CIS 4190/5190: Lec 21 Mon Nov 18, 2024

Introduction to ML for Recommendations (And Graph Neural Networks)

Robot Image Credit: Viktoriya Sukhanova © 123RF.com

Recommender Systems are Everywhere

What media to consume

prime video

Disney+

NETFLIX

MUSIC

hulu

pandora

YouTube

Image: https://medium.com/@PhilAutelitano/pitching-your-idea-to-netflix-and-hulu-its-like-a-book-people-1e173430d0c

Recommender Systems are Everywhere

What news you see

Google News

reddit

Image: http://www.sapientis.co.za/services/profile-collage/

Recommender Systems are Everywhere

What products to buy

amazon.com

B&H PHOTO - VIDEO - PRO AUDIO

ebay

THE HOME DEPOT

Etsy

audible

LOWE'S

Walmart

Image: https://www.curiouskeeda.com/business/10-weird-products-available-on-amazon/

Recommender Systems are Everywhere

Who to date

match

coffee meets bagel

bumble

POF PlentyOfFish

Hinge

okcupid

tinder

eharmony

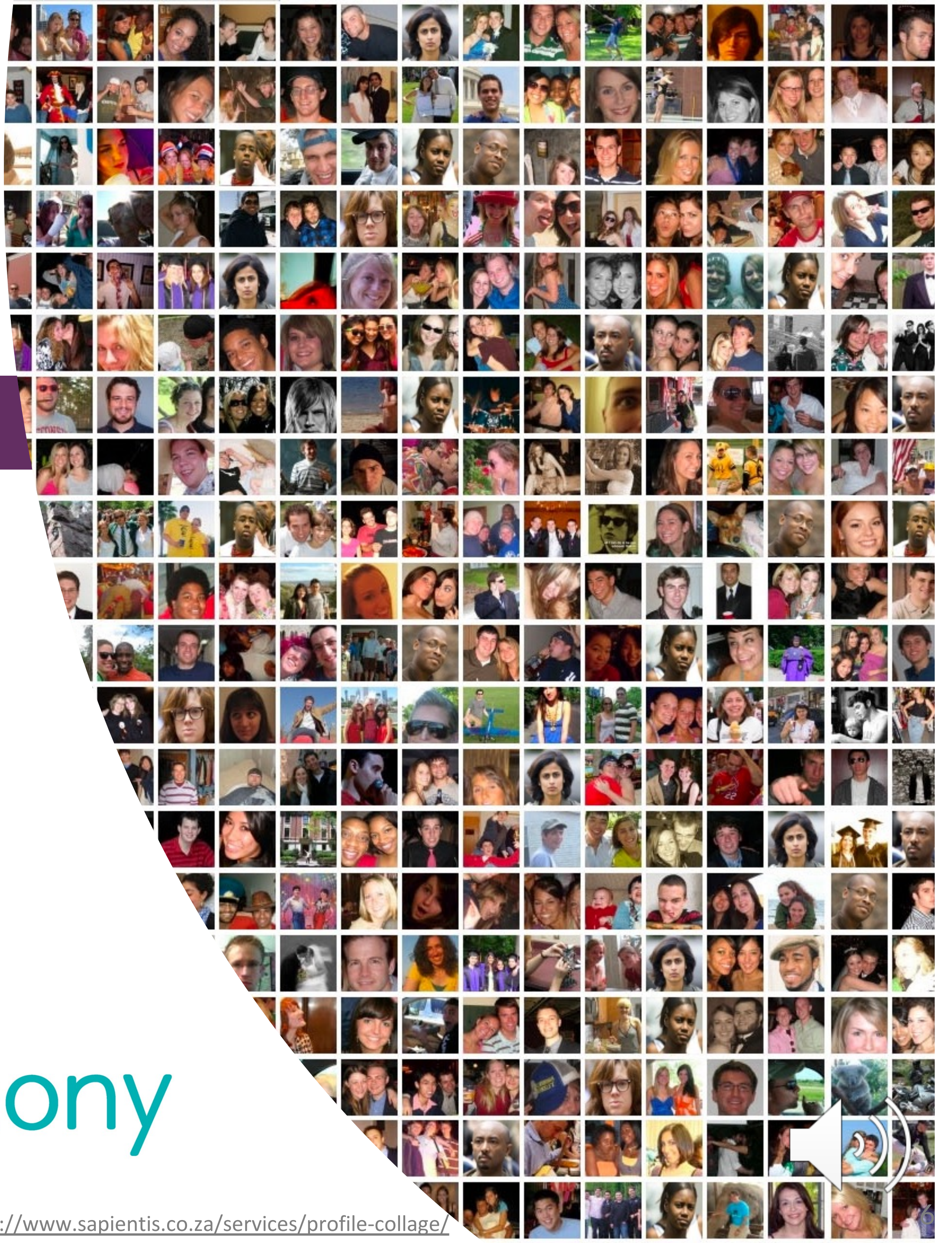


Image: http://www.sapientis.co.za/services/profile-collage/

Real Impact

Recommendations account for:

- 75% of movies watched on Netflix [1]
- 60% YouTube video clicks [2]
- 35% of Amazon sales [1]

Approximately 40% of committed relationships begin online [3]

Sources:

1. McKinsey & Company (Oct 2013): https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers [Note: non-authoritative source; estimates only]
2. J. Davidson, et al. (2010). The YouTube video recommendation system. Proc. of the 4th ACM Conference on Recommender systems (RecSys). doi.org/10.1145/1864708.1864770
3. M. Rosenfeld, et al. (2019). Disintermediating your friends: How online dating in the United States displaces other ways of meeting. Proc. National Academy of Sciences 116(36).

Image: Harryarts - https://www.freepik.com/free-vector/colorful-paint-splashes_1098069.htm

Stores Group Products Based on Consumer Buying Habits

Products that are commonly purchased together are displayed together

Image: https://fitsmallbusiness.com/visual-merchandising-guide/

Website Advertisements are Based on Our Online Activity

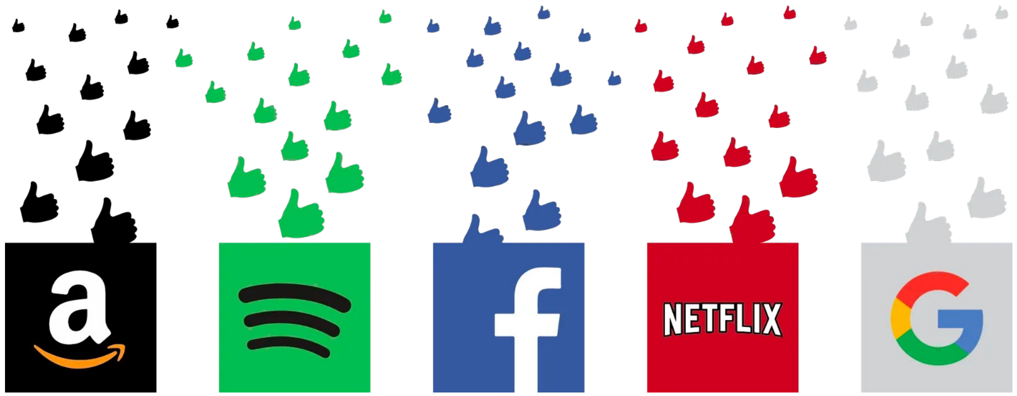
Users are tracked across websites to build consumer profiles

Image:https://frozenfire.com/ad-blocker-plus-digital-ads-future-internet/

Collaborative Filtering

Popularity-Based Recommendations

- Just recommend whatever is currently popular
- Simple and often quite effective

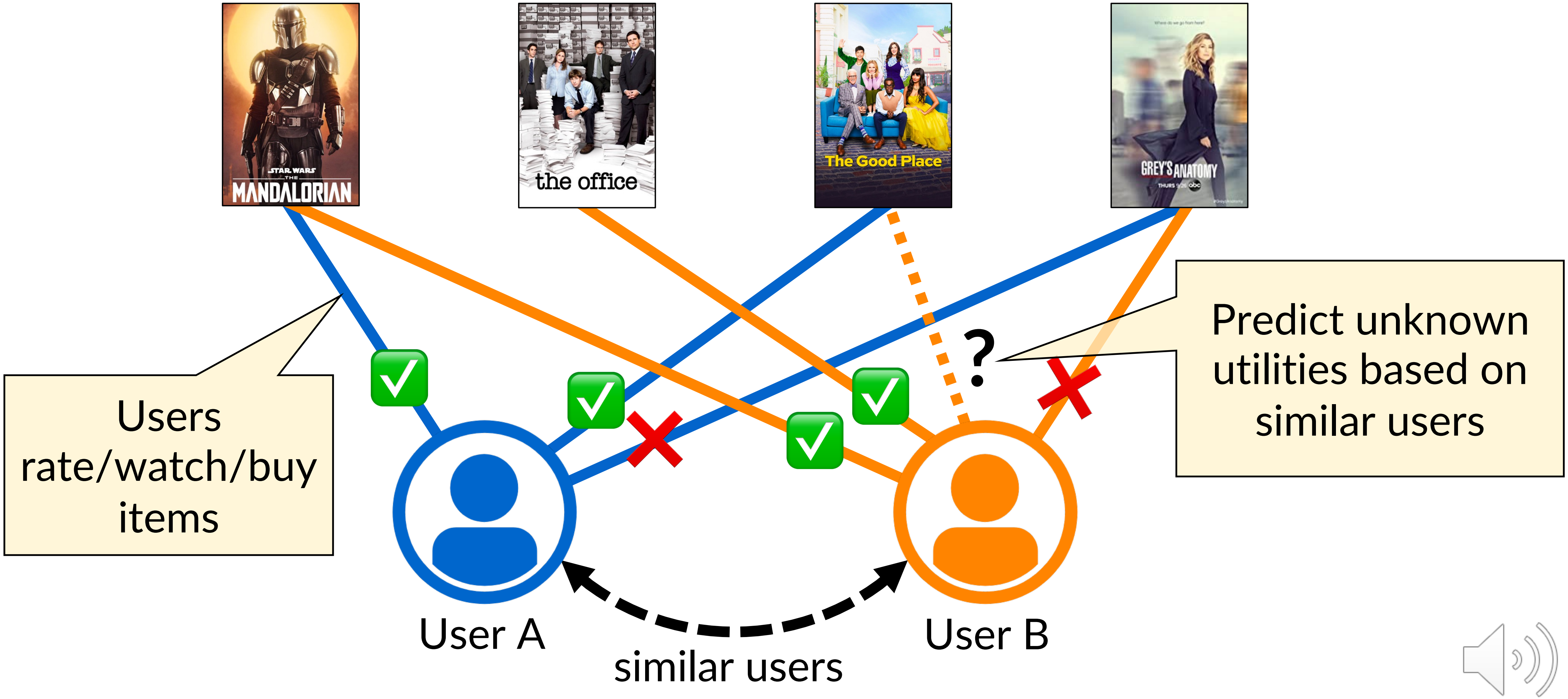


- This uses no information at all about the user!
  - Could improve by tailoring to the user: e.g. their geographical location, age, etc.

Image: https://thedatascientist.com/right-way-recommender-system-startup/

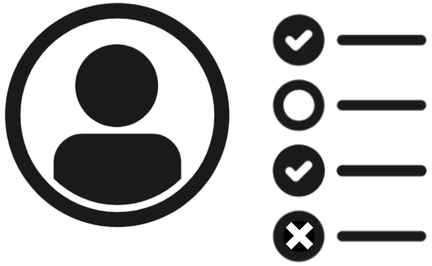
The Recommendation Problem

Predict a user's rating for an item that they have not yet tried



Collaborative Filtering Steps

Collect user-item utilities



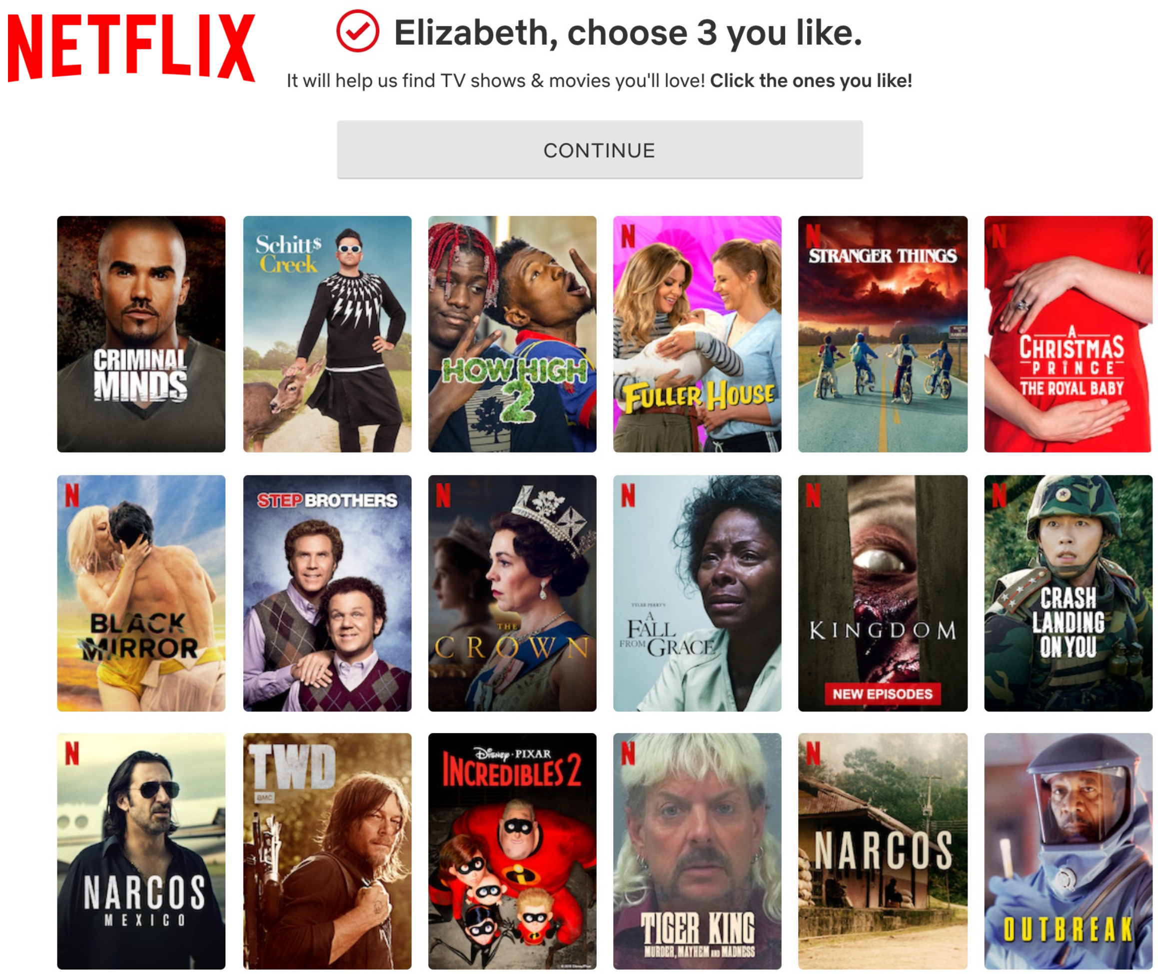
Identify similar users

Predict unknown item utilities
based on other similar users

Measuring User-Item Utilities

Utilities can be based on:

- Explicit rating
- Implicit rating
  - Inferred from user activity
  - e.g., User stops watching movie after 15 minutes
  - e.g., User repeatedly clicks on a particular dating profile



For now, we are not considering user or item attributes/content

e.g. genres and cast of movies/TV shows

Obtaining User Feedback

Low — Feedback Strength — High

- Viewing profile, images, etc.
- Marking as a "favorite"
- Swiping left/right
- Messaging a person
- "Liking" a profile
- "Winking" at a person
- Conversation

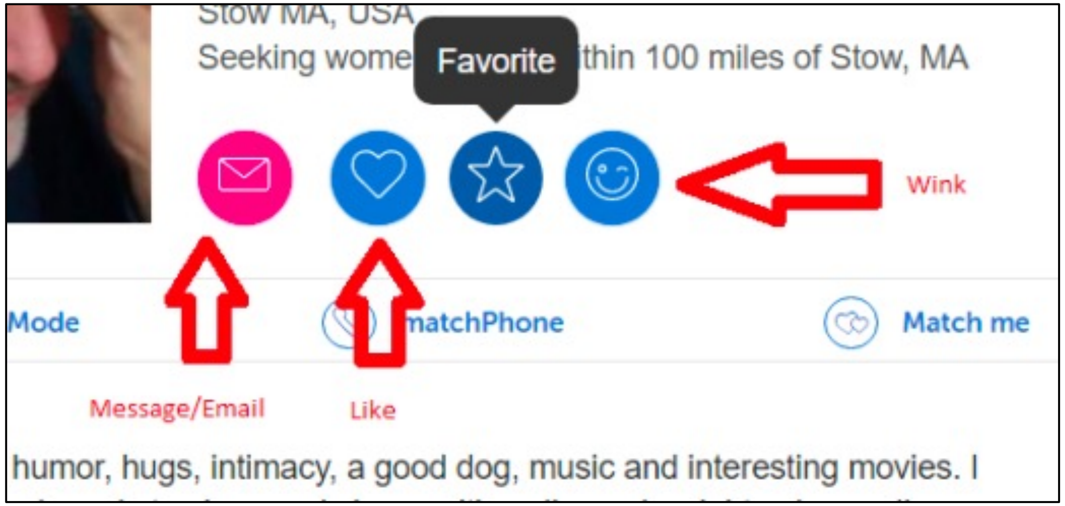


Image: https://medium.com/icebrkr/breaking-down-the-complexity-issues-on-match-com-c0087dbddea2

Challenges with Measuring Utility

Ratings can be misleading

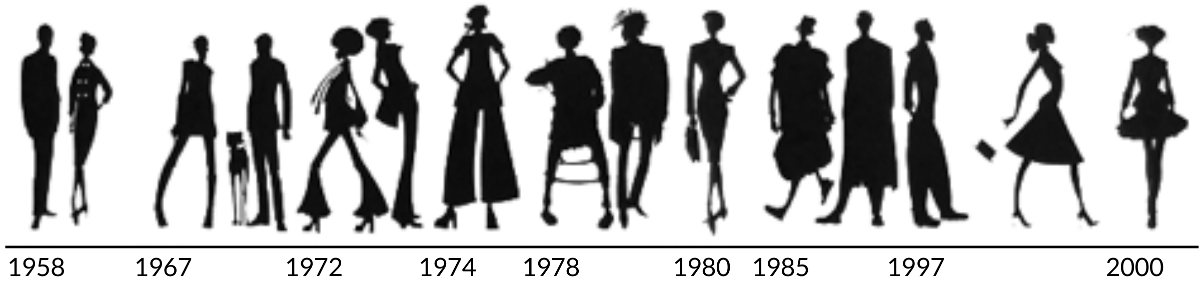
- Sometimes users more likely to rate if experience is especially good or bad
- Users may have different scales
  - Can normalize user ratings, but their "scaling" might not even be linear.
- May need to consider credibility of individual raters (history of ratings)
- Bot farms may skew results through adversarial behavior

Reordered from class to improve flow

Handling Time-Varying Preferences

Aspects of recommendations change over time:

- User preferences change
- Popularity of items change



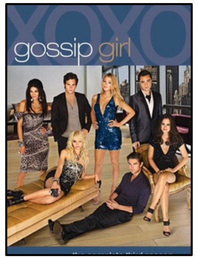
Potential solution: weight more recent measurements over the past

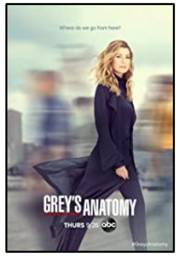
- Could use an exponentially weighted moving average
  - Decay old utilities. For example:
    - If user u has not newly rated item i at time t: $x_{u,i}^{t+1} \leftarrow 0.95\, x_{u,i}^{t}$
    - (Otherwise, set $x_{u,i}$ to the new rating, of course.)

Reordered from class to improve flow

Image: https://finalfashion.ca/the-indefinable-decades/

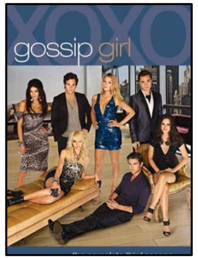
User-Item Utility Matrix

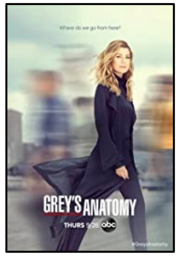




	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	…
Grace	4	5	4	1	5	3	…
Eric	1	4	5	1	5	3	…
Haren	5	5	5	1	3	4	…
Sai	1	2	5	4	3	5	…
Siyan	3	1	1	3	4	5	…
Nikhil	2	3	4	2	2	2	…
Felix	1	1	1	5	2	2	…

User-Item Utility Matrix





	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace	4	5	4	1	5	3	...
Eric	1	4	5	1	5	3	
Haren	5	5	5				
Sai	1	2	5				
Siyan	3	1	1	3	4	5	...
Nikhil	2	3	4	2	2	2	...
Felix	1	1	1	5	2	2	...

x_u

Let x_u be the item utilities for user u

But of course, we don't have all the ratings. We will return to this soon!

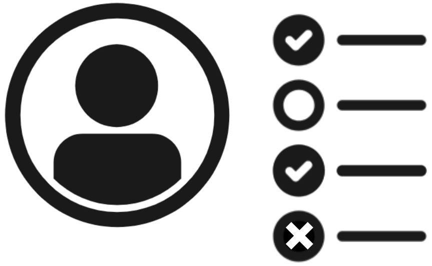
Collaborative Filtering

- **Given:**
  - User-Item Utility Matrix $X_{u,i} = \begin{cases} \text{rating}_{u,i} & \text{if user}_u \text{ rated product}_i \\ \text{N/A} & \text{otherwise} \end{cases}$
  - Assume fixed set of n users and m products
  - **Not given any information about the products!**

- **Problem:** Predict what $X_{u,i}$ would be if it is observed
  - Not quite supervised or unsupervised learning!

Collaborative Filtering Steps

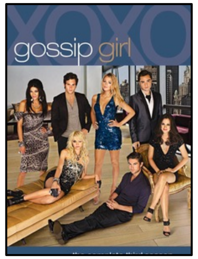
Collect user-item utilities

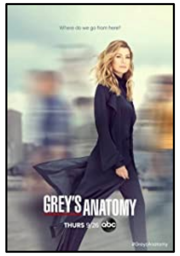


Identify similar users

Predict unknown item utilities
based on other similar users

Correlations Between Users

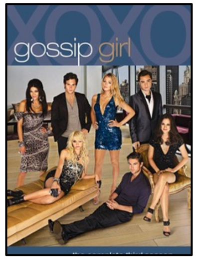


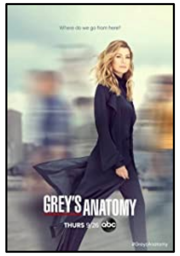


Similar users

	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace	4	5	4	1	5	3	...
Eric	1	4	5	1	5	3	...
Haren	5	5	5	1	3	4	...
Sai	1	2	5	4	3	5	...
Siyan	3	1	1	3	4	5	...
Nikhil	2	3	4	2	2	2	...
Felix	1	1	1	5	2	2	...

Correlations Between Users





Dissimilar users

	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	…
Grace	4	5	4	1	5	3	…
Eric	1	4	5	1	5	3	…
Haren	5	5	5	1	3	4	…
Sai	1	2	5	4	3	5	…
Siyan	3	1	1	3	4	5	…
Nikhil	2	3	4	2	2	2	…
Felix	1	1	1	5	2	2	…

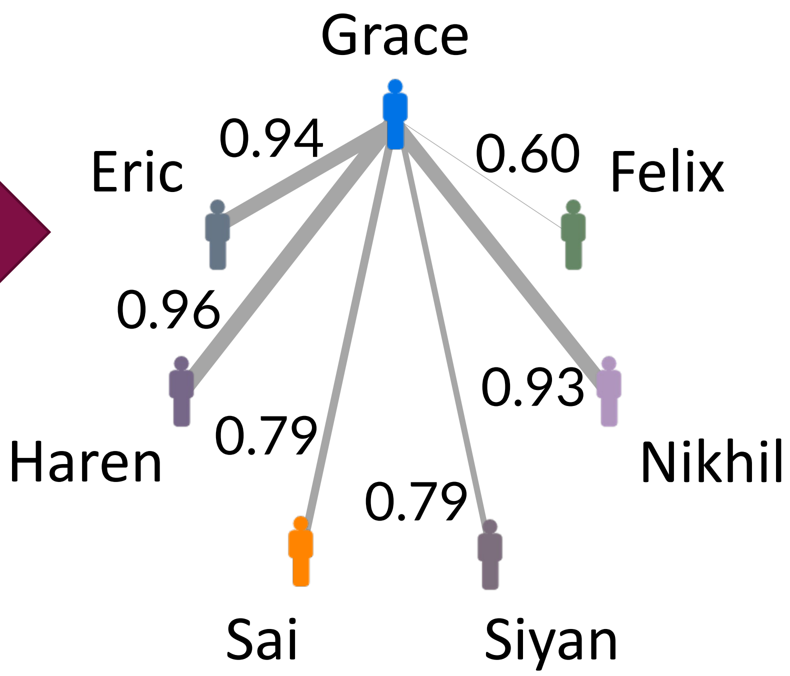
Collaborative Filtering

User-Item Utility Matrix

	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace	4	5	4	1	5	3	...
Eric	1	4	5	1	5	3	...
Haren	5	5	5	1	3	4	...
Sai	1	2	5	4	3	5	...
Siyan	3	1	1	3	4	5	...
Nikhil	2	3	4	2	2	2	...
Felix	1	1	1	5	2	2	...

distance metric

User Similarities



We could then predict unknown item utilities for Grace based on other similar users

Open issues:

- Choice of distance metric
- Dealing with sparse data
- How to combine known user utilities to do the prediction

User Distances

Distance Metrics: Measuring Similarity Between Users

There are many ways to measure user similarity:

- Euclidean similarity
- Cosine similarity
- Pearson correlation

Pros:

- Straightforward to use as a similarity metric
  - Euclidean similarity:

$$\text{similarity}(user_u, user_v) = \frac{1}{1 + \|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2} \in (0, 1]$$

  - Cosine similarity:

$$\text{similarity}(user_u, user_v) = \frac{\boldsymbol{x}_u \cdot \boldsymbol{x}_v}{\|\boldsymbol{x}_u\| \|\boldsymbol{x}_v\|} \in [0, 1]$$

Cons:

- Assumes utilities are calibrated across users
  - i.e., some users might give overall higher ratings than others

Distance Metrics: Measuring Similarity Between Users

There are many ways to measure user similarity:

- Euclidean similarity
- Cosine similarity
- Pearson correlation

Measures the linear correlation between two users' utilities; value $\in [-1,1]$

- Recall, this is formally defined as:

$$\rho = \frac{\text{covariance}(\boldsymbol{x}_u, \boldsymbol{x}_v)}{\text{stdev}(\boldsymbol{x}_u) \times \text{stdev}(\boldsymbol{x}_v)} = \frac{E[(x_{ui} - \bar{x}_u)(x_{vi} - \bar{x}_v)]}{\text{stdev}(\boldsymbol{x}_u) \times \text{stdev}(\boldsymbol{x}_v)}$$

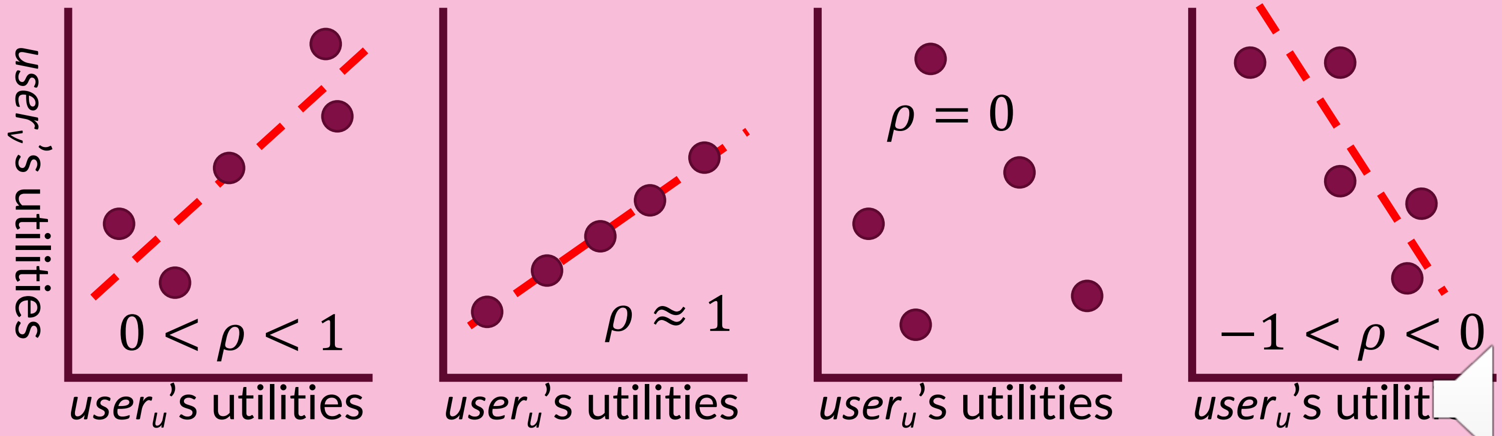
Distance Metrics: Measuring Similarity Between Users

There are many ways to measure user similarity:

- Euclidean similarity
- Cosine similarity
- Pearson correlation

Measures the linear correlation between two users' utilities; value $\in [-1,1]$

- Measuring correlations between users' utilities allows it to handle different scale calibrations
- Related to the slope (+/-) and quality of linear regression fit to the paired points

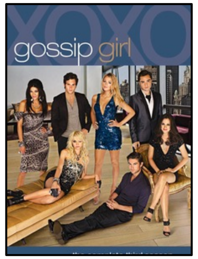


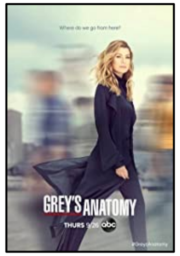
Pearson correlation coefficient ρ is:

- 1 if there is a perfect linear relationship with pos. slope
- 0 if no linear relationship exists
- -1 if perfect linear relationship with neg. slope

The Utility Matrix is Sparse

The Utility Matrix is Sparse





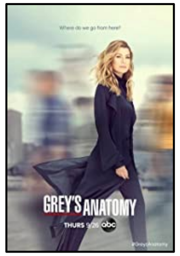
Blanks indicate the user has not rated the item

In practice, the matrix would be much sparser

	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

The goal of collaborative filtering is to predict values for blanks in the utility matrix

Measuring User Similarity with Sparse Utility Data

Can only measure similarity between users using their overlapping items

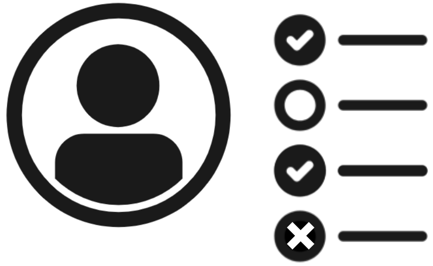
	G	Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

?

Nearest-Neighbor Collaborative Filtering

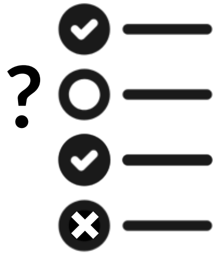
Collaborative Filtering Steps

Collect user-item utilities



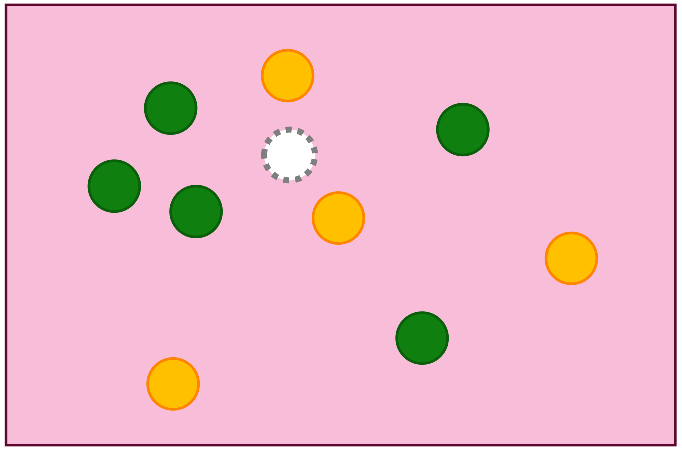
Identify similar users

Predict unknown item utilities based on other similar users



Nearest-Neighbor Collaborative Filtering

- A type of user-to-user collaborative filtering
- Very simple, yet effective



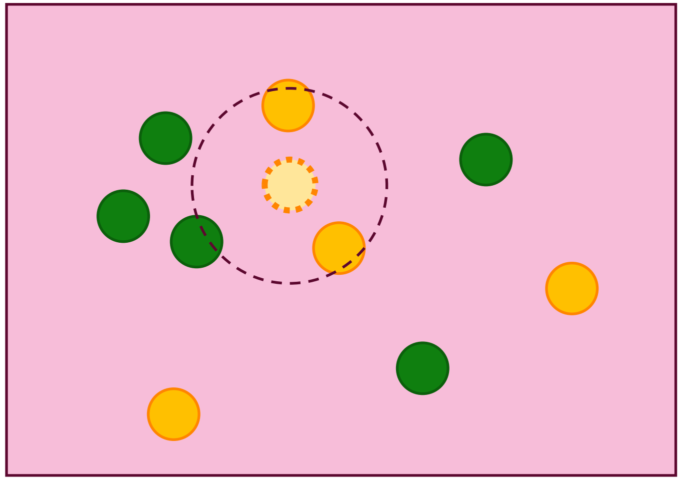
Idea: predict utility of item *i* based on the most-similar users who recorded a utility for that item

- Let $\mathcal{N}$ be the neighborhood set: the most similar users to user *u* who have rated *i*
- Let w_{uv} be a weight $\in [0,1]$ based on the similarity of users *u* and *v*
- Predict user *u*'s utility for item *i* as $\hat{x}_{ui} = \underbrace{\bar{x}_u}_{\text{Offset to this user's mean}} + \underbrace{\sigma_u}_{\text{Scale to this user's range}} \left(\sum_{v \in \mathcal{N}} \underbrace{\frac{(x_{vi} - \bar{x}_v)}{\sigma_v}}_{\text{mean-center and normalize other's utilities}} \times \underbrace{\frac{w_{uv}}{\sum_{v' \in \mathcal{N}} w_{uv'}}}_{\text{normalize weights to sum to 1}} \right)$

Nearest-Neighbor Collaborative Filtering

Ways to select the neighborhood set $\mathcal{N}$:

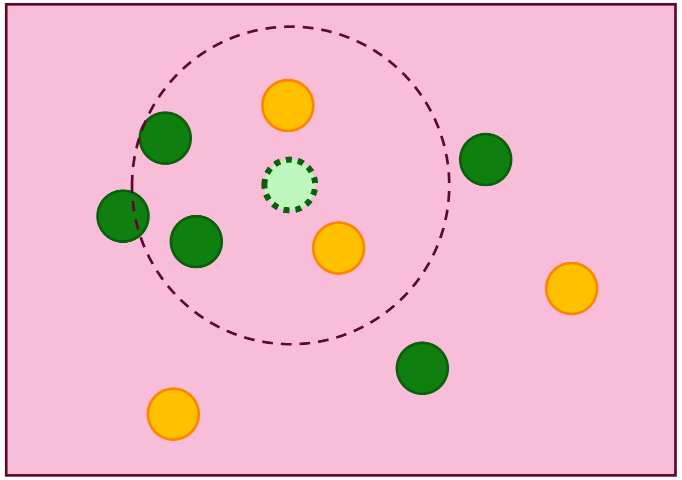
- Based on a threshold of similarity
- Choose top-*k* neighbors by similarity
- Cluster users (e.g. using *k*-means clustering), and choose the entire cluster



k = 3 → orange

Combining utilities:

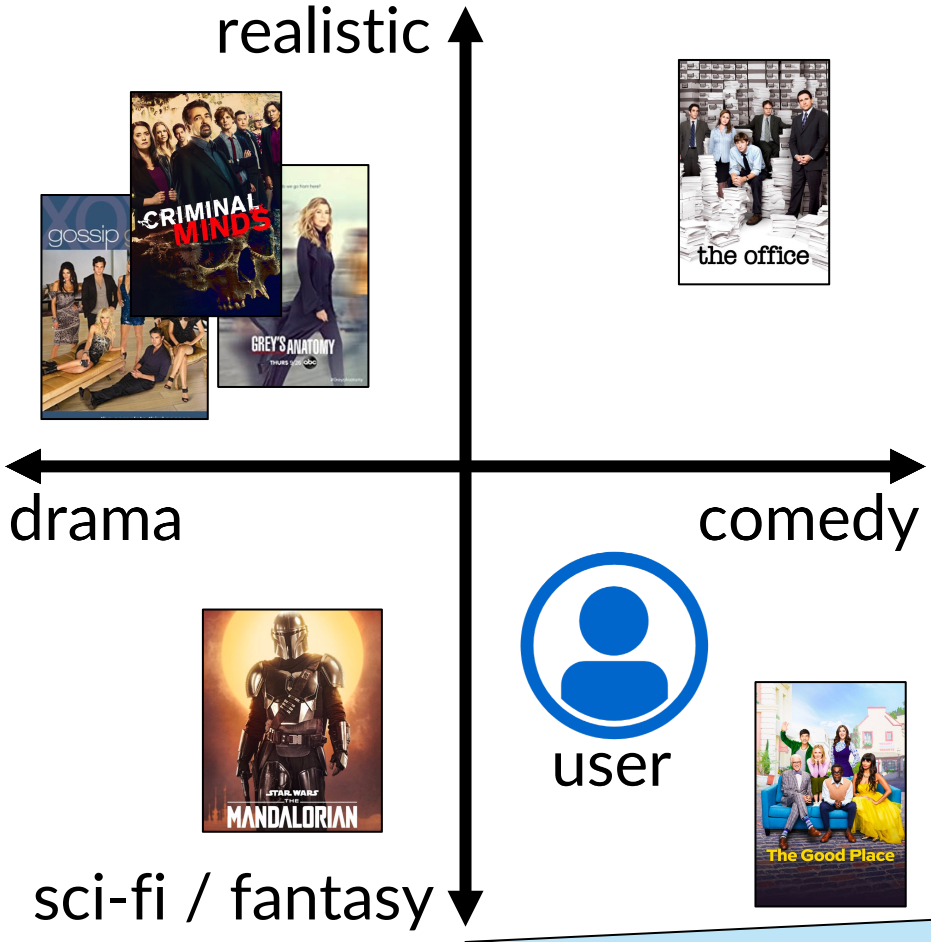
- Mean-centering
- Standardize by user's stdev



k = 5 → green

Matrix Factorization-Based Collaborative Filtering

Matrix Factorization-Based Collaborative Filtering



Idea:

- Represent each item as a vector $\boldsymbol{q}_i \in \mathbb{R}^d$
- Represent each user as a vector $\boldsymbol{p}_u \in \mathbb{R}^d$
- Approximate user *u*'s utility for item *i* as

$$\hat{x}_{ui} = q_i^T p_u$$

These vectors factorize the utility matrix

Matrix Factorization-Based Collaborative Filtering

Determining the factors:

- Just factorize the user-item utility matrix *U* directly via singular value decomposition (SVD)?
  - This will only work if we knew the full matrix, which we don't
- A better way is to fit the model with regularization

$$\min_{q^*,p^*} \sum_{x_{ui} \in U} \left(x_{ui} - q_i^T p_u\right)^2 + + \sum_i ||q_i||_2^2 + \sum_u ||p_u||_2^2$$

  - Solve via stochastic gradient descent or alternating least squares
  - For details, see:
    - Koren, et al. (2009) Matrix factorization techniques for recommender systems. *Computer* 42 (8), ACM. https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf

Injecting Prior Knowledge About Items (and Users)

Pros & Cons of The Methods So Far

Advantages:

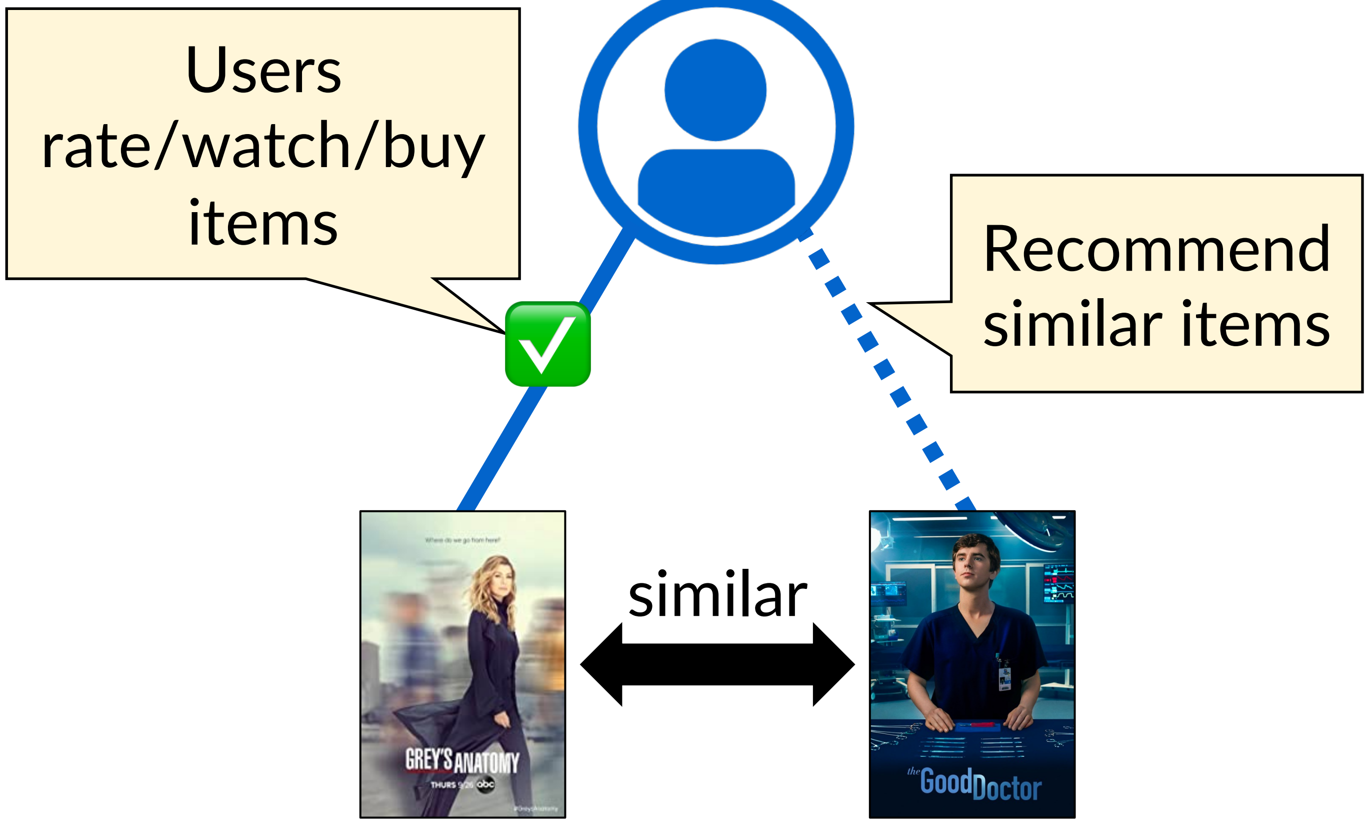
- No domain knowledge needed
  - Item details are irrelevant, only user behavior matters
- Heterogeneous preferences
  - Captures that users may have diverse preferences

Disadvantages:

- Suffers when data is sparse
  - Cannot generalize across items
    - Does not consider item content, and so cannot generalize to similar items
    - e.g. New items have no user feedback, and so the system cannot make recommendations for them
  - Cannot generalize across users

“Content-Based Methods”

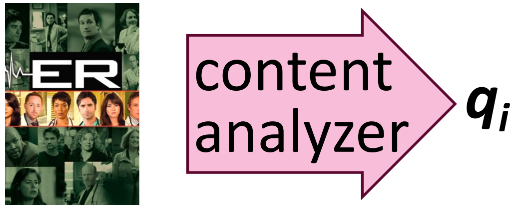
- Vanilla collaborative filtering doesn’t consider user or item attributes/content
- Content-based methods can do:



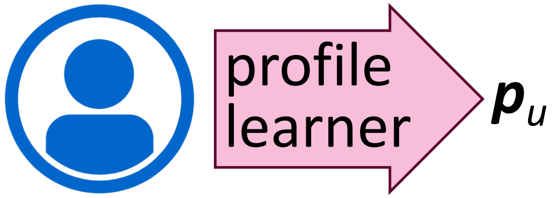
Incorporating User / Item Knowledge

Steps:

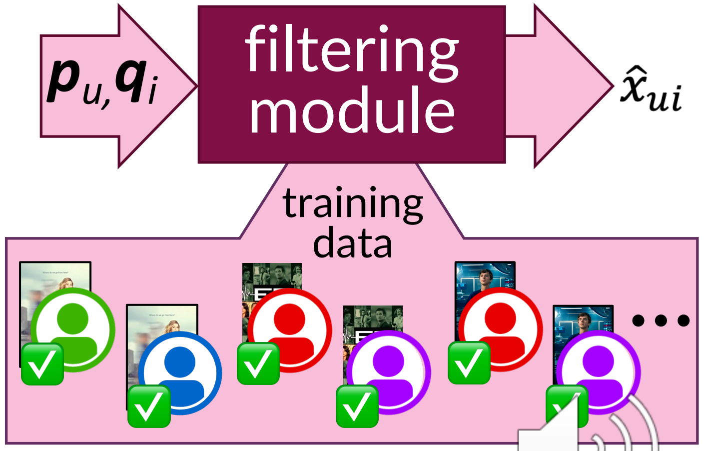
1. Content analysis: Characterize item as feature vector
   - e.g., TF-IDF features of description, image features, etc.



2. Profile learning: Characterize user as feature vector
   - e.g., demographic details, or true/predicted ratings for representative items



3. Filtering module: Learns a classification/regression model for predicting user's utility for an item
   - Train model on items each user has rated

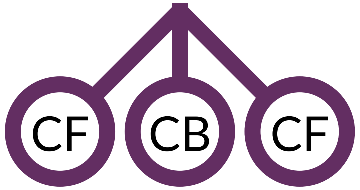


Q: What happens with a new item or new user?

Hybrid Approaches

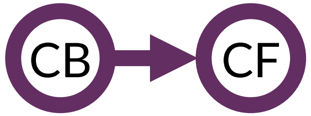
Hybrid Recommenders

Idea: Combine multiple recommenders to improve performance

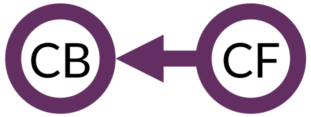


Combining separate recommenders

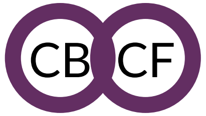
- Can use any ensemble technique: linear weighting, stacking, etc.
- Recall – the Netflix prize winner was a blend of over 800+ recommenders



Adding content-based aspects to collaborative models



- e.g., content-based user profiles to help build collaborator neighborhoods



Adding collaborative-based aspects to content-based models

Models combining content and collaboration

Hybrid Recommenders

Most systems that we use nowadays are hybrid recommenders:

NETFLIX

- Shows other similar users are watching
- Shows similar to others the user has rated/viewed

amazon.com

- Items other similar users have purchased
- Items that are similar to user's past purchases

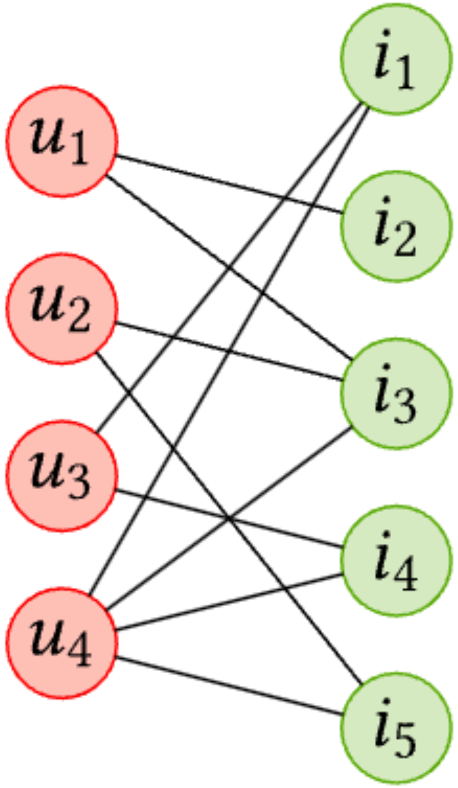
okcupid

- Profiles that other similar users have liked/viewed
- Profiles selected based on user's personal preferences

Graph Neural Networks for Recommender Systems

User-Item Bipartite Graph in Recommender Systems

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



Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey

GNNs in Collaborative Filtering

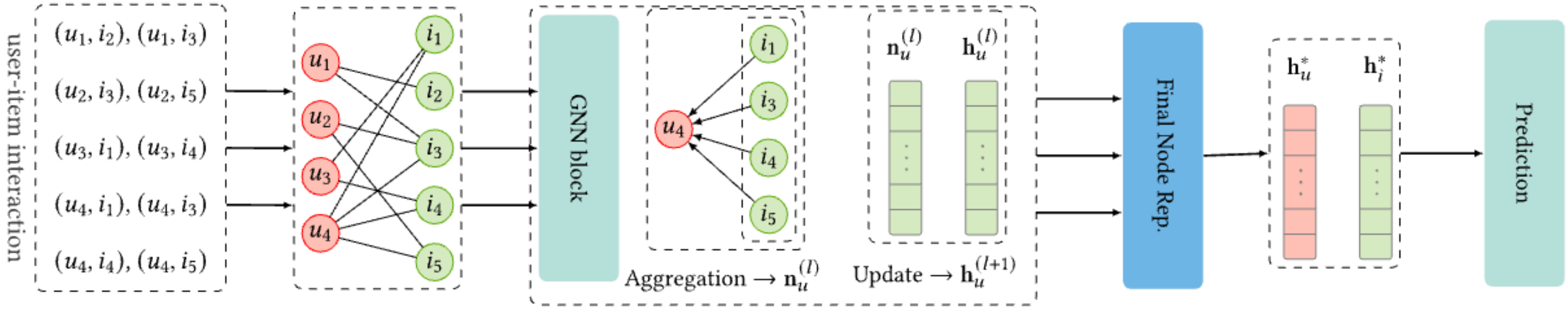


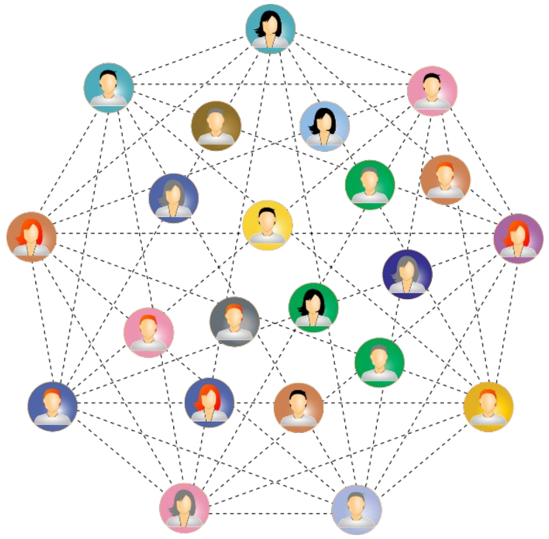
Fig. 2. The overall framework of GNN in user-item collaborative filtering.

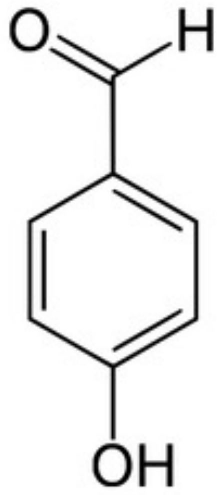
Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey

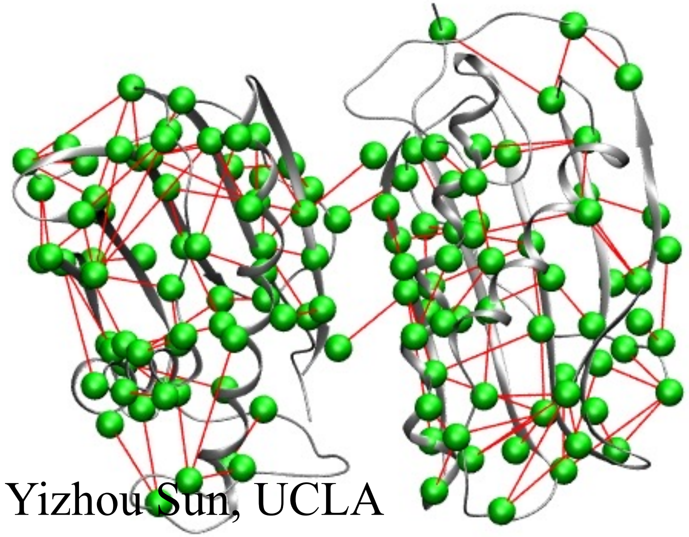
A Detour into GNNs (not necessarily for Recommendation Systems)

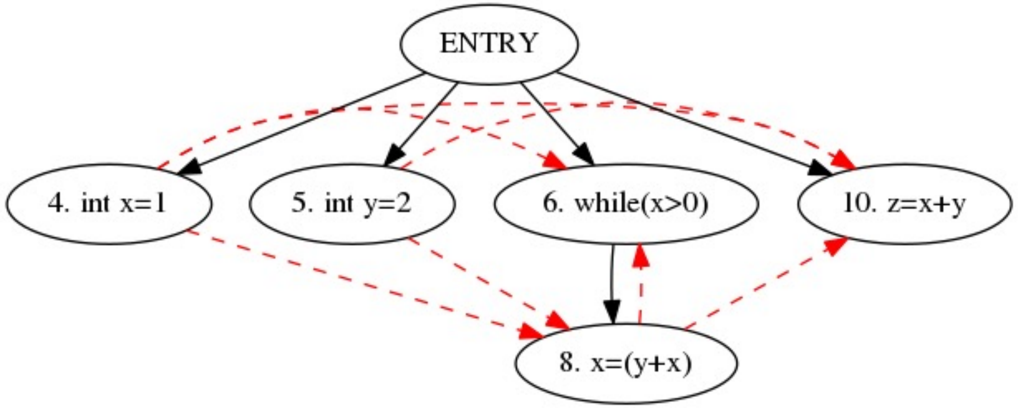
Graph Analysis

- Graphs are ubiquitous
  - Social networks
  - Proteins
  - Chemical compounds
  - Program dependence graph
  - …

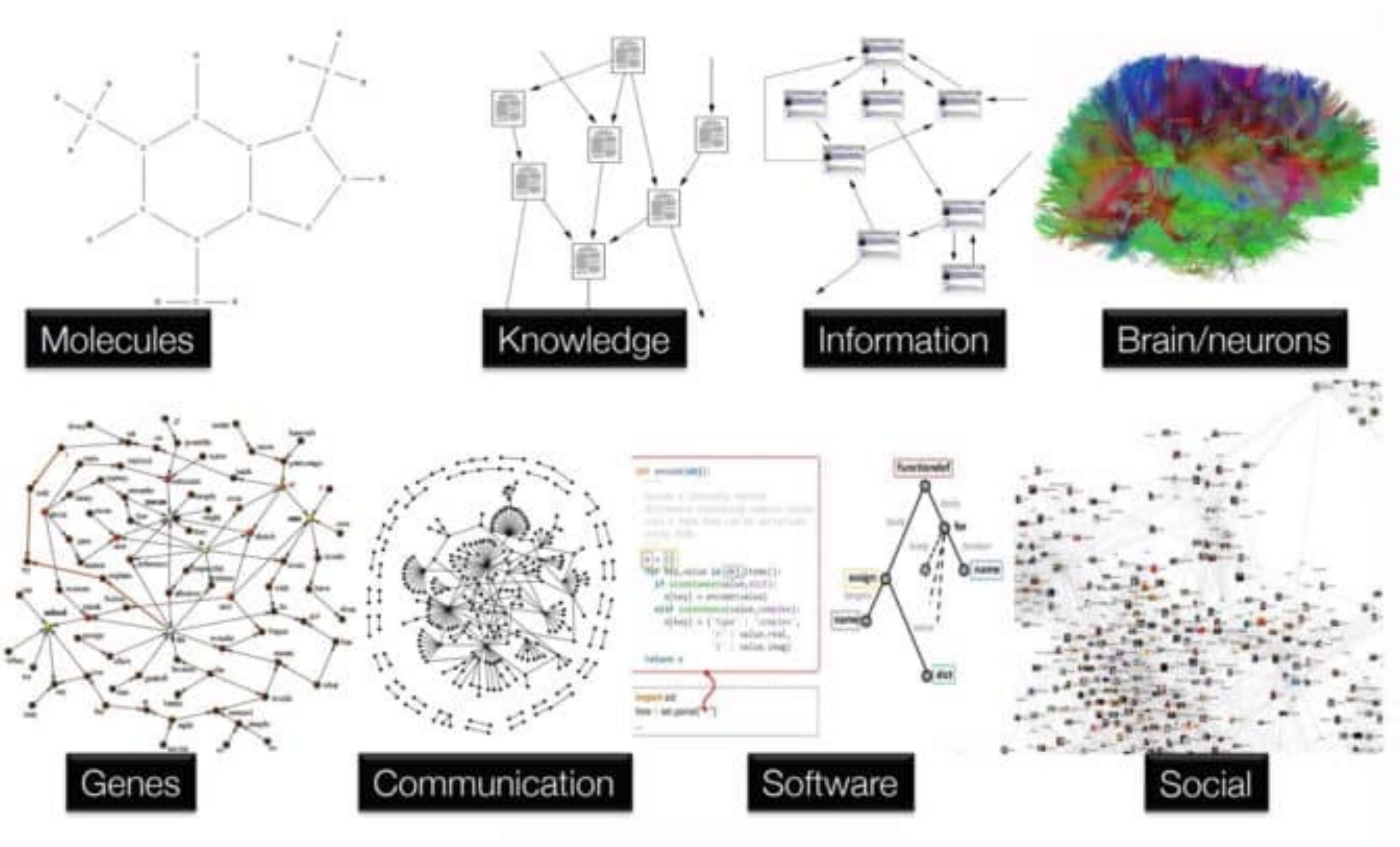








Based on slides by Yizhou Sun, UCLA



[What Are Graph Neural Networks? | NVIDIA Blogs]

Notations

- An attributed graph $G = (V, E)$
  - V: vertex set
  - E: edge set
  - A: adjacency matrix
  - $X \in R^{d_0 \times |V|}$: feature matrix for all the nodes
  - $N(v)$: neighbors of node v
  - h_v^l: Representation vector of node v at Layer l
    - $Note\, h_v^0 = x_v$
  - $H^l \in R^{d_l \times |V|}$: representation matrix

One layer of a GNN

- For a node v at layer t

$$h_v^{(l)} = f\left(\underline{h_v^{(l-1)}}, \underline{\left\{h_u^{(l-1)} | u \in \mathcal{N}(v)\right\}}\right)$$

representation vector from previous layer for node v

representation vectors from previous layer for node v's neighbors

  - Two key things that must happen in f(.):
    - **Aggregation: combine all neighbor features into one new feature for the neighborhood $\boldsymbol{n_v^{(t)}}$**
    - **Update: Combine current central node feature at previous layer + aggregated feature of the neighborhood above**

Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey

Based on slides by Yizhou Sun, UCLA

Common Choices for Aggregation

$$\text{Aggregation: } n_v^{(l-1)} = Aggregator_{l-1}\left(\left\{h_u^{(l-1)}, \forall u \in \mathcal{N}_v\right\}\right)$$

- Mean: $n_v^{(l-1)} = \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(l-1)}}{|\mathcal{N}(v)|}$
- Modified Mean ("graph convolutions"): $n_v^{(l-1)} = \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(l-1)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}}$
- Attention mechanisms: : $n_v^{(l-1)} = \sum_{u \in \mathcal{N}(v)} \alpha_{vu} h_u^{(l-1)}$

α_{vu} are weights computed by attention

Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey
Based on slides by Yizhou Sun, UCLA

Common Choices for Update

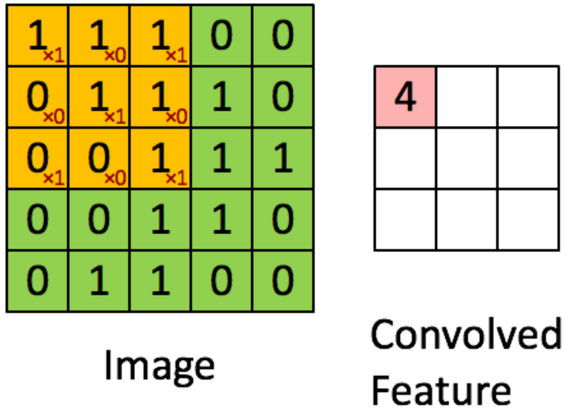
$$\text{Update } h_v^{(l)} = Updater_l\left(h_v^{(l-1)}, n_v^{(l-1)}\right)$$

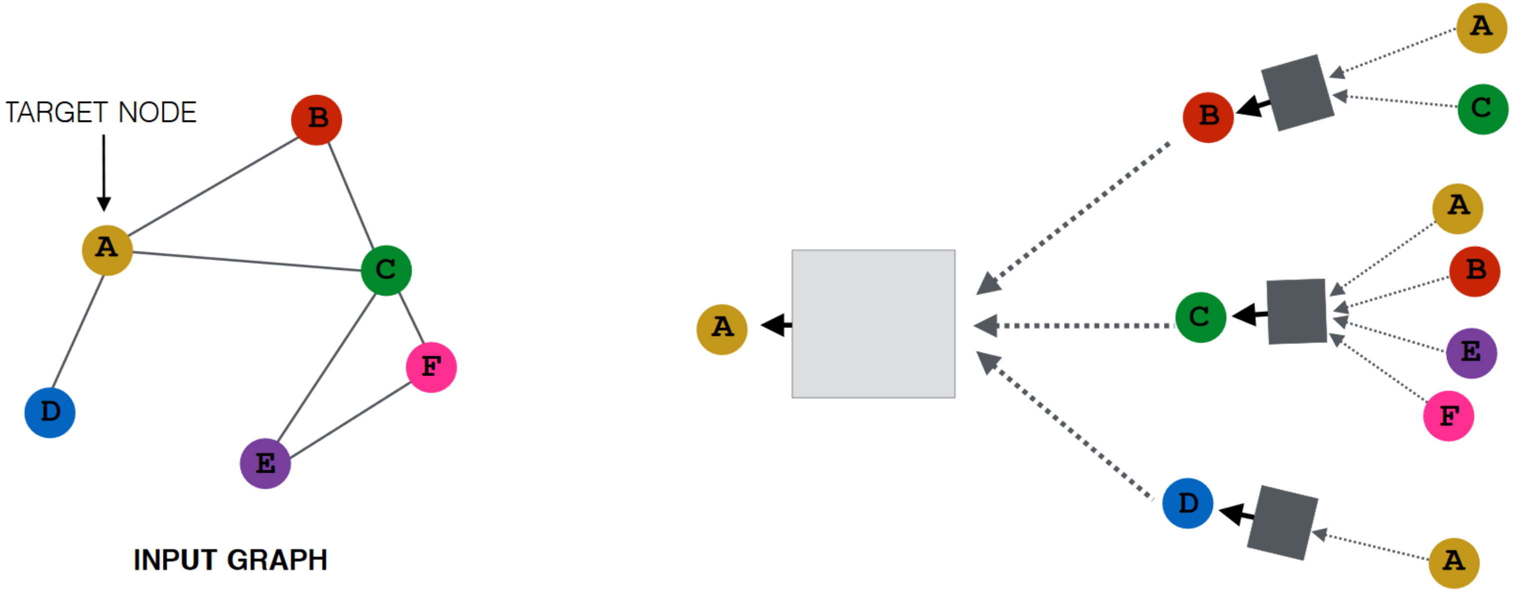
Often a simple linear layer + non-linear activation function

$$\text{e.g. } h_v^{(l)} = \delta\left(W^{(l-1)}.\,concat\left(h_v^{(l-1)}, n_v^{(l-1)}\right) + b^{(l-1)}\right)$$

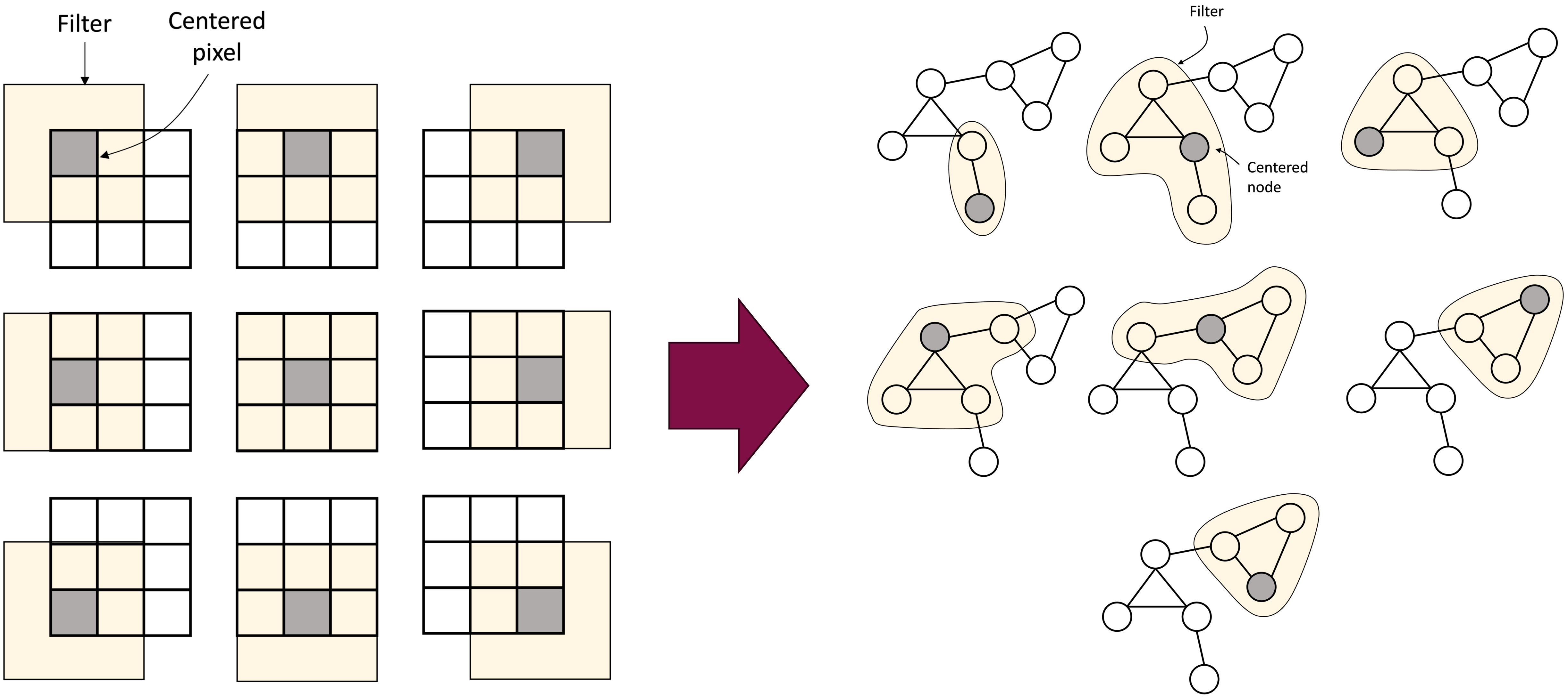
Compare with CNN

- Recall CNN
  - Regular graph
- GNN
  - Extend to irregular graph structure

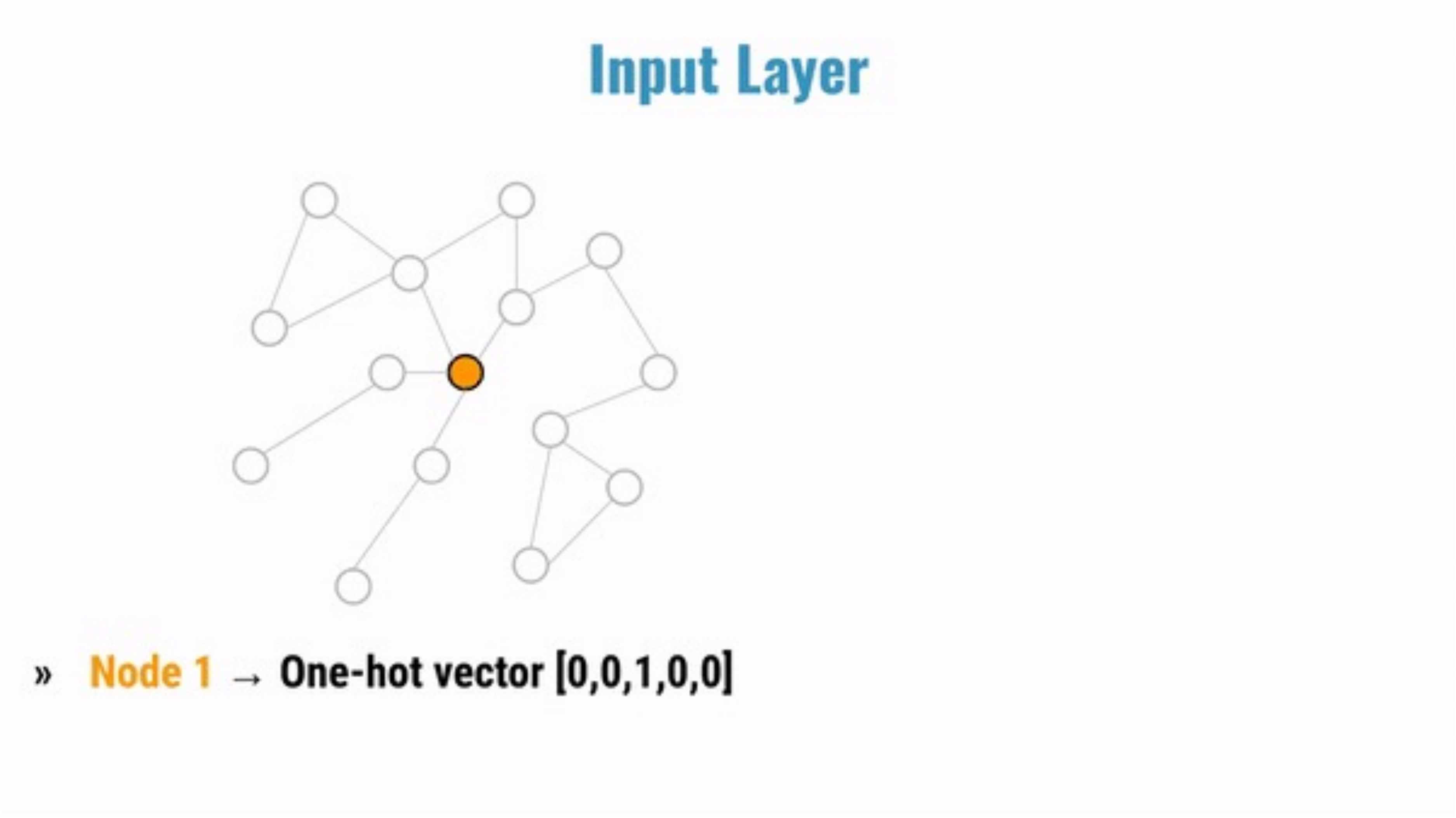




“Graph Convolutions”



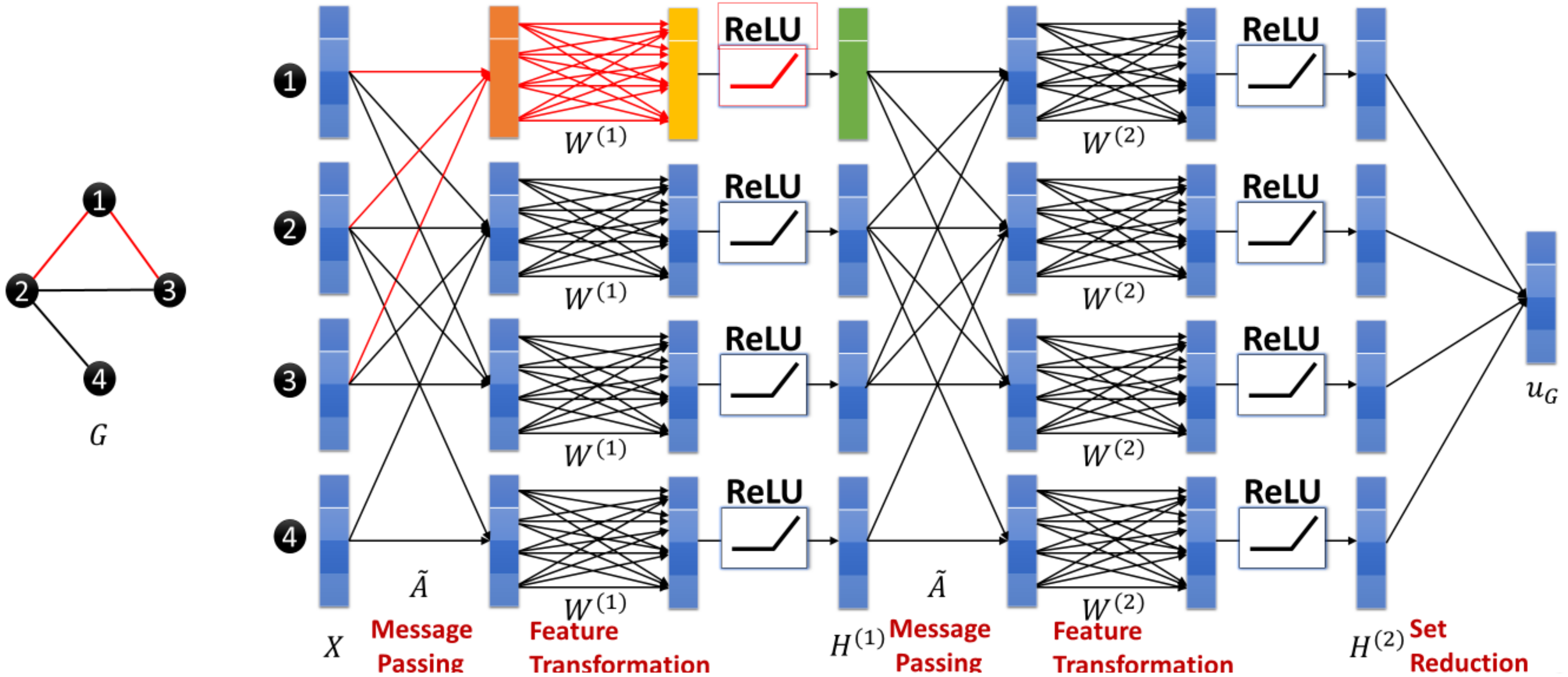
Graph convolutional neural networks - Matthew N. Bernstein



[Understanding the Building Blocks of Graph Neural Networks (Intro) | by Giuseppe Futia | Towards Data Science]

A toy example of 2-layer GCN on a 4-node graph

- Computation graph

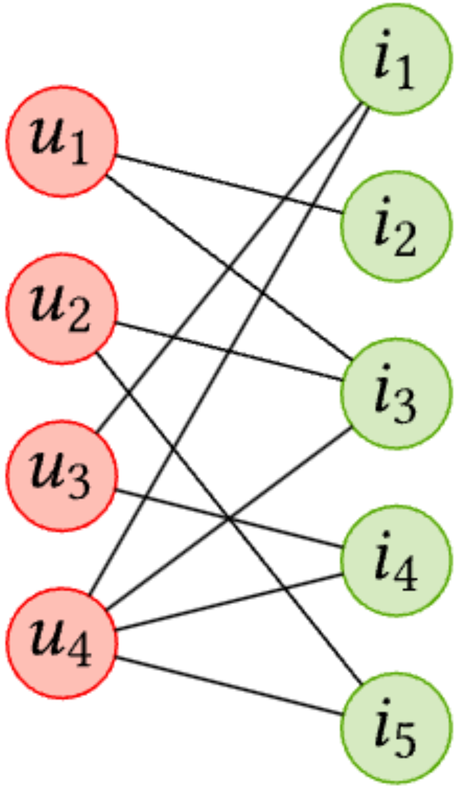


Based on slides by Yizhou Sun, UCLA

GNNs in Recommender Systems?

User-Item Bipartite Graph in Recommender Systems

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



Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey

GNNs in Collaborative Filtering

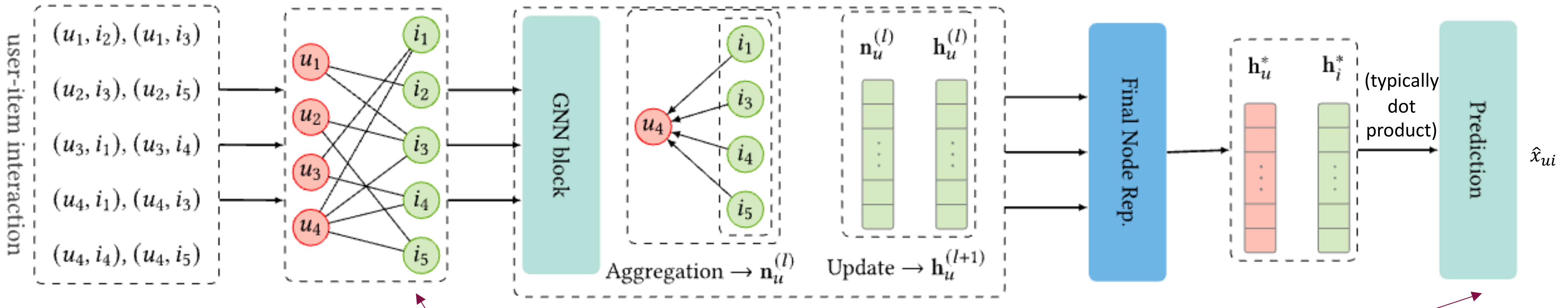
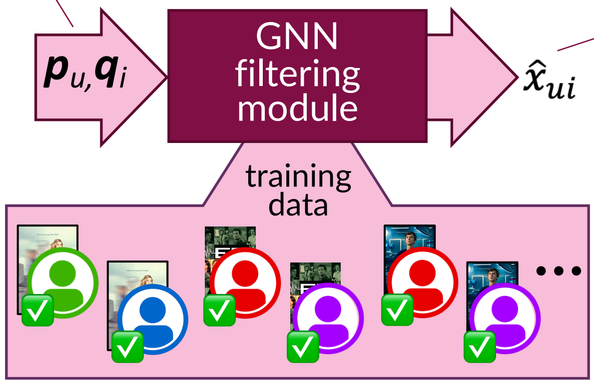


Fig. 2. The overall framework of GNN in user-item collaborative filtering.



Wu et al 2022, Graph Neural Networks in Recommender Systems: A Survey