Recommender Systems and DGL

Quan Gan

AWS Shanghai AI Lab

July 12, 2019

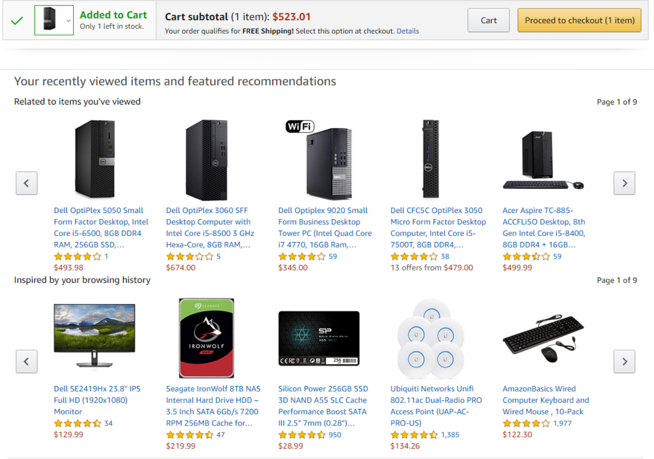
Why Recommendation

Why Recommendation

We don't want our customers to think (hard).

Why Recommendation

Good relevant recommendations make the customers adhere to us.



Why Recommendation

Good relevant recommendations make the customers adhere to us.

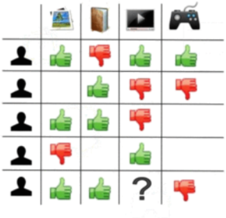
Recommender System: Problem Statement

Image source: Wikipedia

Collaborative Filtering

Collaborative Filtering

$\longrightarrow$



Collaborative Filtering

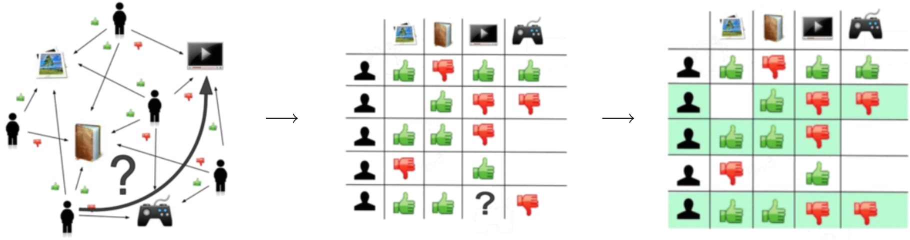


Image source: Wikipedia

Collaborative Filtering: Classical Methods

- **User-based**: Infer how a user i would act to an item j by looking at how users that have similar interactions to user i acted to item j.

Collaborative Filtering: Classical Methods

- **User-based**: Infer how a user i would act to an item j by looking at how users that have similar interactions to user i acted to item j.
  - We have millions of customers.

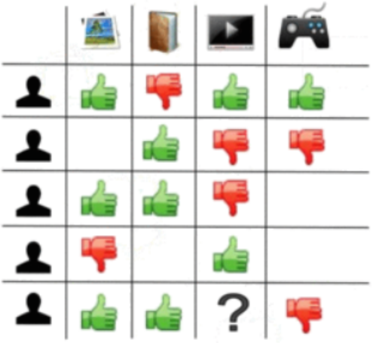
Collaborative Filtering: Classical Methods

- **User-based**: Infer how a user i would act to an item j by looking at how users that have similar interactions to user i acted to item j.
  - We have millions of customers.
  - User profiles change constantly and quickly, requiring frequent rebuilds (which are expensive already).

Collaborative Filtering: Classical Methods

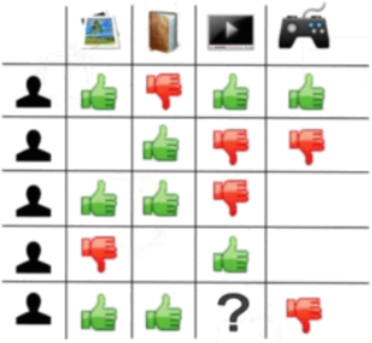
- **User-based**: Infer how a user i would act to an item j by looking at how users that have similar interactions to user i acted to item j.
  - We have millions of customers.
  - User profiles change constantly and quickly, requiring frequent rebuilds (which are expensive already).
  - Not interpretable (can't answer why a user prefers this).

Machine Learning Kicks In



We were "representing" users and items with the items/users that had interactions with them.

Machine Learning Kicks In



We were "representing" users and items with the items/users that had interactions with them.

	Feature 1	Feature 2
User 1	?	?
User 2	?	?
User 3	?	?
User 4	?	?
User 5	?	?

×

	Item 1	Item 2	Item 3	Item 4	Item 5
Feature 1	?	?	?	?	?
Feature 2	?	?	?	?	?

=

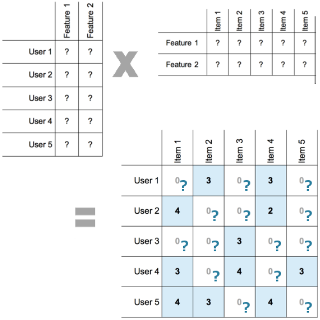
	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0 ?	3	0 ?	3	0 ?
User 2	4	0 ?	0 ?	2	0 ?
User 3	0 ?	0 ?	3	0 ?	0 ?
User 4	3	0 ?	4	0 ?	3
User 5	4	3	0 ?	4	0 ?

Source: Kat Bailey

Can we represent users and items as a set of features?

Latent User/Item Representations

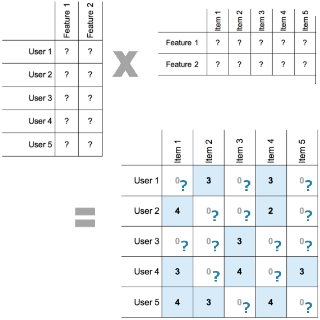
- An item can be described with a set of features (e.g. how sweet some food is).



Source: Kat Bailey

Latent User/Item Representations

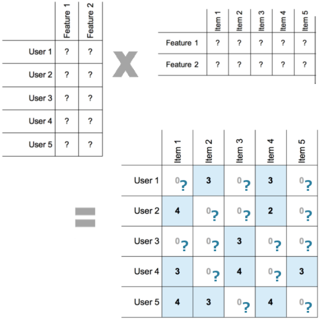
- An item can be described with a set of features (e.g. how sweet some food is).
- A user can be described with preferences of the same set of features (e.g. how much a user likes sweet food).



Source: Kat Bailey

Latent User/Item Representations

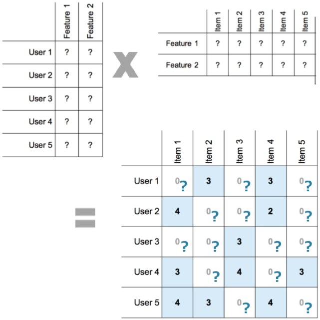
- An item can be described with a set of features (e.g. how sweet some food is).
- A user can be described with preferences of the same set of features (e.g. how much a user likes sweet food).
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Latent User/Item Representations

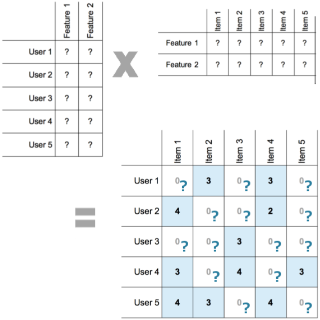
- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with preferences of the same set of features (e.g. how much a user likes sweet food).
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Latent User/Item Representations

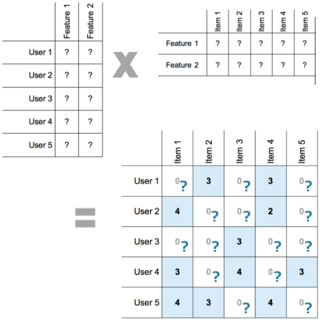
- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Latent User/Item Representations

- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The rating on item j by user i. is defined by $u_i^\top v_j$.

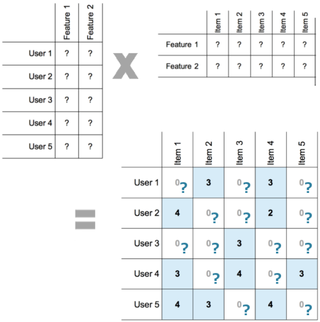


Source: Kat Bailey

Latent User/Item Representations

- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The rating on item j by user i. is defined by $u_i^\top v_j$.
- We minimize

$$\sum_{i,j} \left(r_{i,j} - \left(u_i^\top v_j \right) \right)^2$$

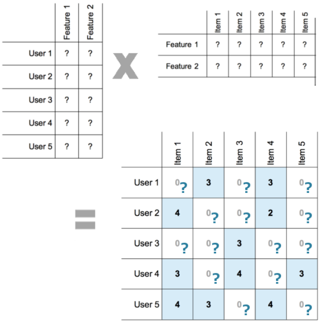


Source: Kat Bailey

Latent User/Item Representations

- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The rating on item j by user i. is defined by $u_i^\top v_j$.
- We minimize

$$\sum_{i,j} \left(r_{i,j} - \left(u_i^\top v_j + b_{u_i} + b_{v_j} \right) \right)^2$$



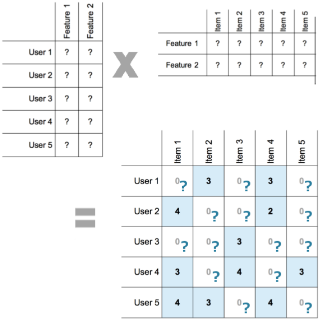
Source: Kat Bailey

Latent User/Item Representations

- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The rating on item j by user i. is defined by $u_i^\top v_j$.
- We minimize

$$\sum_{i,j}\left(r_{i,j} - \left(u_i^\top v_j + b_{u_i} + b_{v_j}\right)\right)^2$$

$$+\alpha\mathcal{R}(U, V)$$



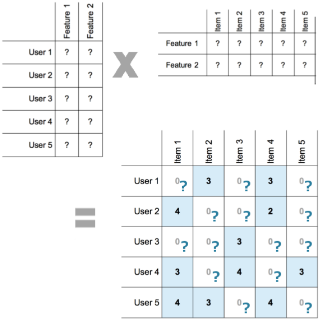
Source: Kat Bailey

Latent User/Item Representations

- An item can be described with a vector v_j (sweet, organic, etc.).
- A user can be described with another vector u_i (likes sweet, likes organic, etc.)
- The rating on item j by user i. is defined by $u_i^\top v_j$.
- We minimize

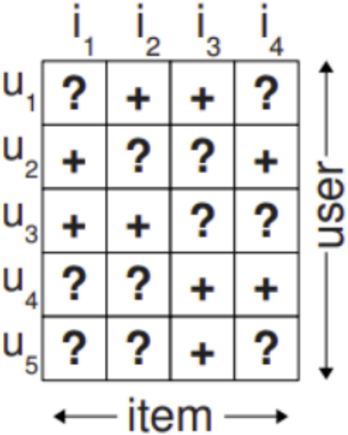
$$\sum_{(i,j)\in\mathcal{B}} \left(r_{i,j} - \left(u_i^\top v_j + b_{u_i} + b_{v_j}\right)\right)^2$$

$$+\alpha\mathcal{R}(U,V)$$



Source: Kat Bailey

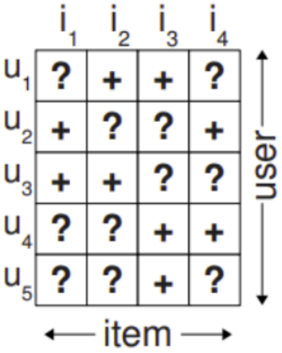
What if we don't have ratings?



Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

Implicit Feedback

- For a given user i, an item being interacted j should have a higher score than another item k which was never being interacted.

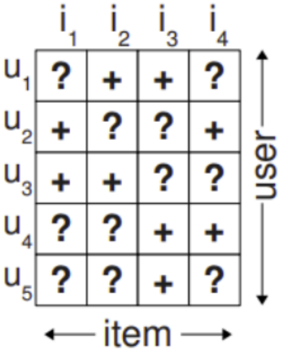


Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

Implicit Feedback

- For a given user i, an item being interacted j should have a higher score than another item k which was never being interacted.
- We maximize

$$\sum_{i,j,k \in I \setminus I_{u_i}} \log \mathrm{sigmoid}\left(\left(u_i^\top v_j - u_i^\top v_k\right)\right)$$



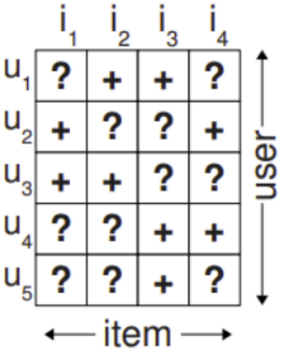
Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

Implicit Feedback

- For a given user i, an item being interacted j should have a higher score than another item k which was never being interacted.
- We maximize

$$\sum_{i,j,k \in I \setminus I_{u_i}} \log \mathrm{sigmoid}\left(\left(u_i^\top v_j - u_i^\top v_k\right)\right)$$

- We usually *sample* one or multiple k when computing gradients (**negative sampling**).
    - Commonly uniformly, but adaptive sampling often helps.



Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

How to get u_i and v_j

- The score function: $u_i^\top v_j$

How to get u_i and v_j

- The score function: $u_i^\top v_j$
- Matrix Factorization: where user and item representations are static and independent of each other (fixing the model).

How to get u_i and v_j

- The score function: $u_i^\top v_j$
- Matrix Factorization: where user and item representations are static and independent of each other (fixing the model).
- RNN To integrate user history, $u_i = f(v_{u,1}, v_{u,2}, \cdots, v_{u,n})$
  - The user representation now depends on his/her previously interacted items.

How to get u_i and v_j

- The score function: $u_i^\top v_j$
- Matrix Factorization: where user and item representations are static and independent of each other (fixing the model).
- RNN To integrate user history, $u_i = f(v_{u,1}, v_{u,2}, \cdots, v_{u,n})$
  - The user representation now depends on his/her previously interacted items.
- Graph-based models to integrate neighboring items/users (in the next slide).
  - The user representation could also depend on behaviors of other users/items.

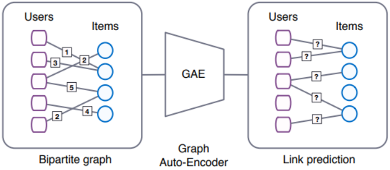
How to get u_i and v_j

- The score function: $u_i^\top v_j$
- Matrix Factorization: where user and item representations are static and independent of each other (fixing the model).
- RNN To integrate user history, $u_i = f(v_{u,1}, v_{u,2}, \cdots, v_{u,n})$
  - The user representation now depends on his/her previously interacted items.
- Graph-based models to integrate neighboring items/users (in the next slide).
  - The user representation could also depend on behaviors of other users/items.
- Can combine with content-based recommendation (i.e. with user and item features).

How to get u_i and v_j

- The score function: $u_i^\top v_j$
- Matrix Factorization: where user and item representations are static and independent of each other (fixing the model).
- RNN To integrate user history, $u_i = f(v_{u,1}, v_{u,2}, \cdots, v_{u,n})$
  - The user representation now depends on his/her previously interacted items.
- Graph-based models to integrate neighboring items/users (in the next slide).
  - The user representation could also depend on behaviors of other users/items.
- Can combine with content-based recommendation (i.e. with user and item features).
- Scoring function can also change (e.g. to bilinear $u_i^\top Q v_j$)

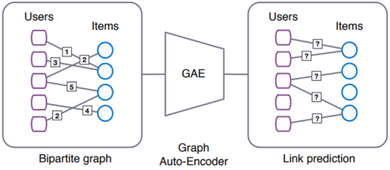
GCMC: Learning u_i and v_j from User-Item Graph



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W_r x_{v_j}$

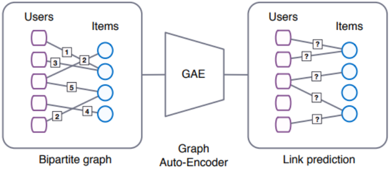
GCMC: Learning u_i and v_j from User-Item Graph



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W_r x_{v_j}$
2. $h_{u_i} = \sigma \left[\mathrm{agg} \left(\sum_{v_j \in \mathcal{N}_{u_i,1}} \mu_{v_j \to u_i, 1}, \cdots, \sum_{v_j \in \mathcal{N}_{u_i,R}} \mu_{v_j \to u_i, R} \right) \right]$

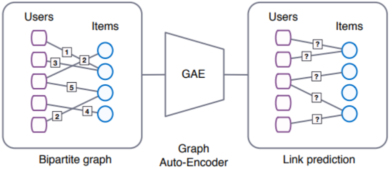
GCMC: Learning u_i and v_j from User-Item Graph



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W_r x_{v_j}$
2. $h_{u_i} = \sigma \left[\mathrm{agg} \left(\sum_{v_j \in \mathcal{N}_{u_i,1}} \mu_{v_j \to u_i, 1}, \cdots, \sum_{v_j \in \mathcal{N}_{u_i,R}} \mu_{v_j \to u_i, R} \right) \right]$
3. $u_i = \sigma(W_u h_{u_i})$ and similarly we compute v_j

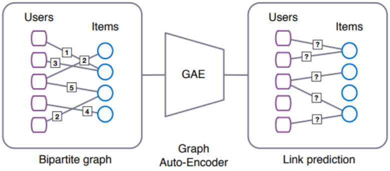
GCMC: Learning u_i and v_j from User-Item Graph



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W_r x_{v_j}$
2. $h_{u_i} = \sigma \left[\mathrm{agg} \left(\sum_{v_j \in \mathcal{N}_{u_i,1}} \mu_{v_j \to u_i, 1}, \cdots, \sum_{v_j \in \mathcal{N}_{u_i,R}} \mu_{v_j \to u_i, R} \right) \right]$
3. $u_i = \sigma(W_u h_{u_i})$ and similarly we compute v_j
4. $p(\hat{M}_{ij} = r) = \mathrm{softmax}(u_i^\top Q_r v_j)$

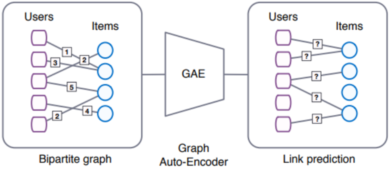
GCMC: Learning u_i and v_j from User-Item Graph



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W_r x_{v_j}$
2. $h_{u_i} = \sigma \left[\mathrm{agg} \left(\sum_{v_j \in \mathcal{N}_{u_i,1}} \mu_{v_j \to u_i, 1}, \cdots, \sum_{v_j \in \mathcal{N}_{u_i,R}} \mu_{v_j \to u_i, R} \right) \right]$
3. $u_i = \sigma(W_u h_{u_i})$ and similarly we compute v_j
4. $p(\hat{M}_{ij} = r) = \mathrm{softmax}(u_i^\top Q_r v_j)$
5. When new interactions are added, just re-run the forward pass on the new graph to get new u_i and v_j.

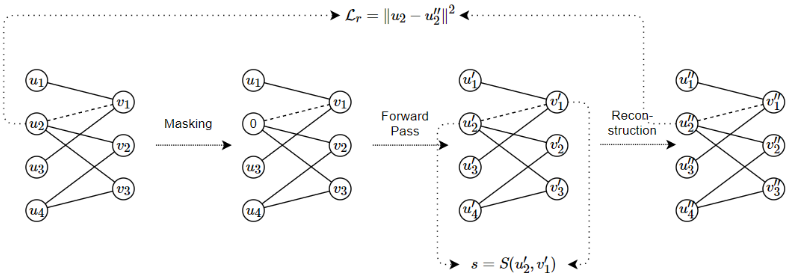
Simplifying GCMC to GraphSAGE



Source: *Graph Convolutional Matrix Completion*, van den Berg et al. 2017

1. $\mu_{v_j \to u_i, r} = \frac{1}{c_{u_i v_j}} W x_{v_j}$
2. $h_{u_i} =$
   $\sigma \left[\text{agg} \left(\sum_{v_j \in \mathcal{S}(\mathcal{N}_{u_i,1})} \mu_{v_j \to u_i, 1}, \cdots, \sum_{v_j \in \mathcal{S}(\mathcal{N}_{u_i,R})} \mu_{v_j \to u_i, R} \right) \right]$
3. $u_i = \sigma(W_u h_{u_i})$ and similarly we compute v_j
4. $r_{i,j} = u_i^\top v_j$ to predict rating
5. When new interactions are added, just re-run the forward pass on the new graph to get new u_i and v_j.

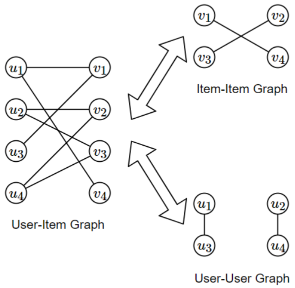
Learning u_i and v_j with Star-GCN



- Vanilla GCMC can't deal with new users/items without features (but with a few interactions).
- STAR-GCN
  - "Mask" the user/item embedding to 0 as if it is new.
  - Reconstruct the embedding after the forward pass and reconstruction pass.

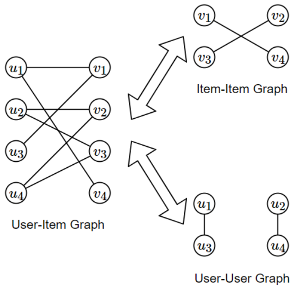
Learning v_j and u_i from Item-Item and User-User Graph

- Decompose the user-item graph into user-user graph and item-item graph.



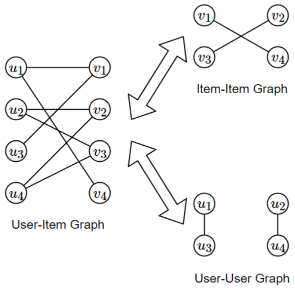
Learning v_j and u_i from Item-Item and User-User Graph

- Decompose the user-item graph into user-user graph and item-item graph.
- Get u_i with a Graph Convolutional Network on user-user graph and v_j on item-item graph.



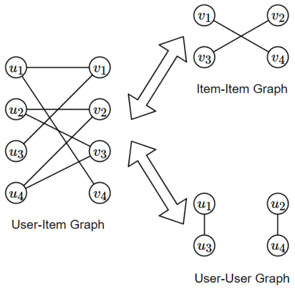
Learning v_j and u_i from Item-Item and User-User Graph

- Decompose the user-item graph into user-user graph and item-item graph.
- Get u_i with a Graph Convolutional Network on user-user graph and v_j on item-item graph.
- Compute $r_{i,j} = u_i^\top v_j$



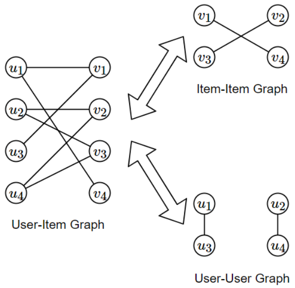
Learning v_j and u_i from Item-Item and User-User Graph

- Decompose the user-item graph into user-user graph and item-item graph.
- Get u_i with a Graph Convolutional Network on user-user graph and v_j on item-item graph.
- Compute $r_{i,j} = u_i^\top v_j$
- u_i and v_j can be learned with
    - Direct neighbor sampling (GraphSAGE)
    - Random-walk based neighbor sampling (PinSAGE)



Learning v_j and u_i from Item-Item and User-User Graph

- Decompose the user-item graph into user-user graph and item-item graph.
- Get u_i with a Graph Convolutional Network on user-user graph and v_j on item-item graph.
- Compute $r_{i,j} = u_i^\top v_j$
- u_i and v_j can be learned with
  - Direct neighbor sampling (GraphSAGE)
  - Random-walk based neighbor sampling (PinSAGE)
- When new interactions are added, just re-run the forward pass on the new graph to get new u_i and v_j.



Other Meaningful Aspects to Consider

- **Cold-start**: What if we have *new* users and items coming in, with few to no historical interactions?

Other Meaningful Aspects to Consider

- **Cold-start**: What if we have *new* users and items coming in, with few to no historical interactions?
- **Bias correction**: The training dataset usually comes from the result of a *previous recommender system*. How to mitigate the bias?

Other Meaningful Aspects to Consider

- **Cold-start**: What if we have *new* users and items coming in, with few to no historical interactions?
- **Bias correction**: The training dataset usually comes from the result of a *previous recommender system*. How to mitigate the bias?
- **Diversity**: Always recommending the same items (or even the same kind of item) to a user would make him/her feel *bored*.

Other Meaningful Aspects to Consider

- **Cold-start**: What if we have *new* users and items coming in, with few to no historical interactions?
- **Bias correction**: The training dataset usually comes from the result of a *previous recommender system*. How to mitigate the bias?
- **Diversity**: Always recommending the same items (or even the same kind of item) to a user would make him/her feel *bored*.
- **Fraud**: How to detect and deal with fabricated explicit feedbacks (e.g. fake ratings and reviews)?

Coding Session

GraphSAGE on bipartite user-item graph.