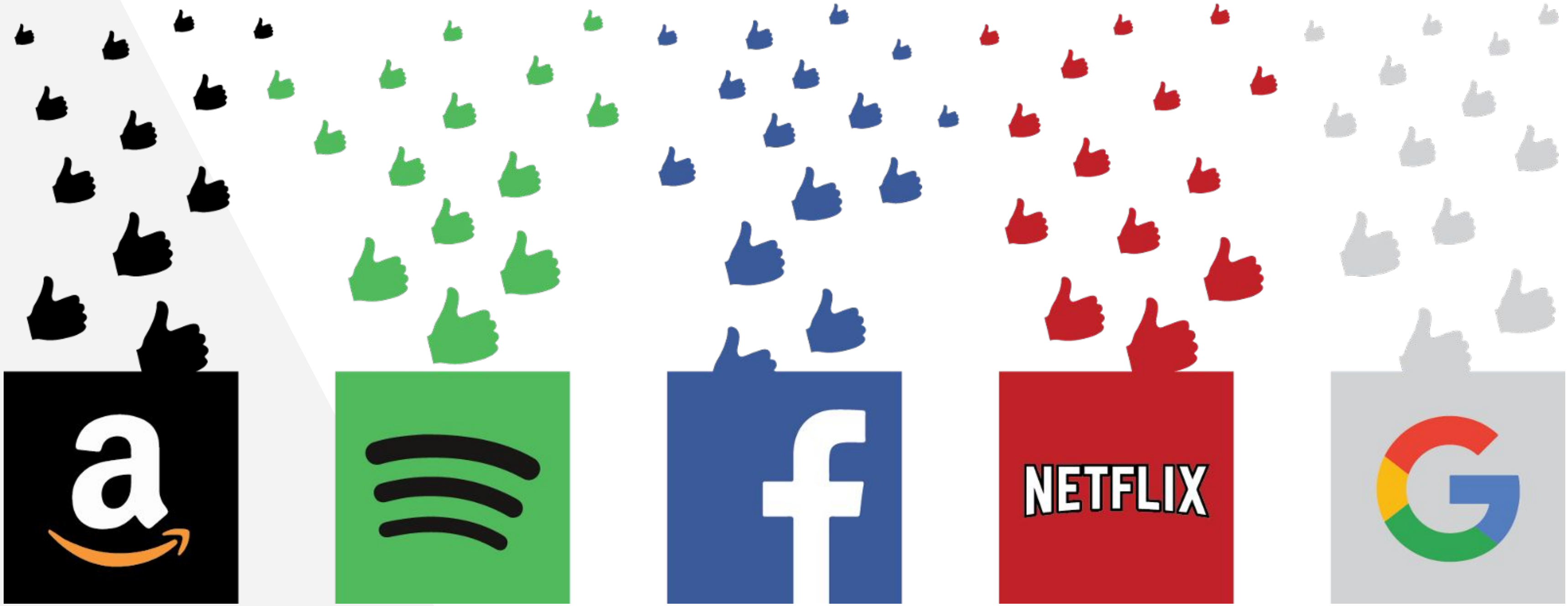
RECOMMENDER SYSTEMS

Why should we care about Recommender Systems?


NETFLIX

“

35% of Amazon.com’s revenue is generated by its recommendation engine

Source: http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

”

3 Types of Recommendations

Hand Curated

- ‘My Favorites’
- ‘Essential XYZ all ABCs must own’

Aggregations

- ‘Top Ten XYZ’
- ‘Most Popular’
- ‘Trending’
- ‘Recently Uploaded’

Personalized

- Google search results
- Amazon
- Spotify
- Netflix
- The future

3 Types of Recommender Systems

Content Based Recommendations

Main idea: Recommend items to customer x similar to previous items rated highly by x

Collaborative Filtering

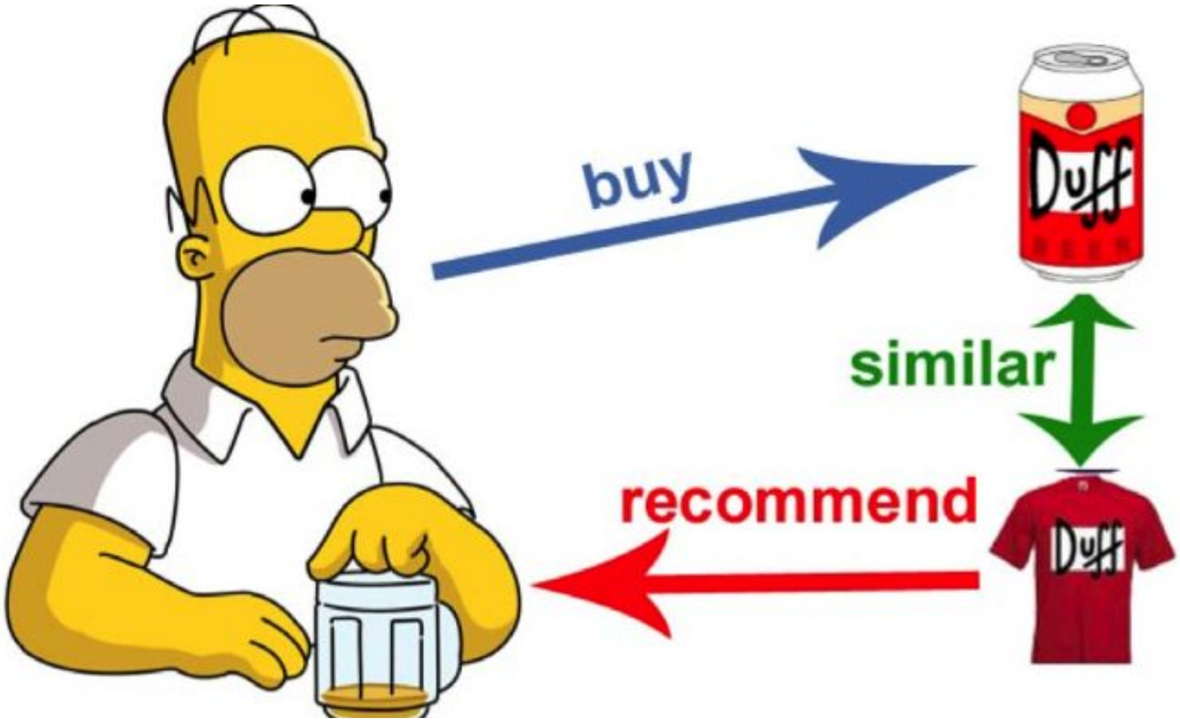
Main idea: Making automatic predictions about the interests of a user (filtering) by collecting preferences from many many users (collaborating)

Latent Factor Based

Main idea: Users and items are are characterized by latent factors (hidden and specific to domain)

Same as ‘concepts’ we discussed in class (PCA, SVD)

Content Based Recommendations



How to Find Similar Items – We've Done This!

1. Turn items into vectors based on features
2. Compute Similarities of items to other items (cosine similarity)
3. Make a user profile and find missing ratings

Content Based: Pros and Cons

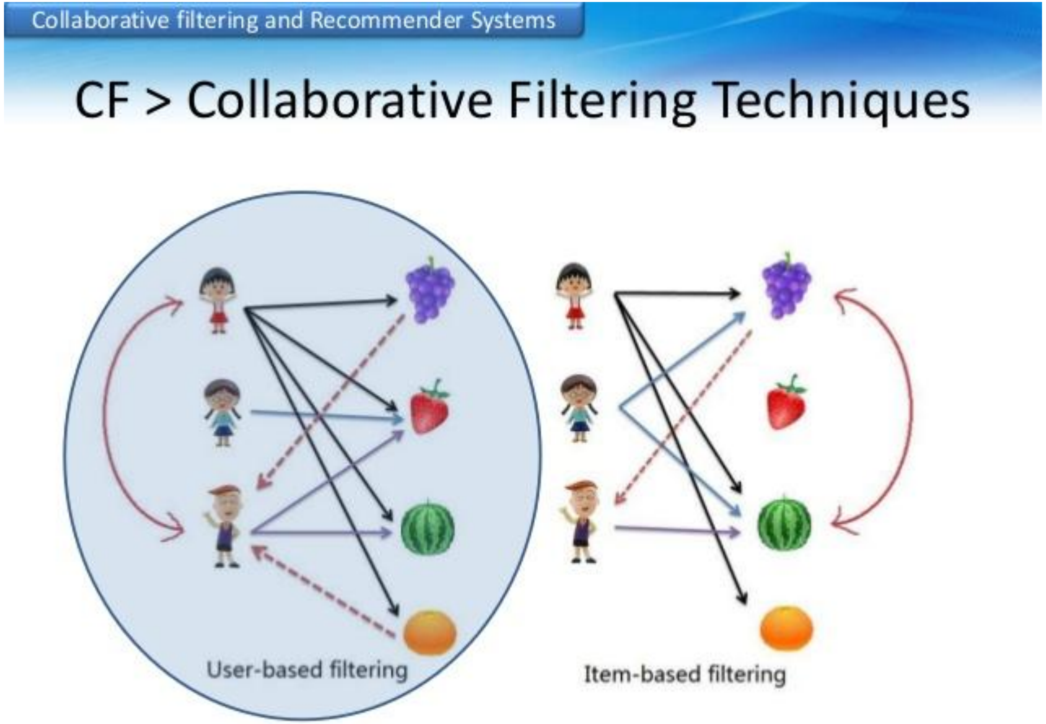
Pros

- Don't't’ need any ratings
- Can recommend new items or unpopular items
- Interpretability
  - Not a black box
  - You can can explain recommendations because we have the features

Cons

- Feature Selection is hard
  - Extremely hard to ‘evaluate’ best subset
- Overfit to user’s profile
  - Doesn’t incorporate user’s many tastes
- Doesn’t incorporate other user’s ratings or activity
  - Missing out on a treasure trove of information

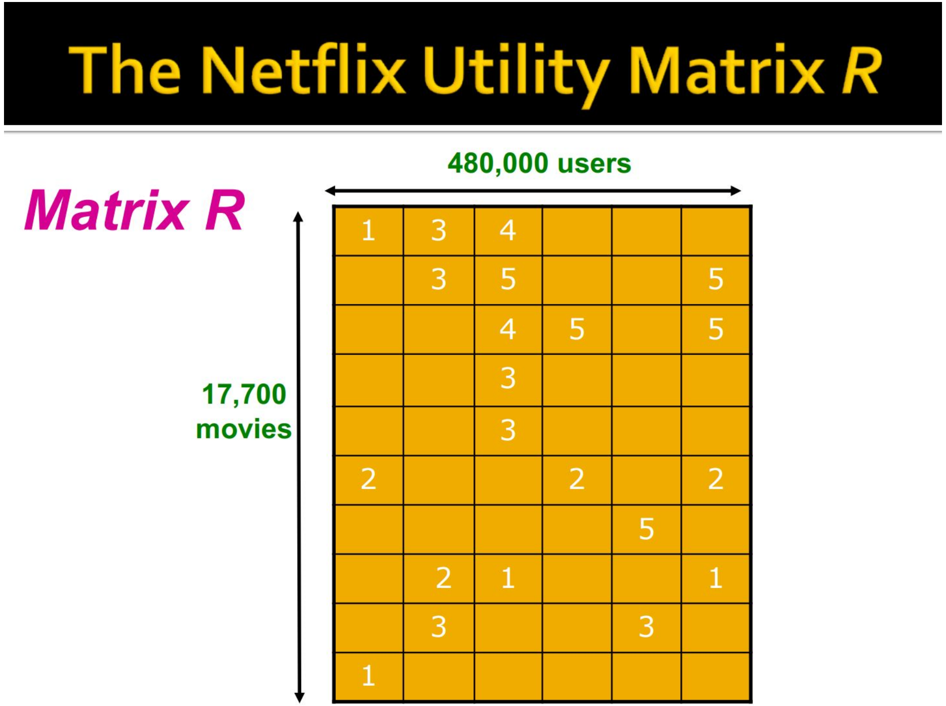
Collaborative Filtering



Using a Utility Matrix

What is the matrix?

- User-Item or Item-User matrix where ratings are the values
- Extremely sparse
- Extremely large



What if I never rate things?

Explicit Ratings

- Actually rate items
- Reviews (sentiment analysis)
- Pay people to rate things (surveys)

Implicit Ratings

- Guess ratings from user's actions
  - Buying an item is 'good' rating
  - Looking at items is 'good' rating
  - Returning an item is 'very bad'

Two Main Types

Item-Item CF

Look for N items that are “similar” to the items that user X has already rated (highly) and recommend most similar items

User-User CF

Find set N of other users whose ratings are “similar” to X’s ratings

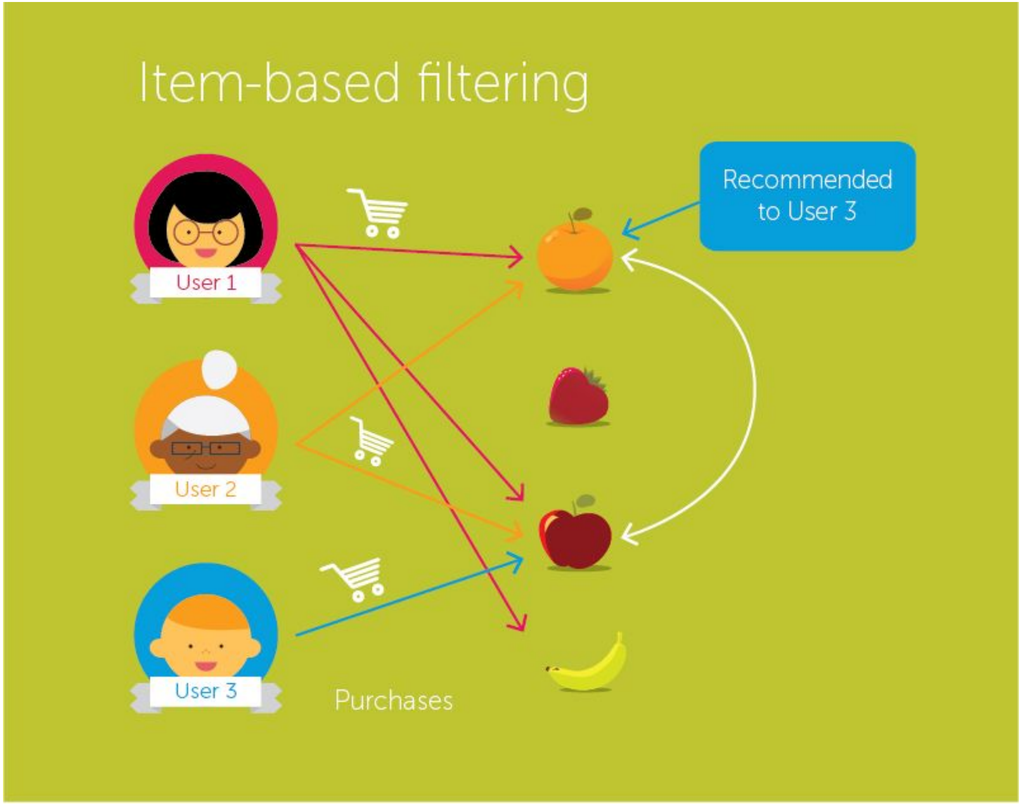
BUT WAIT!

How is Item-Item Collaborative Filtering any different than Content Based Recommendations?

ANSWER

Item-Item Collaborative Filtering does not use 'item features' in its similarity function. It finds similar items based on 'item neighborhoods' which are created based on user behaviour (usually ratings).

Item-Item Collaborative Filtering



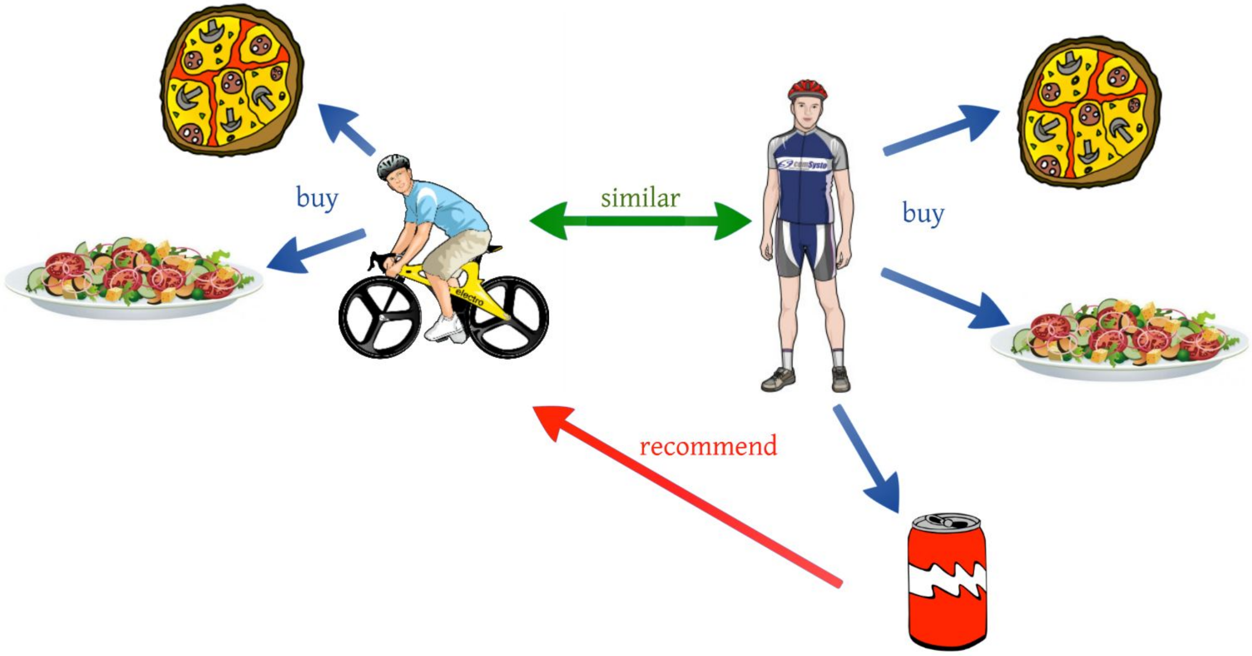
Item-Item Collaborative Filtering

For item i, find other similar items

Estimate rating for i based on ratings for similar items

Do this for every missing item in user's profile

User-User Collaborative Filtering



User-User Collaborative Filtering

Consider a User X

Find set N of other users whose ratings are “similar” to X’s ratings

Estimate X’s missing ratings based on ratings of users in N

Item-Item vs User-User

- *Item-Item is almost always better than User-User*
- *Why? Because items are simpler and users have multiple ‘tastes’*

Collaborative Filtering: Pros and Cons

Pros

- Works on any type of item/product
  - Don’t need features
  - Amazon uses this so they can treat everything as a ‘product’

Cons

- Cold Start Problem
- User-Item Rating matrix is always sparse
- New items can’t be recommended
- ‘Harry Potter’ problem. CF usually recommends super popular items
  - We saw this in our project

Latent Factor Models

Latent Factor Models – Factorization with SVD

SVD Provides:

- User-Concept Matrix
  - P (reduced V)
- Item-Concept Matrix
  - Q (reduced U)

So What?

- We can calculate ratings of an item for a user using Product of Factors
- Multiply row in item-concept matrix by column in concept-user
- We did this on the midterm!

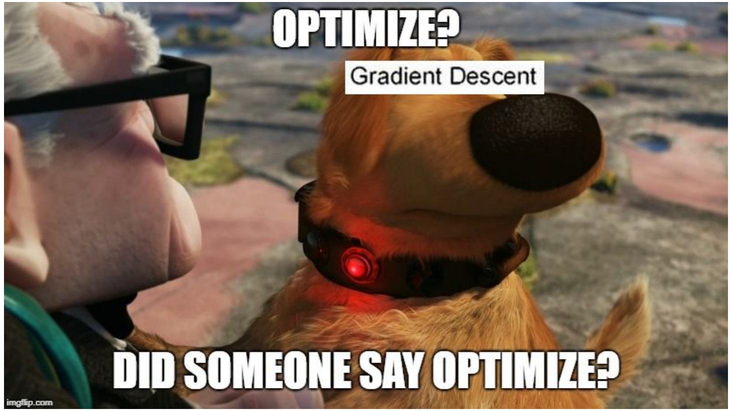
Latent Factor Models - Optimizing Latent Factors

SVD Properties

- SVD is a ‘perfect’ reconstruction of the Utility Matrix
  - HUGE issue: missing ratings are treated as 0 by SVD
  - This is really not logical
- We’re trying to predict ratings!

What can we do?

- Optimize the components of SVD (P and Q) to give best predictions to test data



Latent Factor Optimization

1. Initialize P and Q (missing values as 0 like normal)
2. Minimize SSE on training ratings (be careful to overfit!)
3. Do Gradient Descent on each element of P and Q
4. Repeat until convergence. Result is a better P and Q for prediction!

Which should I use?

- It depends
- CRISP-DM - Domain understanding is critical for recommendation systems
- Almost all ‘advanced’ systems are Hybrid Systems
- Latent Factor being the most popular ‘core’ component

What about this Neural Network stuff?

Yes! You can use Neural Networks and Deep Learning to enhance recommendation systems

“Neural networks are the second best way of doing just about anything" - John Denker

Neural Network Applications

Embedding

In Content Based systems, items can have massive amounts of features

Embed the item features into a more condensed, ‘better’, learned representation

Latent Factor Mapping

Deep Convolutional Networks have been used to find optimal P and Q matrices

Item2Vec

Learn embeddings for collaborative filtering (learning optimal user and item neighborhoods)

Trust Based Recommender Systems

Another way we can improve recommendations is to add a layer of 'trust' to them. We often prefer the recommendation of someone we 'trust' more than a recommendation from a random person or some black box algorithm.

Neural Network Applications

Calculating Trust

‘Trust’ is a very subjective term and hard to find

Researchers have been using Social Graphs to calculate ‘trust’ mathematically

Very similar to Prof. Bari’s Flockmates and Leaders work

No Social Graph?

You can actually model Utility Matrix as a graph!

Rating, Buying, etc an item creates an ‘edge’

You can now calculate ‘neighborhoods’ and infer trust between them

Trust Based Ant Recommender System (TARS)

Adds pheromones to users as they gain popularity

New users (cold start problem) are instantly attracted to high pheromone users

Sources

- http://www.mmds.org/
- https://arxiv.org/ftp/arxiv/papers/1603/1603.04259.pdf
- https://arxiv.org/pdf/1606.07659.pdf
- https://nycdatascience.com/blog/student-works/deep-learning-meets-recommendation-systems/
- http://www.ideal.ece.utexas.edu/seminar/LatentFactorModels.pdf
- https://www.sciencedirect.com/science/article/pii/S0957417411010864
- https://www.info.ucl.ac.be/~pvr/TrustBasedRecommendation.pdf

THANKS!

Any questions?