Recommenders I

Frank Burkholder

Data Science Immersive

Objectives

- Define a recommender and describe the general approaches to providing recommendations.
- Define the difference between explicit and implicit ratings.
- Be able to describe, conceptually, how neighborhood-based collaborative filtering works.
- List common neighborhood similarity metrics.
- Explain how a neighborhood-based rating is calculated.
- List problems associated with neighborhood based collaborative filtering.
- Be able to code, and put into a class, your own collaborative, neighborhood based recommender (morning assignment).

Recommenders

- A recommender is an information filtering system that seeks to predict a user’s rating/preference for something.
- Generally, recommenders make recommendations one of three ways:
    - **Popularity** - recommend what’s most popular (trending on Twitter)
    - **Content-based** - use similarities based on attributes of items (text descriptions, features) to group/cluster them and provide recommendations for things that are similar based on user preference.
        - Example: SkiRunRecommender
    - **Collaborative filtering (CF)** - for a given user, use ratings/preferences of similar users to predict which unrated items the user would like. (more later)
        - Examples: Board Game Geek Recommender InstaBookRec
- Hybrid approaches are possible.

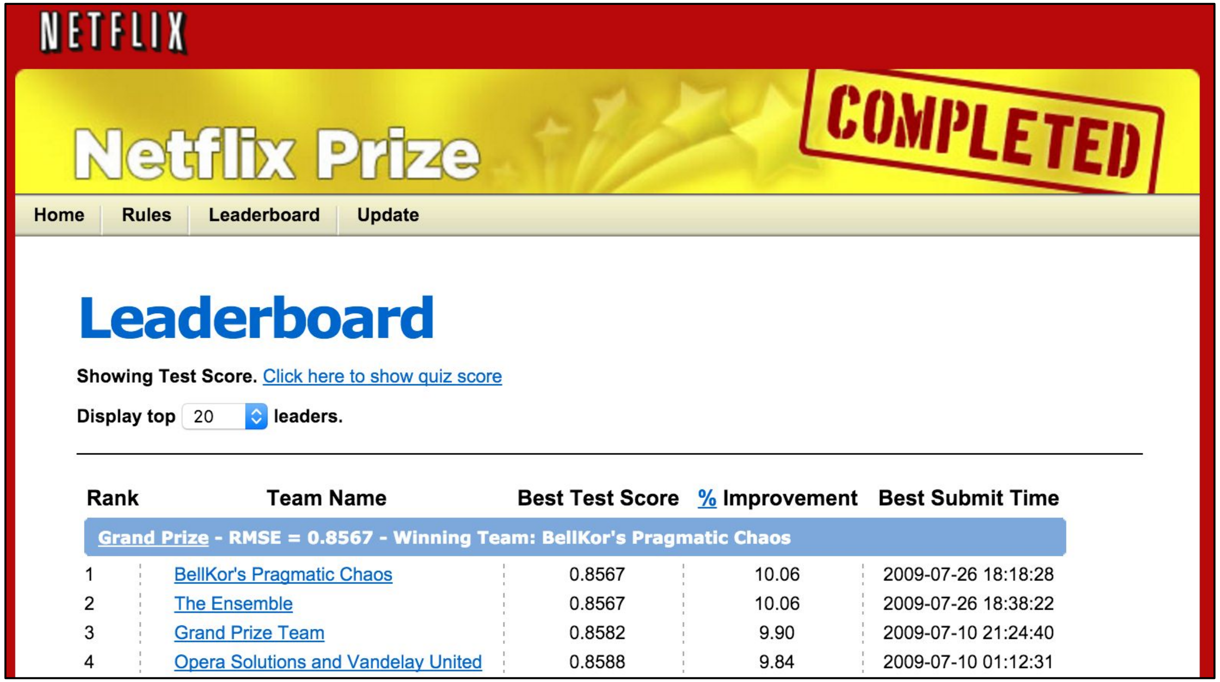
Aside - Netflix's $1,000,0000 prize

Netflix would award a team $1M when their movie recommender outperformed Netflix's recommender by 10% as measured by the RMSE on held-out ratings (1-5).

Oct. 2006 - July 2009

This popularized (and advanced) recommender systems.

link



Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31

Training Dataset: 100,480,507 ratings that 480,189 users gave to 17,770 movies
Test Dataset: 2,817,131 rating predictions (predict a rating 1-5 for each)
Evaluation Metric: RMSE

Ratings - Explicit and Implicit

Explicit rating
Allow user to unequivocally quantify how much he/she liked something.
Examples: Did you like this song? How many stars would you give this movie?

Implicit rating
Makes inference from user behavior.
How many times have you played that track? Which songs are you playing? What movies are you watching?

Explicit or implicit:
How long you look at your Facebook feed.
Thumbs up/down on a Facebook post.

Collaborative filtering

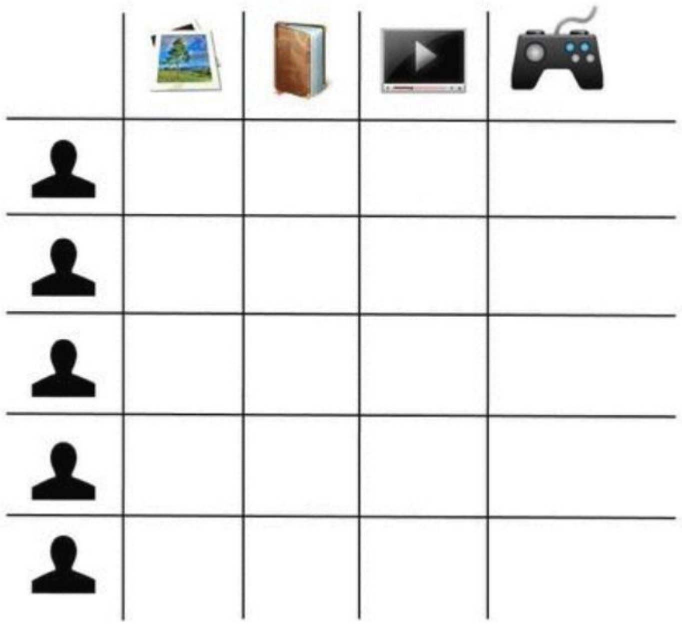
Conceptually: use ratings/preferences of similar users or items to predict which unrated items the user would like. Utilizes user behavior, not item content.

Types of collaborative filtering:

- **Memory**, neighborhood based: Look for k-NN most similar items/users to provide the rating/preference (this morning).
- **Model,** matrix factorization based: Usually use dimensionality reduction and latent factors to make a model to predict ratings (this afternoon).
- **Hybrid**: mixes both and more. Recommenders are an active area of research.

Collaborative filtering, neighborhood-based

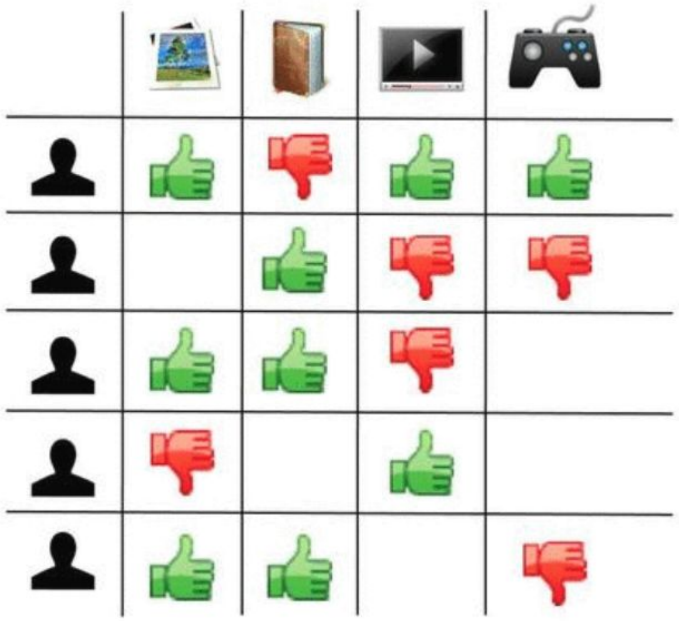
A conceptual walk-through:



source: Wikipedia

Collaborative filtering, neighborhood-based

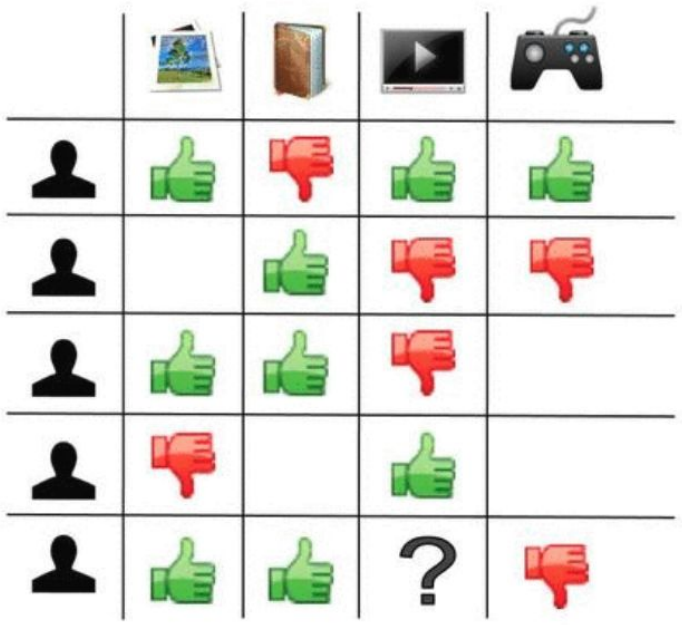
A conceptual walk-through:



source: Wikipedia

Collaborative filtering, neighborhood-based

A conceptual walk-through:

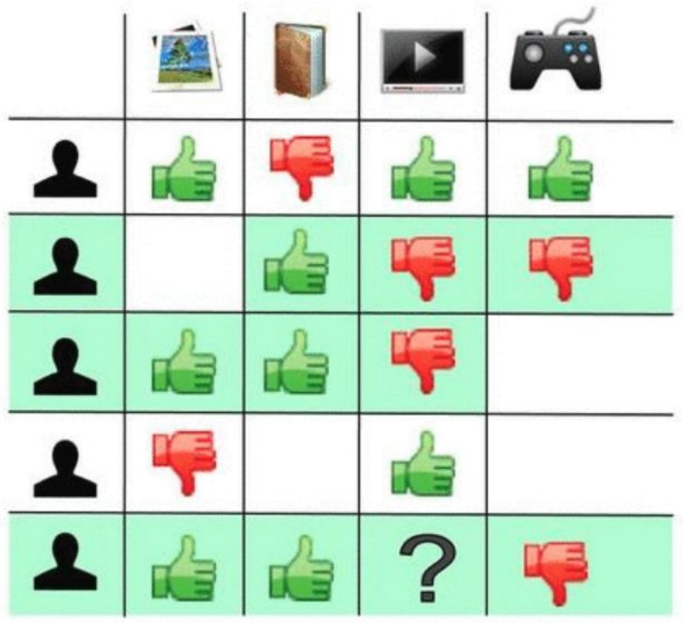


👍	👎	👍	👍
	👍	👎	👎
👍	👍	👎	
👎		👍	
👍	👍	?	👎

source: Wikipedia

Collaborative filtering, neighborhood-based

A conceptual walk-through:

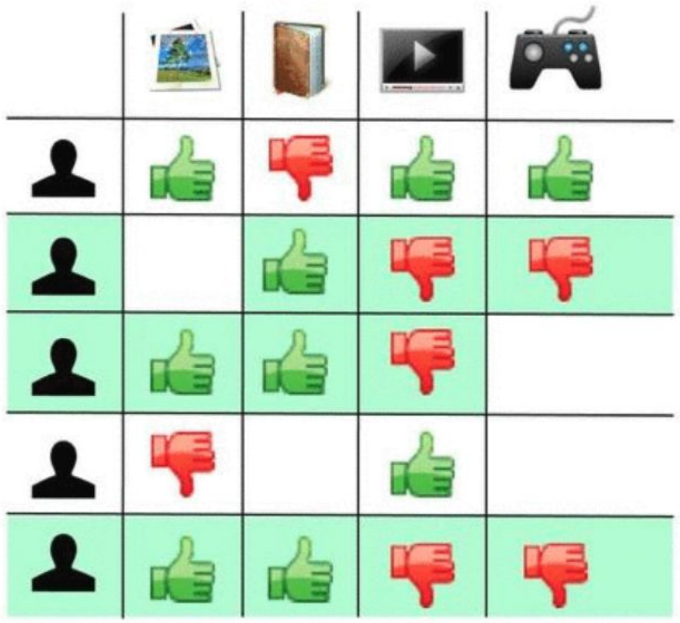


	🖼️	📖	▶️	🎮
👤	👍	👎	👍	👍
👤		👍	👎	👎
👤	👍	👍	👎	
👤	👎		👍	
👤	👍	👍	?	👎

source: Wikipedia

Collaborative filtering, neighborhood-based

A conceptual walk-through:



[source: Wikipedia]

Collaborative filtering, neighborhood-based

The process:

1) Gather ratings (explicit/implicit).

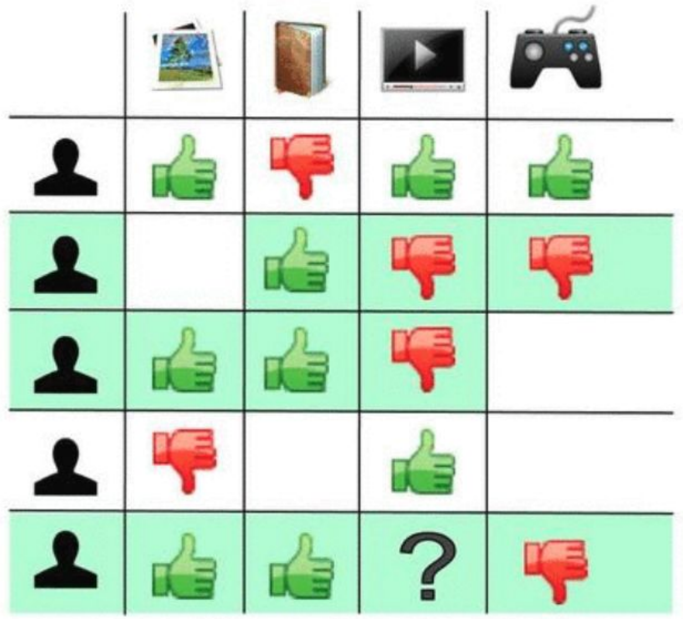
2) Put ratings in a matrix (usually rows are users, columns are items).

3) Determine where you need a rating (usually items for a given user).

4) Realize that your rows and columns are both vectors.

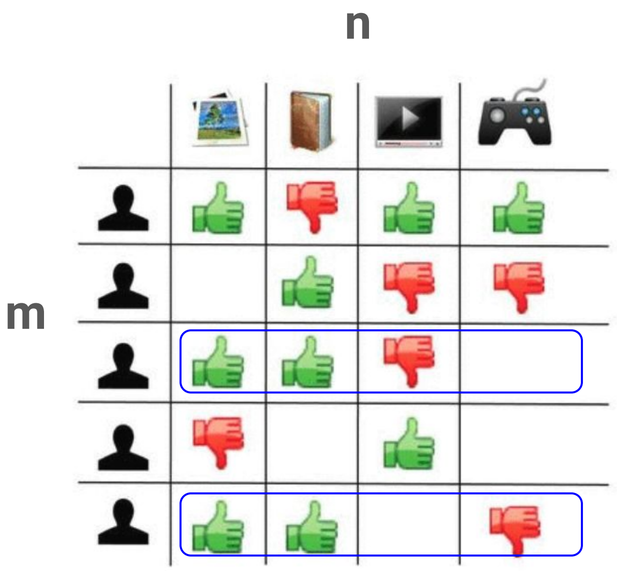
5) Can find similar vectors (user-user or item-item).

6) Use the similarity of k users/rows to make a new rating.

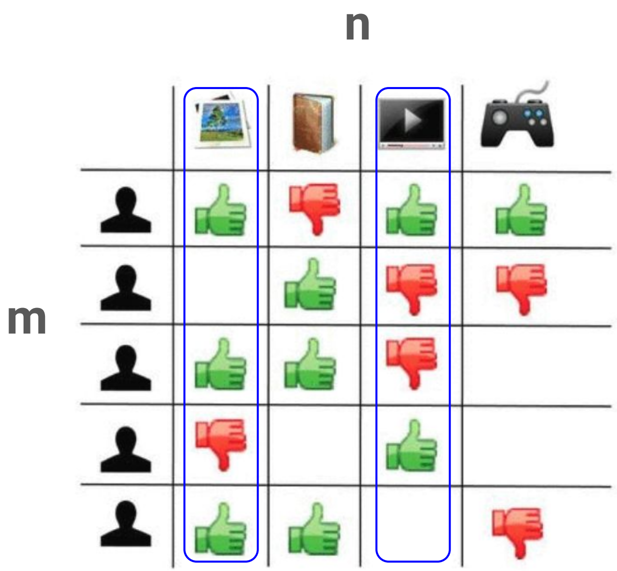


source: Wikipedia

What's better: User-User or Item-Item?



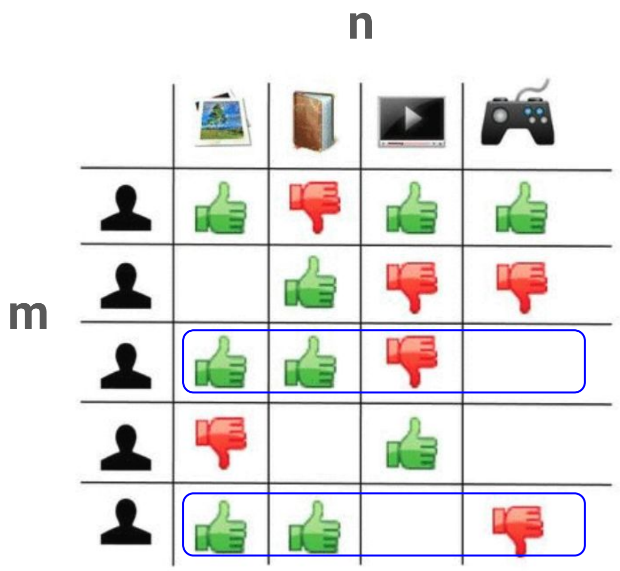
User-User



Item-Item

Order (number) of computations for each?
Express using **m** and **n**.

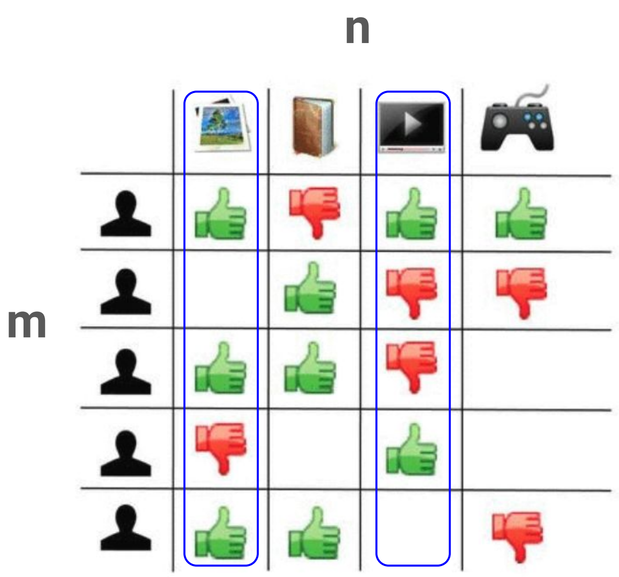
What’s better: User-User or Item-Item?



User-User

$O(\mathbf{m} * (\mathbf{m} * \mathbf{n}))$

$O(\mathbf{m}^2\mathbf{n})$



Item-Item

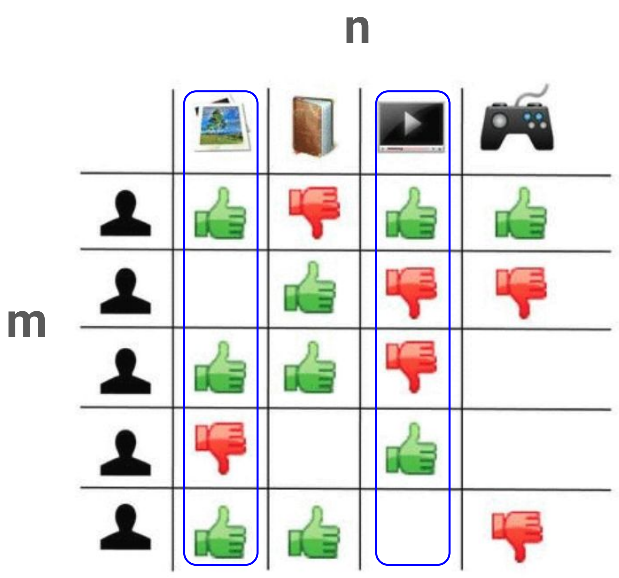
$O(\mathbf{n} * (\mathbf{n} * \mathbf{m}))$

$O(\mathbf{m}\mathbf{n}^2)$

What's better: User-User or Item-Item?

Usually Item-Item is preferred because:

- m is usually >> n
  Consider the Netflix prize:
  m ~ 500,000, n ~ 20,000
- Item ratings more heavily populated than user ratings - more robust results.



Item-Item

$O(\mathbf{n} * (\mathbf{n} * \mathbf{m}))$
$O(\mathbf{mn^2})$

Calculating Similarity

Euclidean distance
The “straight line” distance between two vectors

$$\mathrm{dist}(a,b) = ||a - b|| = \sqrt{\sum_i (a_i - b_i)^2}$$

Euclidean similarity
Ranges from 0 (not similar) to 1 (very similar)

$$\mathrm{similarity}(a,b) = \frac{1}{1 + \mathrm{dist}(a,b)}$$

Notes: Often not as useful as cosine similarity in high dimensional spaces.

Calculating Similarity

Cosine similarity

How much do the vectors point the same way? (-1 opposite, 0 orthogonal, +1 same)

$$\cos(\theta_{a,b}) = \frac{a \cdot b}{||a||||b||} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Standardized cosine similarity

Ranges from 0 (not similar) to 1 (very similar)

$$\text{similarity}(a, b) = 0.5 + 0.5 * \cos(\theta_{a,b})$$

Notes: Very useful in recommenders.

E.g. User1 = [6, 6, 3], User2 = [2, 2, 1], similarity = 1.0

Calculating Similarity

Pearson’s correlation coefficient (R)
Quantifies the strength of a linear relationship between two vectors.
-1 inversely related, 0 no relationship, 1 positively correlated.

$$\mathrm{pearson}(a,b) = \frac{\mathrm{cov}(a,b)}{\mathrm{std}(a) * \mathrm{std}(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\sqrt{\sum_i (b_i - \bar{b})^2}}$$

Pearson similarity
Ranges from 0 (not similar) to 1 (very similar)

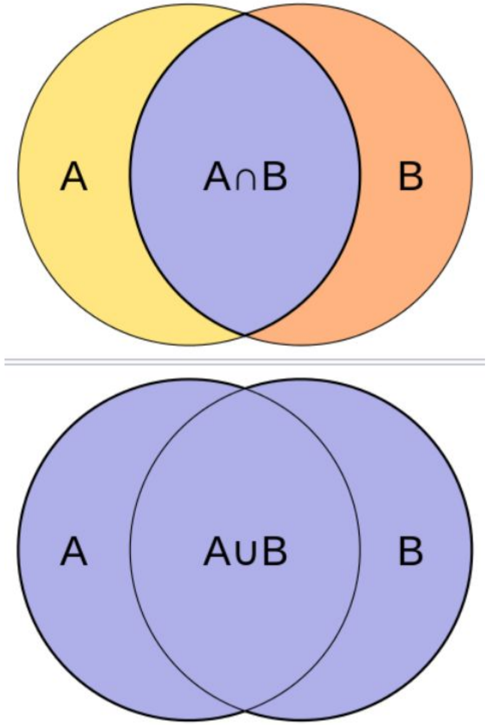
$$\mathrm{similarity}(a,b) = 0.5 + 0.5 * \mathrm{pearson}(a,b)$$

Notes: Useful in recommenders.
E.g. User1 = [6, 6, 3], User2 = [2, 2, 1], similarity = 1.0

Calculating Similarity

Jaccard Index (or Similarity Coefficient)
It's a statistic for comparing the similarity of sample sets.
Ranges from 0 (no overlap or similarity) to 1 (complete overlap or similarity)

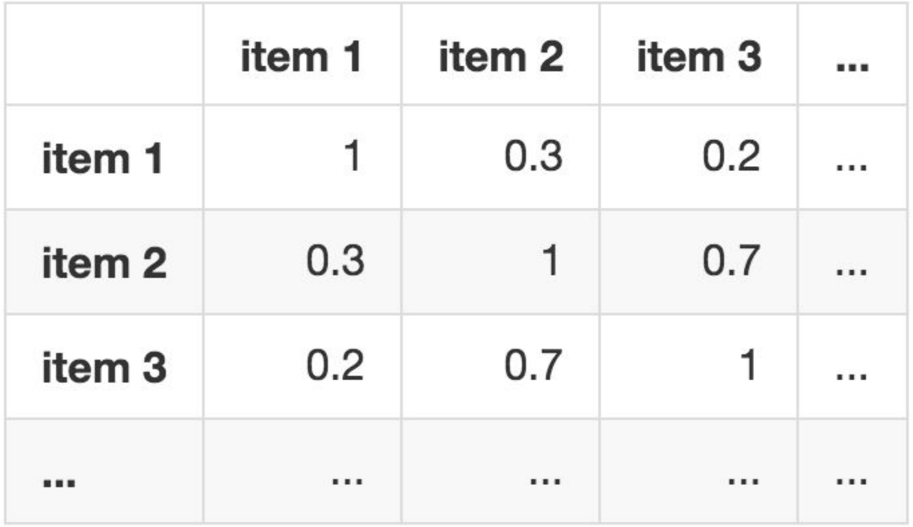
$$\text{similarity}(a, b) = \frac{|U_a \cap U_b|}{|U_a \cup U_b|}$$



Notes: Good for implicit rankings.

Similarity Matrix

Use the similarity metric to calculate the similarity of all the items to each other, resulting in a similarity matrix.



	item 1	item 2	item 3	...
item 1	1	0.3	0.2	...
item 2	0.3	1	0.7	...
item 3	0.2	0.7	1	...
...	...	...	...	...

Using similarities and ratings to predict ratings

Say user *u* hasn't rated item *i*. We want to predict the rating that this user *would* give this item.

$$\text{rating}(u,i) = \frac{\sum_{j \in I_u} \text{similarity}(i,j) * r_{u,j}}{\sum_{j \in I_u} \text{similarity}(i,j)}$$

$$I_u = \text{ set of items rated by user } u$$

$$r_{u,j} = \text{ user } u\text{'s rating of item } j$$

We order by descending predicted rating for a single user, and recommend the top number of items to the user.

Simple neighborhood-based collab. filtering demo

`collab_filter_itemitem.ipynb`

Using similarities and ratings to predict ratings

This calculation of predicted ratings can be very costly. To mitigate this issue, we will only consider the *n* most similar items to an item when calculating the prediction.

$$\text{rating}(u, i) = \frac{\sum_{j \in I_u \cap N_i} \text{similarity}(i, j) * r_{u,j}}{\sum_{j \in I_u \cap N_i} \text{similarity}(i, j)}$$

$$I_u = \text{ set of items rated by user } u$$

$$r_{u,j} = \text{ user } u\text{'s rating of item } j$$

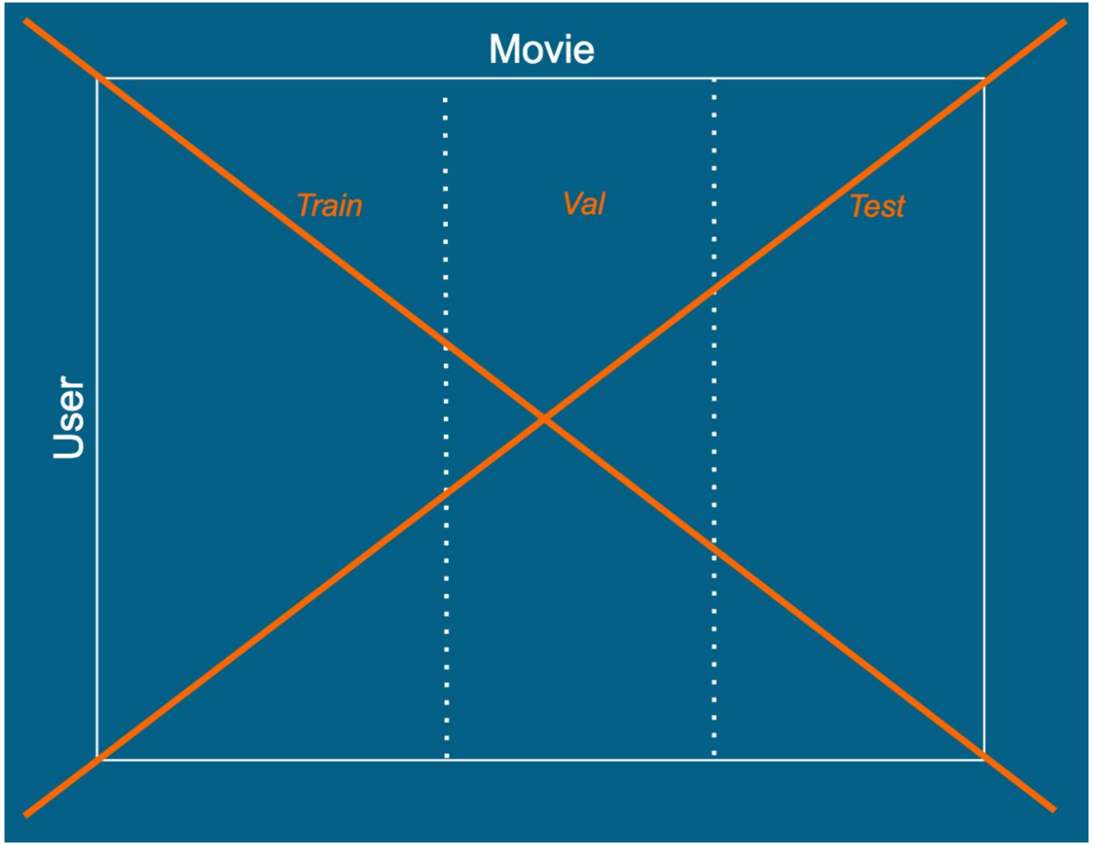
$$N_i \text{ is the } n \text{ items which are most similar to item } i$$

Collaborative filtering difficulties

- **Cold start**: For a new user that has no ratings, what do you recommend to him/her? For a new item with no ratings, how do you recommend it?
- **Scalability**: For neighborhood based item-item recommenders, $\mathbf{mn^2}$ similarity metrics must be calculated (hopefully the night before) before ratings can be calculated. Requires a lot of computation power.
- **Sparsity**: There are often many items but the ratings are sparse. High dimensionality without data. Overfitting!
    - Enter CF dimensionality reduction technique, matrix factorization (this afternoon)
- **How do you verify your recommender is “right?”**

Cross Validation, done poorly

With no data on the Val and Test items, no predictions can be made.



Cross Validation, better...but

? means held-out, known ratings.

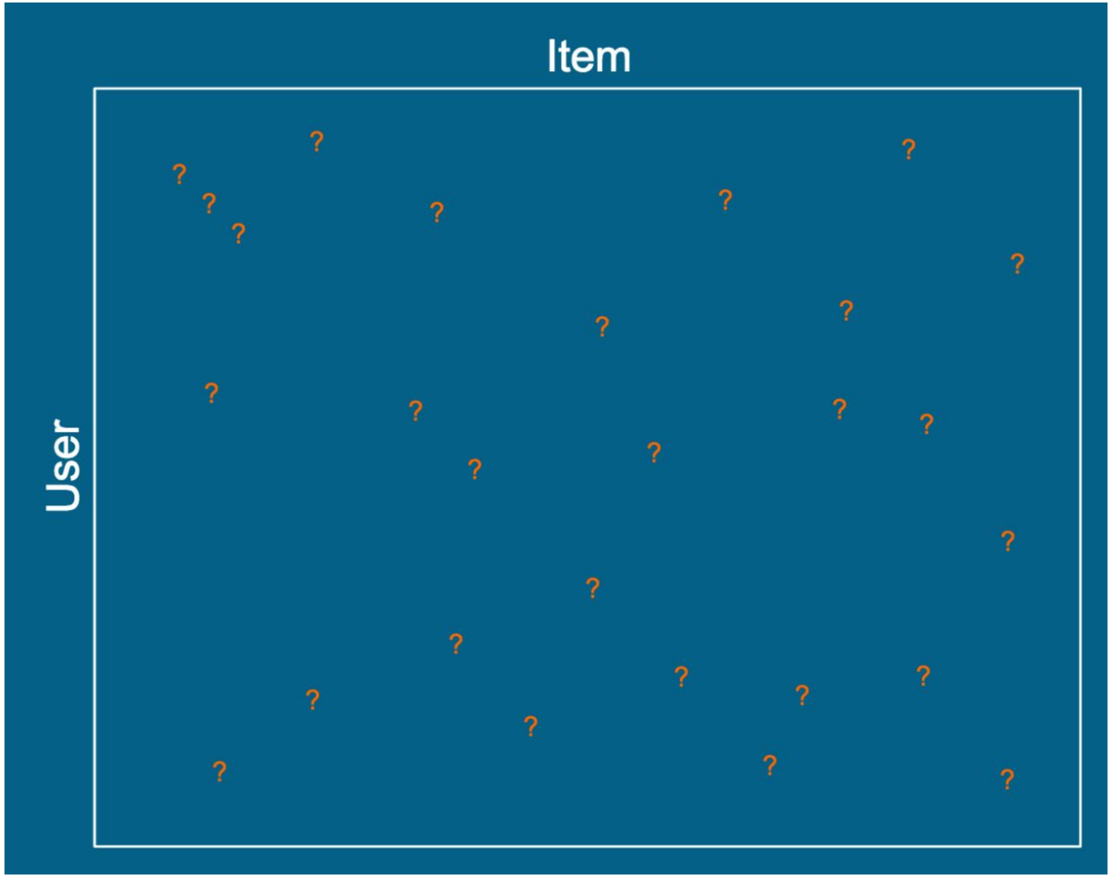
We can calculate MSE between the known ratings and the predictions over the holdout set.

But do we really care about predicting ratings? The purpose of recommender is to present things that the user wants to see next.

Only care about presenting items in right order, not their absolute rating.

Better options:

- segment train/val/holdout in time
- A/B testing



Objectives

- Define a recommender and describe the general approaches to providing recommendations.
- Define the difference between explicit and implicit ratings.
- Be able to describe, conceptually, how neighborhood-based collaborative filtering works.
- List common neighborhood similarity metrics.
- Explain how a neighborhood-based rating is calculated.
- List problems associated with neighborhood based collaborative filtering.
- Be able to code, and put into a class, your own collaborative, neighborhood based recommender (morning assignment).