formulation of MF:

■ Inference: $\hat{R}_{ui} = \langle e_u, e_i \rangle = \sum_{f=1 \dots d} e_{uf} e_{if}$

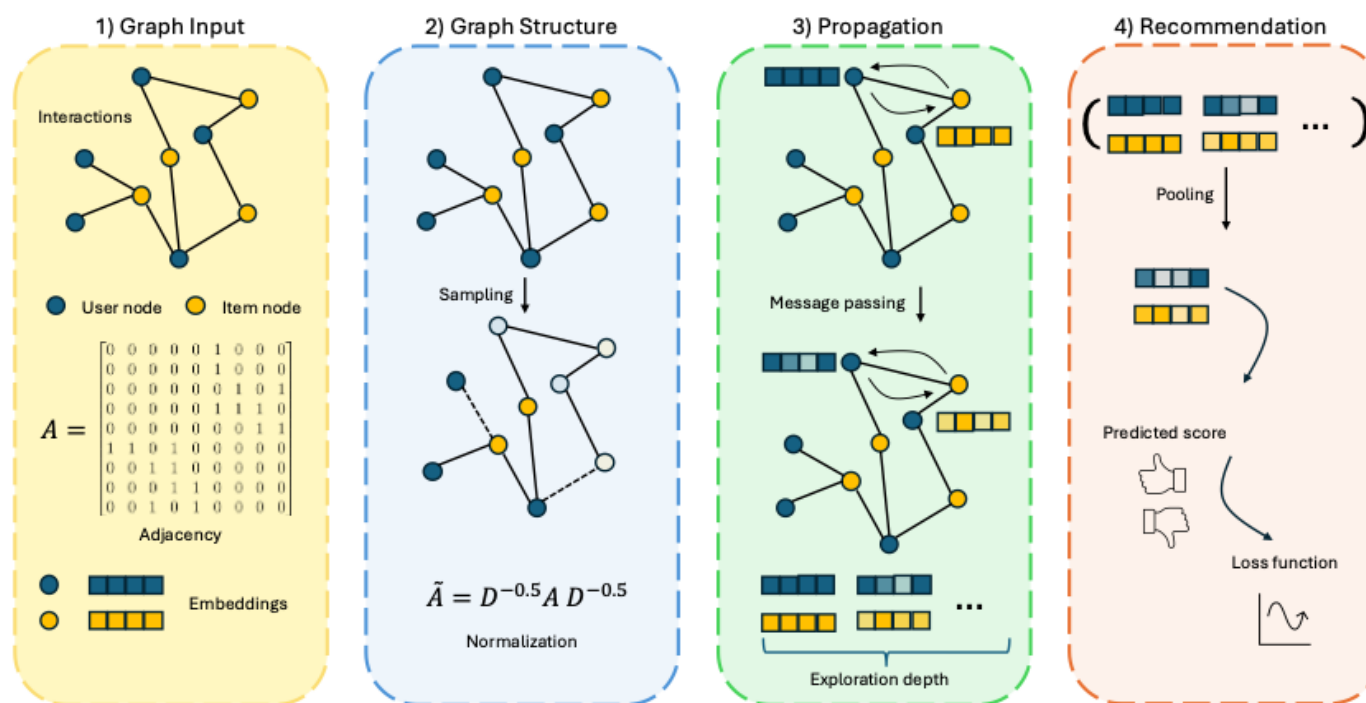
No use of the user-item graph 🙄

■ Loss: $L_{BPR} = \sum_{(u,i,j) \in T} -\ln \sigma(\hat{R}_{ui} - \hat{R}_{uj})$

User-item interactions in BPR
😍 ...
...but only at 1-hop
😞

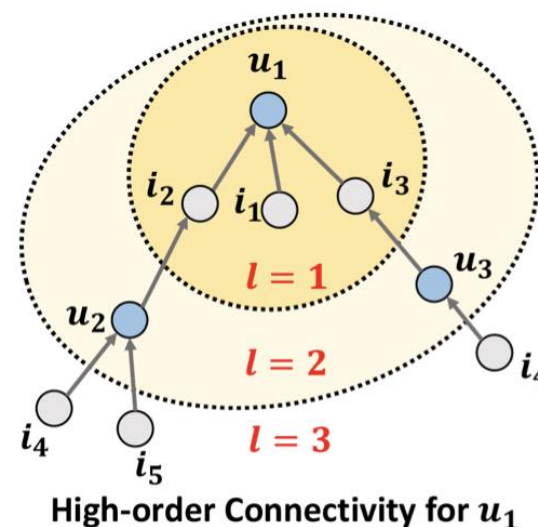
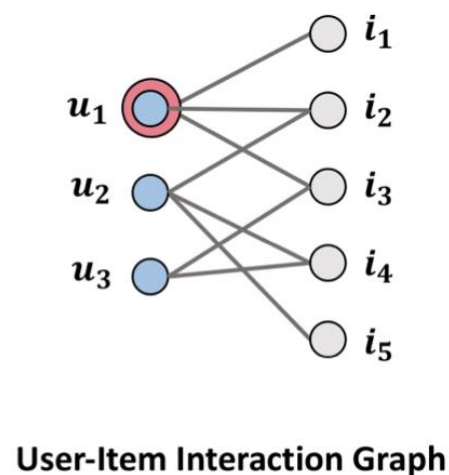
CAN WE DO ANYTHING ABOUT THIS?

YES! We need to incorporate message-passing graph neural networks (GNNs) into the MF pipeline!



NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (1/6)

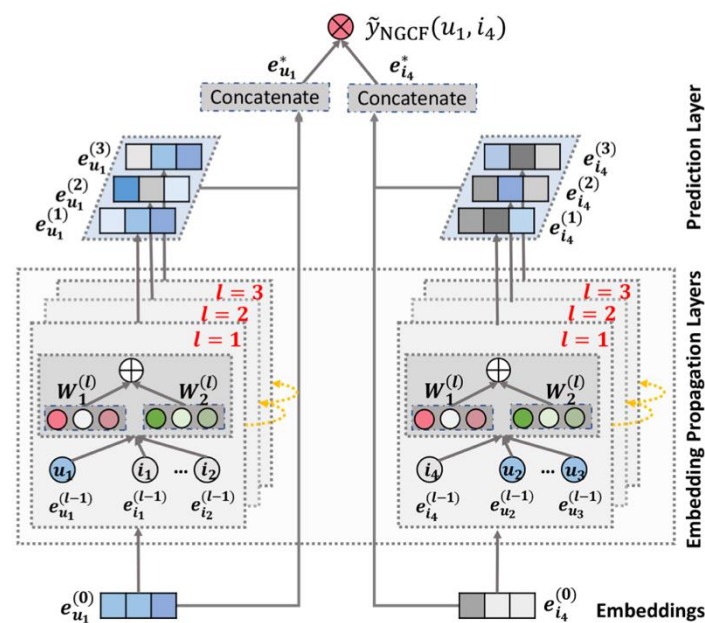
- Propagate the collaborative signal at multiple hops in the user-item graph
- Message passing is like GCN but considers the interdependence of neighbour nodes



NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (2/6)

- Initialize the node embeddings are the users' and items' embeddings

$$e_u^{(0)} = e_u, \quad e_i^{(0)} = e_i$$



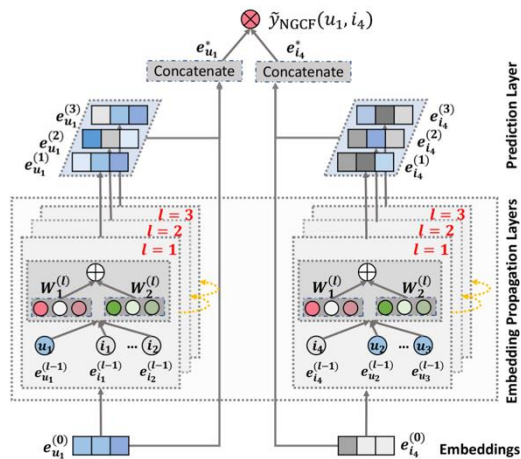
NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (3/6)

- Perform the following message-passing

$$e_u^{(l)} = \text{LeakyReLU}\left(\sum_{i \in N_u} \frac{W_{\text{neigh}}^{(l)} e_i^{(l-1)} + W_{\text{inter}}^{(l)} (e_u^{(l-1)} \odot e_i^{(l-1)})}{\sqrt{|N_u| |N_i|}}\right)$$

Neighbour contribution

Ego node – neighbour interdependence

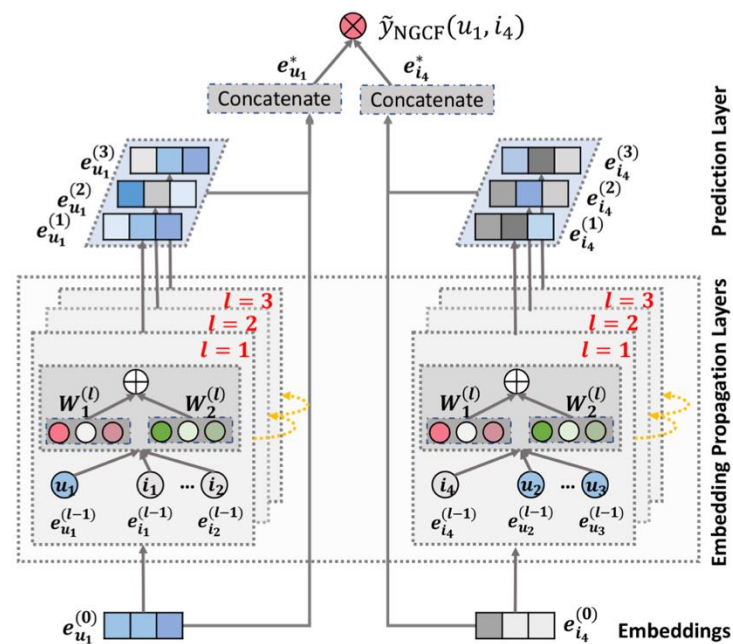


NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (4/6)

- Aggregate all layers' outputs

$$e_u^* = e_u^{(0)} \parallel e_u^{(1)} \parallel \dots \parallel e_u^{(L)}$$

Concatenation

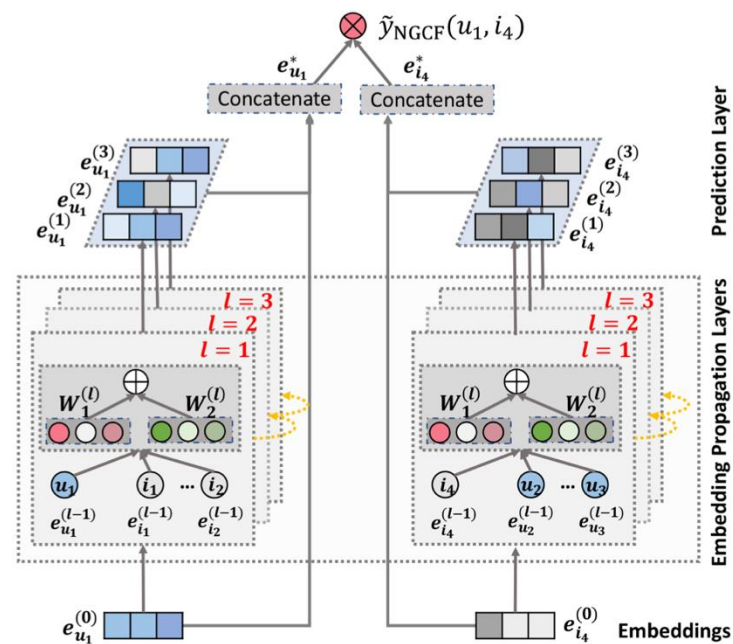


NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (5/6)

- Finally, calculate the dot product

$$\hat{R}_{ui} = e_u^{*T} \cdot e_i^*$$

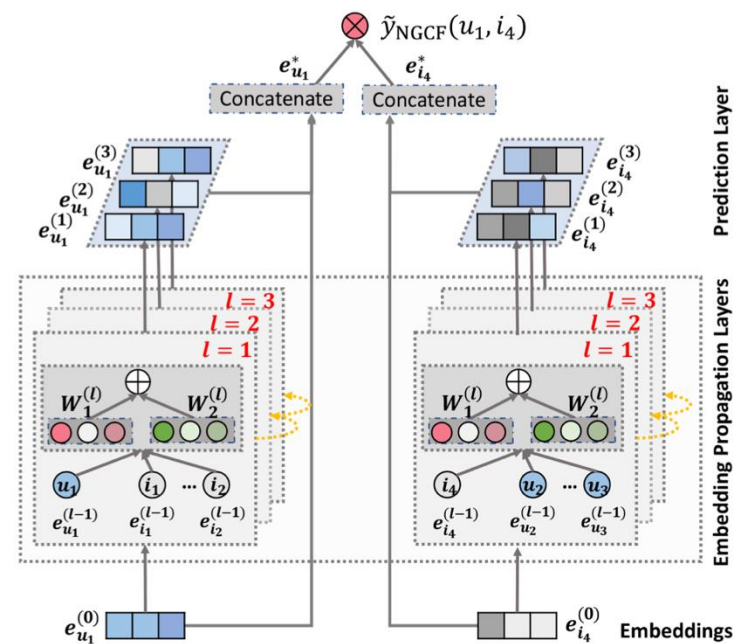
Final embedding layers
after the message passing



NEURAL GRAPH COLLABORATIVE FILTERING (NGCF) (6/6)

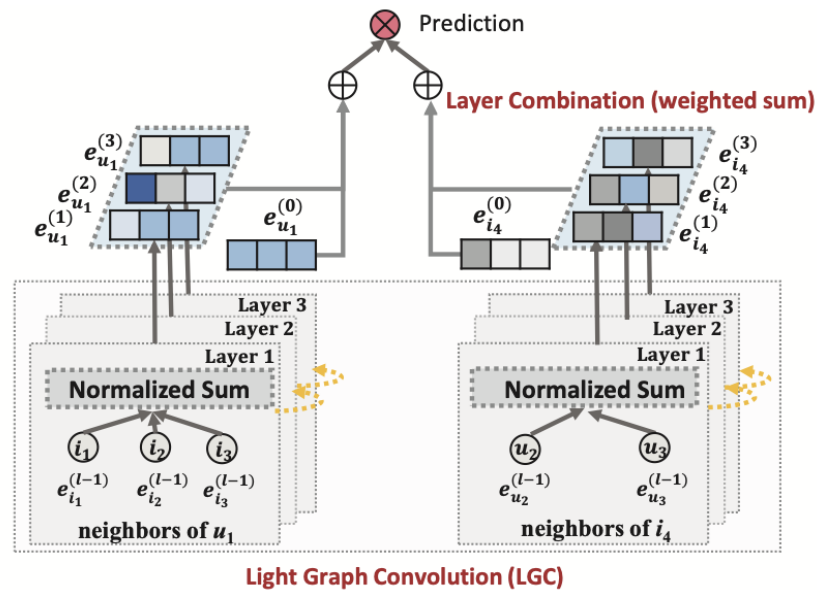
Training algorithm

- For epoch in EPOCHS:
 - For batch in BATCHES:
 - Perform message passing on the whole graph
 - Predict on the batch triples only
 - Compute BPR loss
 - Backward on the loss



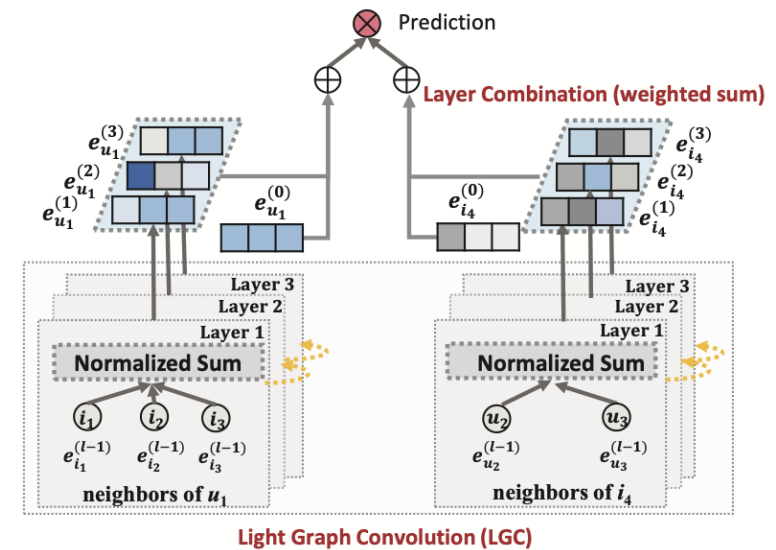
LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (I/I0)

- Simplifies the message-passing schema from NGCF
- Improves the performance (still one of the strongest baselines so far!)



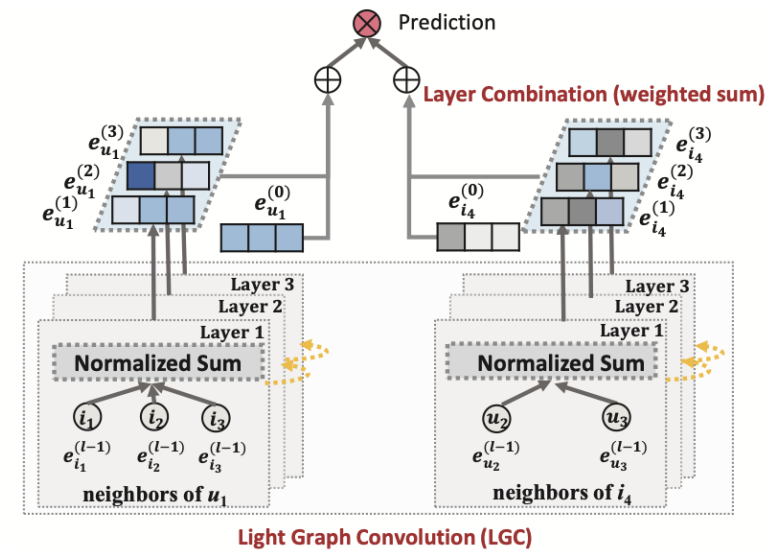
LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (2/10)

- Let us consider the GCN message passing once again



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (3/10)

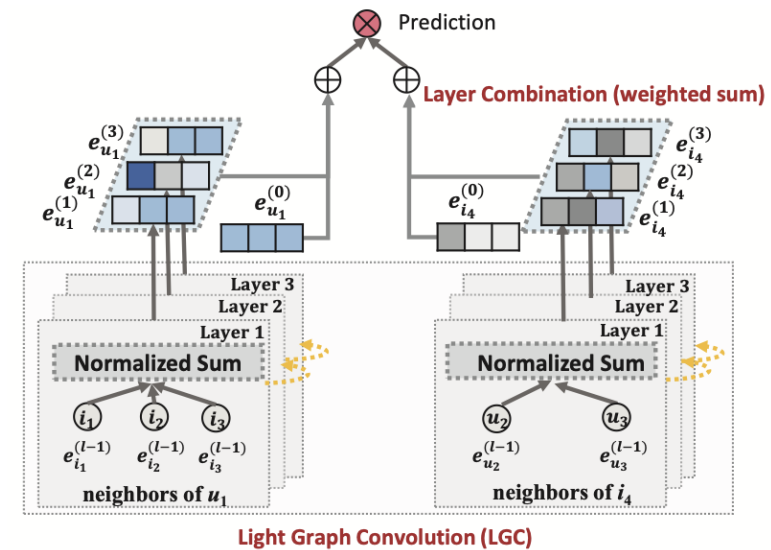
- Let us consider the GCN message passing once again
- Let $\tilde{A} = D^{-1/2}AD^{-1/2}$ be the normalized adjacency matrix



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (4/10)

- Let us consider the GCN message passing once again
- Let $\tilde{A} = D^{-1/2}AD^{-1/2}$ be the normalized adjacency matrix
- We compute the message-passing:

$$e^{(l)} = \text{ReLU}(\tilde{A}e^{(l-1)}W^{(l)})$$

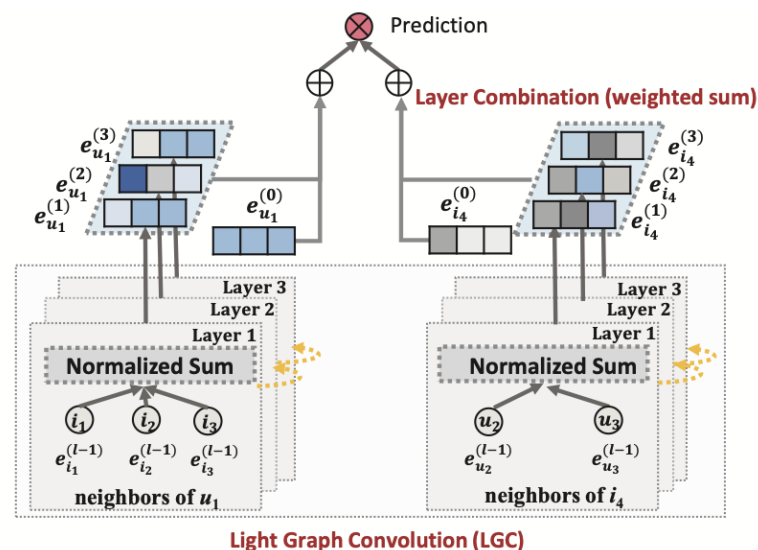


LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (5/10)

- Let us consider the GCN message passing once again
- Let $\tilde{A} = D^{-1/2}AD^{-1/2}$ be the normalized adjacency matrix
- We compute the message-passing:

$$e^{(l)} = \text{ReLU}(\tilde{A}e^{(l-1)}W^{(l)})$$

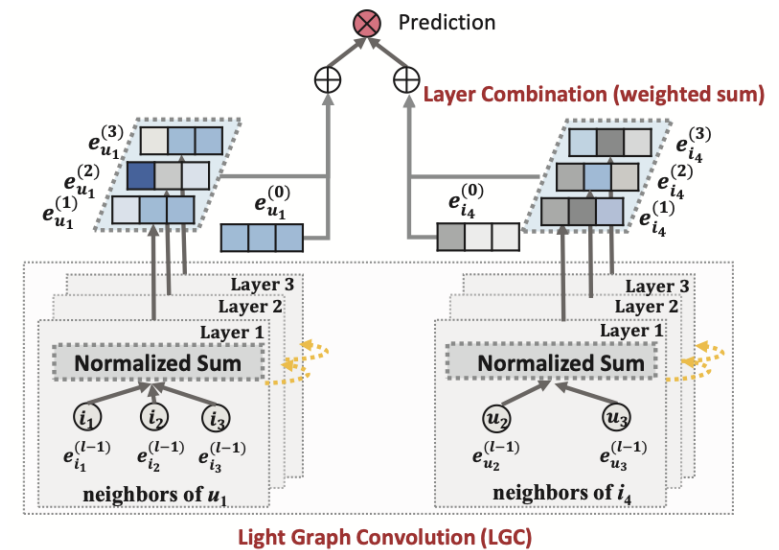
Can we remove the non-linearity?



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (6/10)

- We compute the message-passing:

$$\begin{aligned}
 e^{(l)} &= \tilde{A} e^{(l-1)} W^{(l)} \\
 &\quad \tilde{A} e^{(l-2)} W^{(l-1)} \\
 &\quad \tilde{A} e^{(l-3)} W^{(l-2)} \\
 &\quad \vdots \\
 &\quad \tilde{A} e^{(0)} W^{(0)}
 \end{aligned}$$

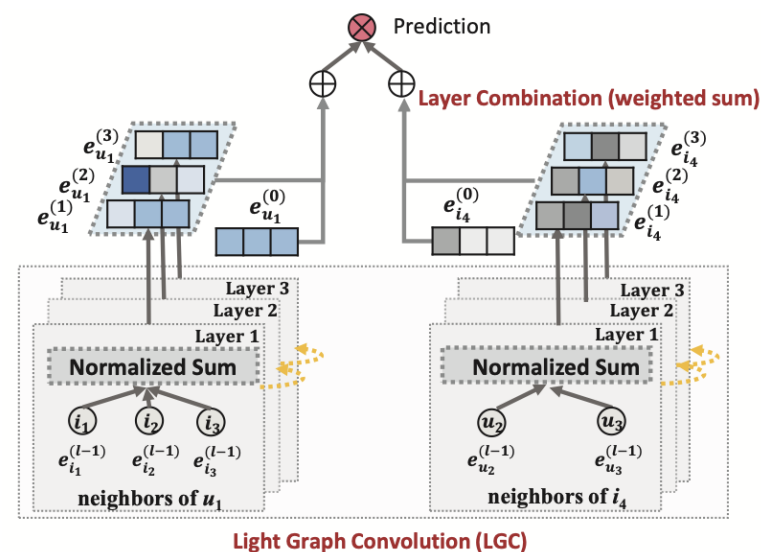


LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (7/10)

- We end up having a simple and non-iterative formulation of the message-passing:

$$e^{(l)} = (\tilde{A})^l e^{(0)} W^{(l)} W^{(l-1)} W^{(l-2)} \dots$$

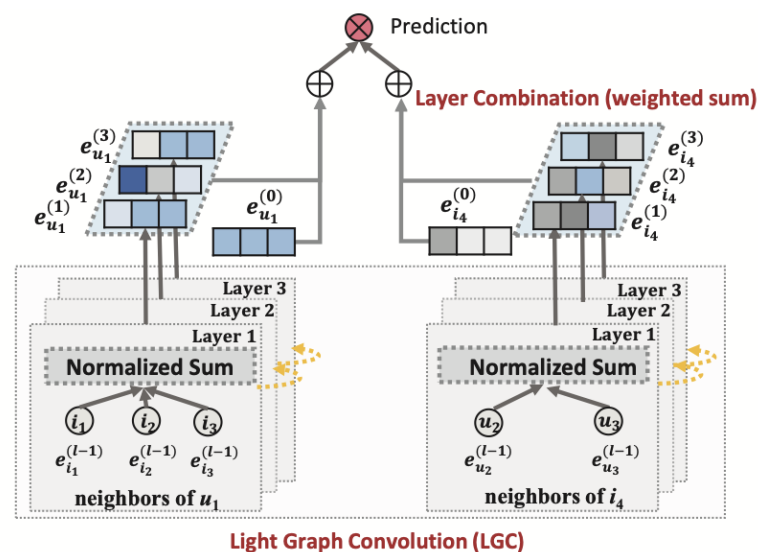
Matrix power



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (8/10)

- Can we remove the linear projection as well?
- Yes! Users' and items' embeddings are already expressive enough!

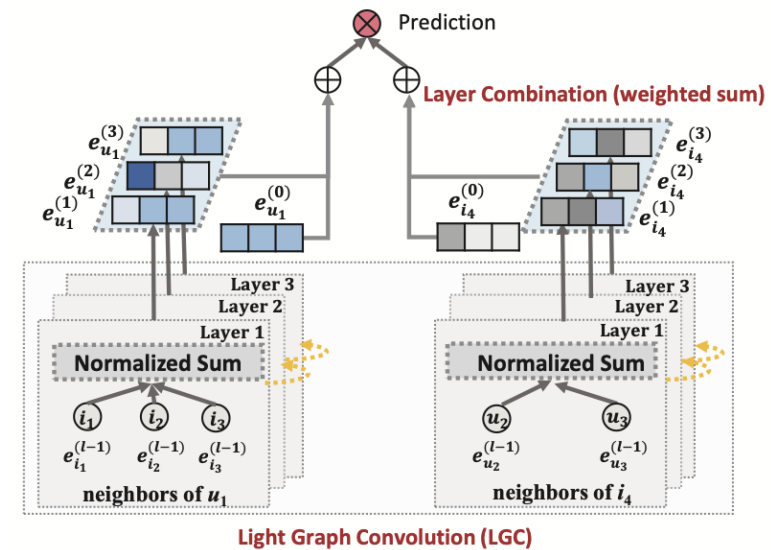
$$e^{(l)} = (\tilde{A})^l e^{(0)}$$



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (9/10)

- The message-passing schema of LightGCN is:

$$e_u^{(l)} = \sum_{i \in N_u} \frac{e_i^{(l-1)}}{\sqrt{|N_u| |N_i|}}$$



LIGHT GRAPH CONVOLUTIONAL NETWORK (LIGHTGCN) (10/10)

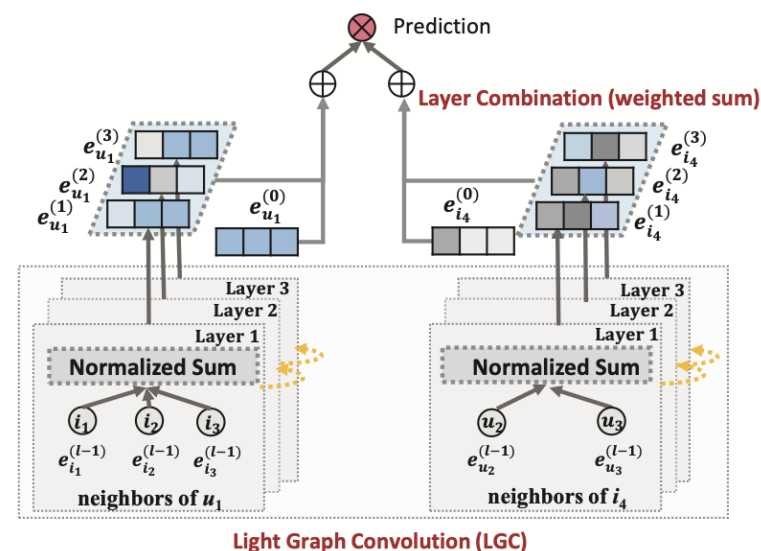
- The message-passing schema of LightGCN is:

$$e_u^{(l)} = \sum_{i \in N_u} \frac{e_i^{(l-1)}}{\sqrt{|N_u| |N_i|}}$$

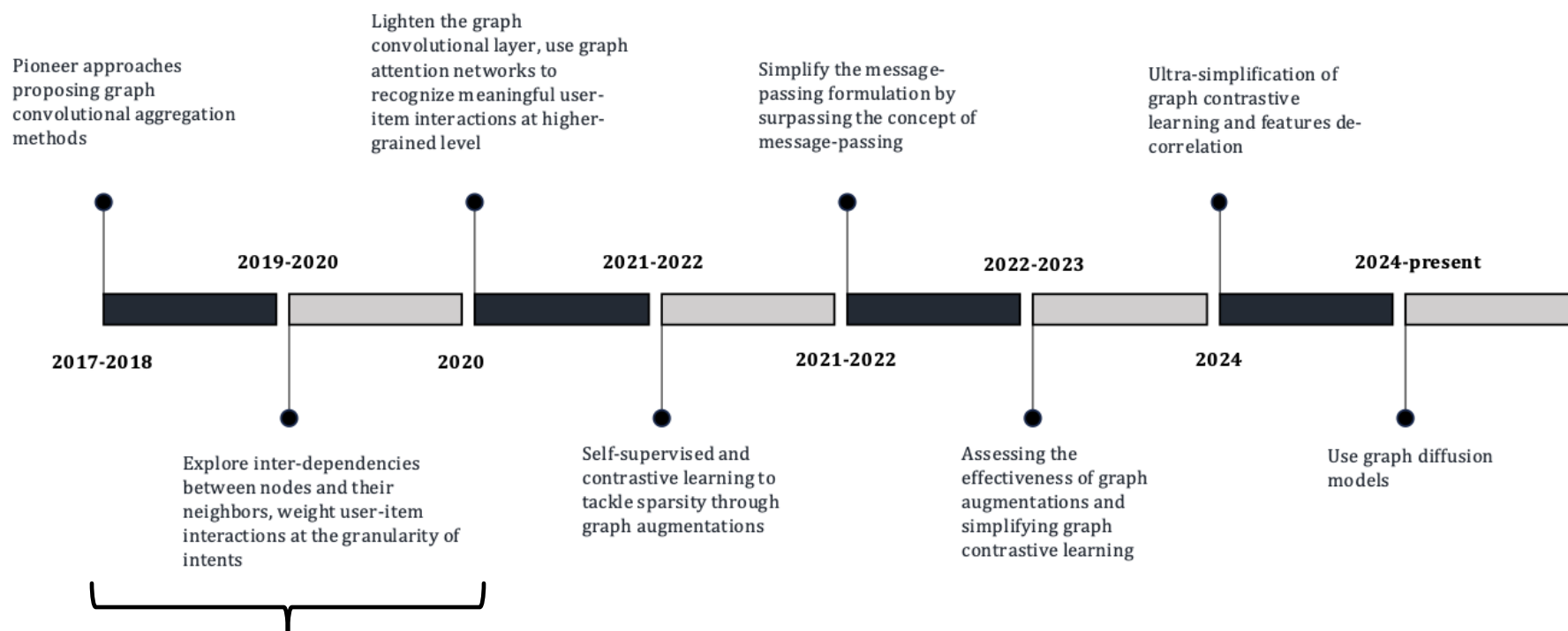
- The layers embeddings are aggregated to obtain:

$$e_u^* = \alpha_0 e_u^{(0)} + \alpha_1 e_u^{(1)} + \dots + \alpha_u^{(L)} e_u^{(L)}$$

↓
Scaling factor $(1/(1-l))$



SO, WHERE ARE WE NOW?



THANKS! ANY QUESTIONS?

DANIELE MALITESTA

Postdoc researcher

CentraleSupélec, Inria, Université Paris-Saclay

Gif-sur-Yvette, France

Email: daniele.malitest@centralesupelec.fr

Website: <https://danielemalitest.github.io>

