Recommender Systems and DGL

Quan Gan

AWS Shanghai AI Lab

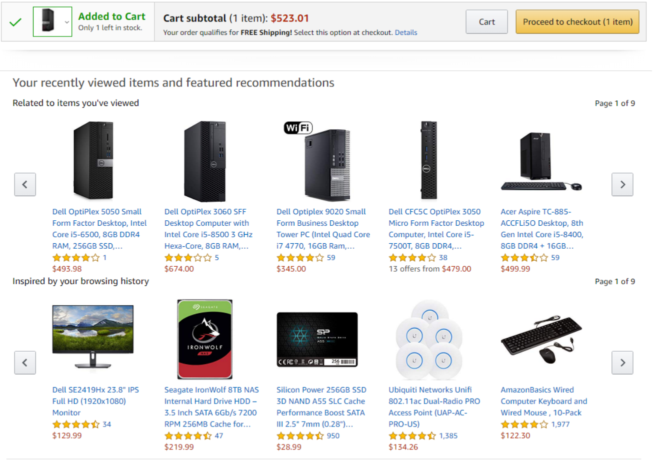
July 12, 2019

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.

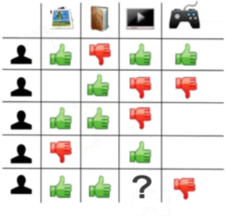
Recommender System: Problem Statement

Image source: Wikipedia

Collaborative Filtering

Collaborative Filtering

$\longrightarrow$



Collaborative Filtering

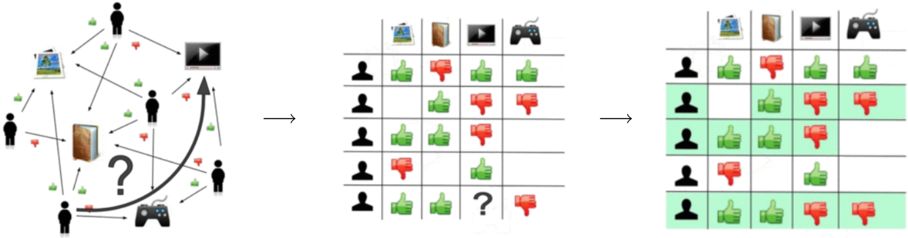
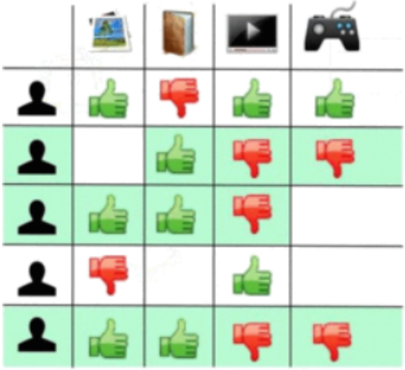


Image source: Wikipedia

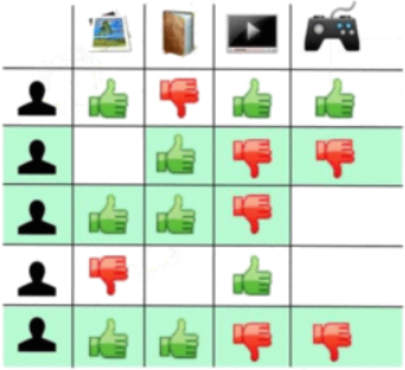
Collaborative Filtering: Classical Methods

- **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.



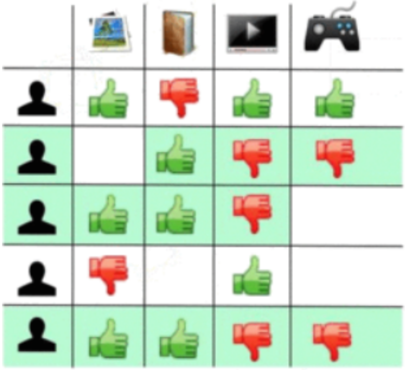
Collaborative Filtering: Classical Methods

- **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).



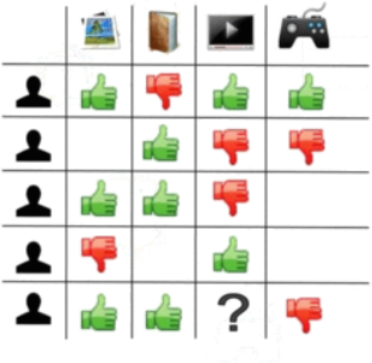
Collaborative Filtering: Classical Methods

- **Item-based**: Infer how a user i would act to an item j by looking at how items that have similar interactions to item j were being acted by user i.
    - Amazon used to have fewer items than users.

Collaborative Filtering: Classical Methods

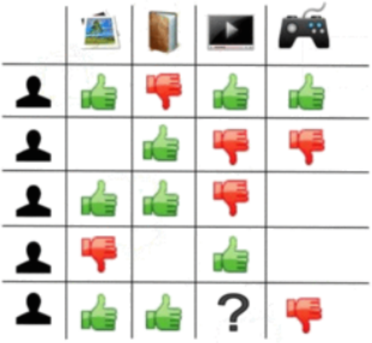
- **Item-based**: Infer how a user i would act to an item j by looking at how items that have similar interactions to item j were being acted by user i.
  - Amazon used to have fewer items than users.
  - Now we also have millions of items.

Machine Learning Kicks In

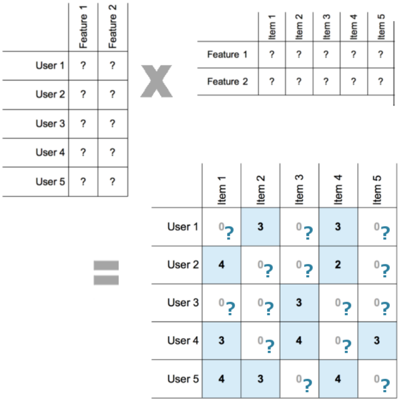


We were "representing" users and items with the items/users that had interactions with them.

Machine Learning Kicks In



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

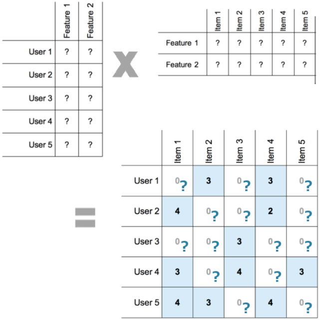


Source: Kat Bailey

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

Latent Factor Model

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



Source: Kat Bailey

Latent Factor Model

- 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).

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

X

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

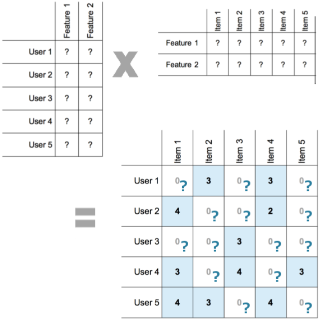
=

	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

Latent Factor Model

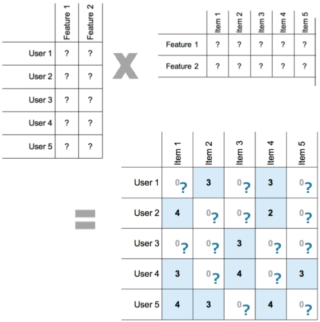
- An item can be described with a set of features (e.g. how sweet some food is).
- A user can be described with preferences of the same set of features (e.g. how much a user likes sweet food).
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

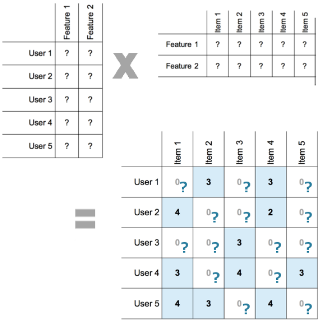
- An item can be described with a vector v_j.
- A user can be described with preferences of the same set of features (e.g. how much a user likes sweet food).
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

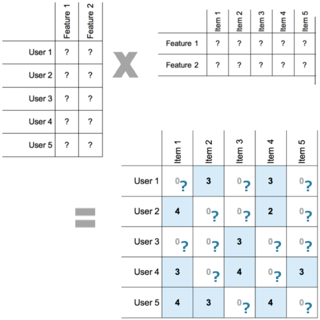
- An item can be described with a vector v_j.
- A user can be described with another vector u_i
- The interaction is defined by how well the item features match the user preferences.



Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- An item can be described with a vector v_j.
- A user can be described with another vector u_i
- The rating on item j by user i. is defined by $u_i^\top v_j$.

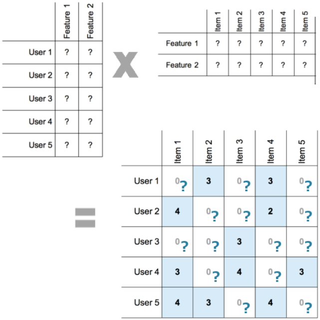


Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

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

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

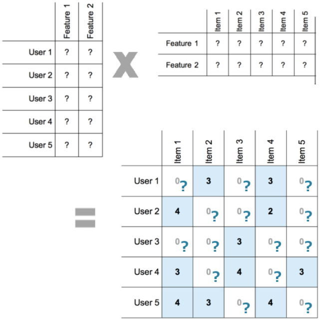


Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

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

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



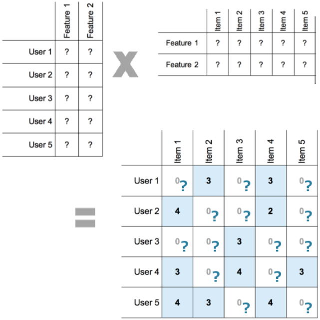
Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- We minimize

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

- What if the vectors are of **very** high dimensions?



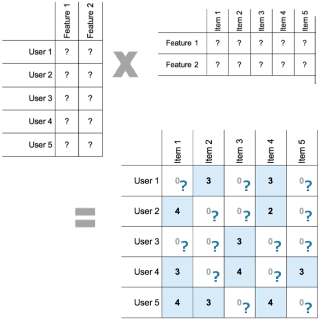
Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- We minimize

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

- What if the vectors are of **very** high dimensions?
  - Use low-dimensional u_i and v_j.

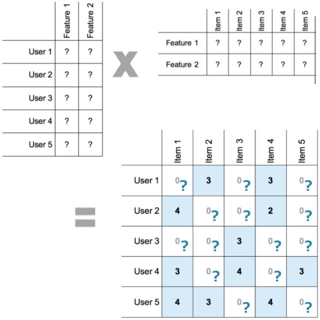


Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- We minimize

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



Source: Kat Bailey

- What if the vectors are of **very** high dimensions?
  - Use low-dimensional u_i and v_j.
- What if one product receives only one but very high rating?

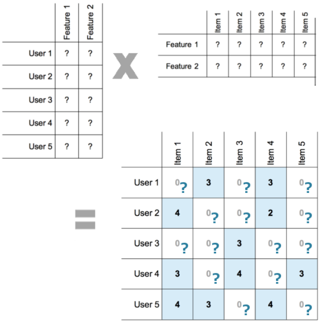
Matrix Factorization (Explicit Feedback)

- We minimize

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

$$+\alpha \left(\|U\|_F^2 + \|V\|_F^2 \right)$$

- What if the vectors are of **very** high dimensions?
  - Use low-dimensional u_i and v_j.
- What if one product receives only one but very high rating?
  - Penalize the magnitude of parameters.



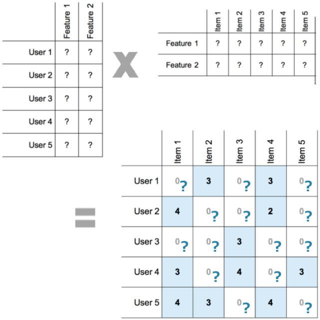
Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- We minimize

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

$$+\alpha \left(\|U\|_F^2 + \|V\|_F^2 \right)$$



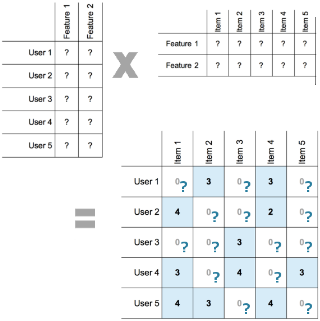
Source: Kat Bailey

Matrix Factorization (Explicit Feedback)

- We minimize

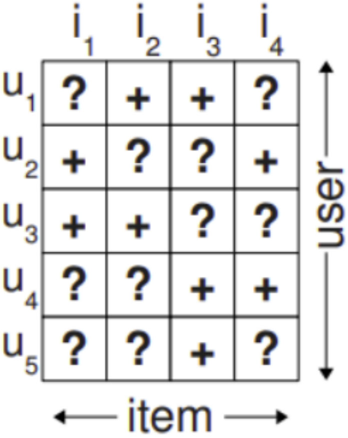
$$\sum_{(i,j)\in\mathcal{B}} \left(r_{i,j} - \left(u_i^\top v_j + b_{u_i} + b_{u_j}\right)\right)^2$$
$$+\alpha\left(\|U\|_F^2 + \|V\|_F^2\right)$$

- Optimize with Stochastic Gradient Descent
  - Each time we sample a minibatch of interactions, and compute the gradient from this minibatch only.



Source: Kat Bailey

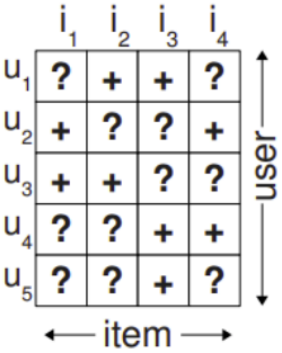
What if we don't have ratings?



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

Matrix Factorization (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.

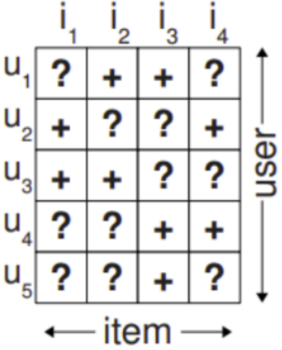


Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

Matrix Factorization (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

$$\sum_{i,j,k \in I \setminus I_{u_i}} \log \frac{1}{1 + \exp\left(u_i^\top v_k - u_i^\top v_j\right)}$$



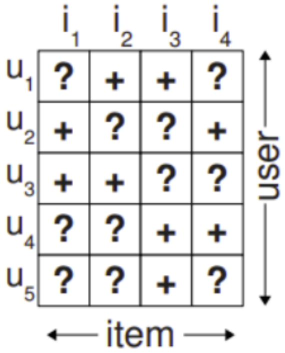
Source: *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Rendle et al. 2012

Matrix Factorization (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

$$\sum_{i,j,k \in I \setminus I_{u_i}} \log \frac{1}{1 + \exp\left(u_i^\top v_k - u_i^\top v_j\right)}$$

- 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

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)?

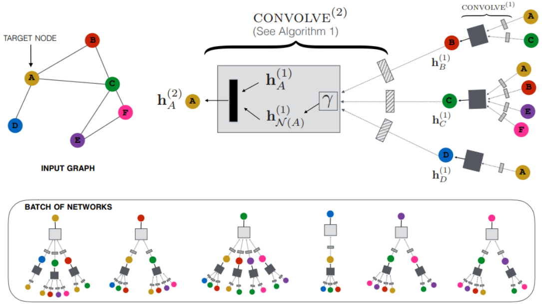
Extensions of Matrix Factorization

- The score function of vanilla MF: $u_i^\top v_j$ 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).
- Can combine both explicit and implicit feedbacks.

Extensions of Matrix Factorization (with Graph-based Models)

- Graph Convolutional Networks on an item copurchase graph
  - When new interactions are added, the copurchase graph and the item representation would also change as well.
  - GCN can also be replaced with other models such as GraphSAGE or PinSAGE (*Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems*, You et al., 2019)
- GCN on the user-item bipartite graph itself
  - With one layer, becomes GCMC (van den Berg et al., 2017)
  - With neighbor sampling, becomes GraphSAGE.
  - Latest work include STAR-GCN (Zhang et al., 2019) which also deal with cold start problems.

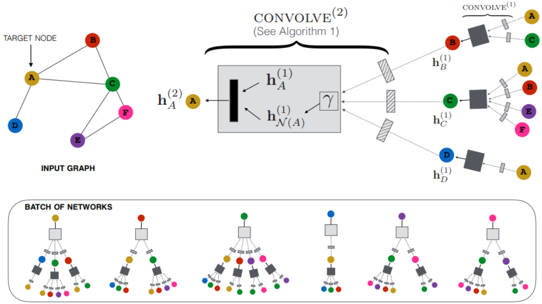
Learning Item Representations with GCN



Source: *Graph Convolutional Neural Networks for Web-Scale Recommender Systems*, Ying et al. 2018

- The graph is a copurchase graph.

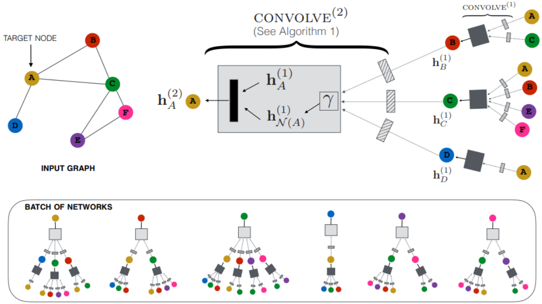
Learning Item Representations with GCN



Source: *Graph Convolutional Neural Networks for Web-Scale Recommender Systems*, Ying et al. 2018

- The graph is a copurchase graph.
- Item features can be projected before GCN.

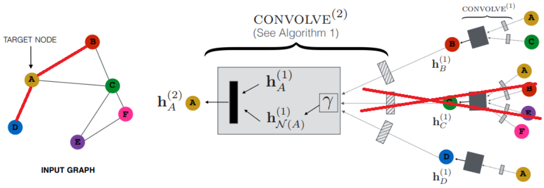
Learning Item Representations with GCN



Source: *Graph Convolutional Neural Networks for Web-Scale Recommender Systems*, Ying et al. 2018

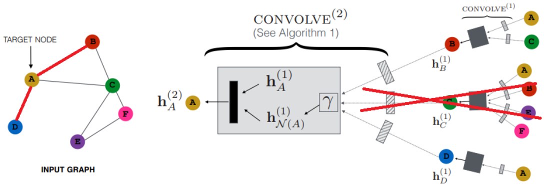
- The graph is a copurchase graph.
- Item features can be projected before GCN.
- During inference, when new copurchases/items appear, we can just recompute the embeddings on the new graph with trained parameters.

Learning Item Representations with GCN



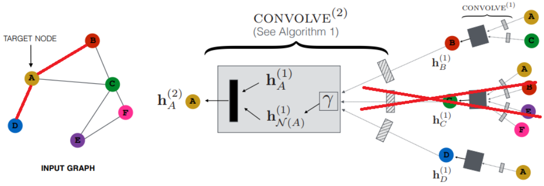
- GraphSAGE - neighbor sampling

Learning Item Representations with GCN



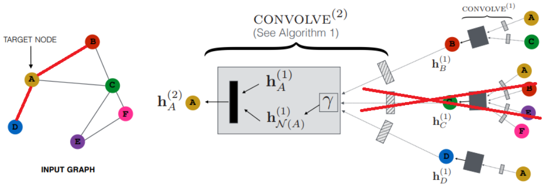
- GraphSAGE - neighbor sampling
- PinSAGE - Neighbors determined by "top-K most frequent nodes visited by random walk with restarts"

Learning Item Representations with GCN



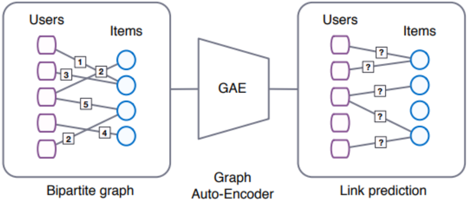
- GraphSAGE - neighbor sampling
- PinSAGE - Neighbors determined by "top-K most frequent nodes visited by random walk with restarts"
  - In the example: if $K = 2$, then B could also be a neighbor of D.

Learning Item Representations with GCN



- GraphSAGE - neighbor sampling
- PinSAGE - Neighbors determined by "top-K most frequent nodes visited by random walk with restarts"
  - In the example: if $K = 2$, then B could also be a neighbor of D.
  - For concrete details of the random walk algorithm, please refer to *Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time*, Eksombatchai et al., 2017

Learning Both User and Item Representations with GCMC



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

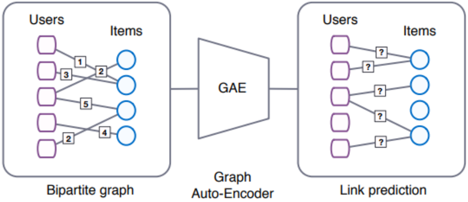
$$\mu_{j \to i, r} = \frac{1}{c_{ij}} W_r x_j$$

$$h_i = \sigma \left[\mathrm{accum} \left(\sum_{j \in \mathcal{N}_{i,1}} \mu_{j \to i, 1}, \cdots, \sum_{j \in \mathcal{N}_{i,R}} \mu_{j \to i, R} \right) \right]$$

$$u_i = \sigma(W h_i)$$

$$p(\hat{M}_{ij} = r) = \mathrm{softmax}(u_i^\top Q_r v_j)$$

Learning Both User and Item Representations with GCMC



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

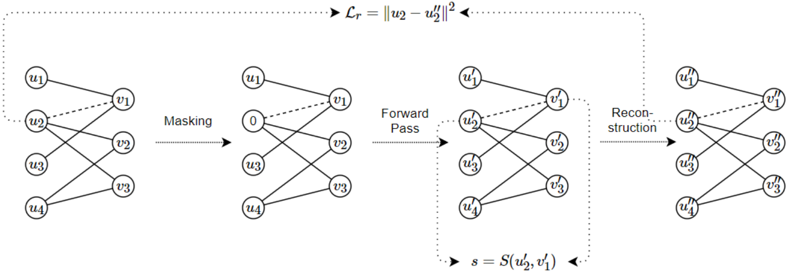
$$\mu_{j\to i,r} = \frac{1}{c_{ij}} W_r x_j$$

$$h_i = \sigma\left[\mathrm{accum}\left(\sum_{j\in\mathcal{S}(\mathcal{N}_{i,1})} \mu_{j\to i,1}, \cdots, \sum_{j\in\mathcal{S}(\mathcal{N}_{i,R})} \mu_{j\to i,R}\right)\right]$$

$$u_i = \sigma(Wh_i)$$

$$p(\hat{M}_{ij} = r) = \mathrm{softmax}(u_i^\top Q_r v_j)$$

Learning Both User and Item Representations with Star-GCN



- Vanilla GCMC can't deal with new users/items without features and 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.

Coding Session

GraphSAGE on bipartite user-item graph.