Introduction

- *Recommender Systems* predict what you'll like
- Put another way: they predict your preferences
- What are some examples?

Example

- Recommender Systems predict what you'll like
- Put another way: they predict your preferences
- More examples
  - Products
  - Movies
  - Music
  - Books
  - Video games
  - Colleagues
  - Friends

Recommendation Background

- Amazon wants you to buy nice stuff
  - Partially makes up for difficult browsing
- Netflix wants you to like what you're watching
- In the old days, they especially liked when you liked old movies
  - Netflix needed to have at least one copy of many titles, to keep selection up
  - If everyone wants new releases, Netflix needed to buy a lot of copies of one title
  - Much nicer for Netflix if you rented something no one else wanted
  - Offered Netflix Prize; more later

Recommendation Challenges

- Recommendation is common, but surprisingly hard
- Lots of recommendations to make
- 10,000s of products
- Users have very little tolerance for adding preference data; system knows almost nothing about you
- Everyone is different (right?)

Algorithms

- How to predict what movies you like? Features?
- One approach: do it like Web pages
  - Collect data on my movie likes

The Godfather	4
Ernest Goes to Camp	3
Casablanca	2
36 Hours	5
Love and Death	4

  - Collect features: genre, length, year, etc
  - Build score-predictor; recommend high-scorers
- Problems?

Collaborative Filtering

- Unfortunately, film-qualities (*features*) may be difficult to extract
- How to recommend movies without knowing anything about movies?
  - Recommend movies enjoyed by people who are similar to you

Example

	W.	Xanadu	Youngblood	Zorro
Alice	4	2	4	4
Bob	?	2	5	1
Chris	4	2	4	?
Donna	3	?	5	1

How can we estimate scores?

- Filling in missing scores
- Approach #1: average for film

Example

- Average over all users

	W.	Xanadu	Youngblood	Zorro
Alice	4	2	4	4
Bob	3.66	2	5	1
Chris	4	2	4	2
Donna	3	2	5	1

How can we estimate scores?

- Approach #1: average for film
- Approach #2: use rating of closest user
- One way to find user-closeness is with an approach similar to tf-idf
  - Consider u and v's vectors of ratings
    - Each movie is a dimension
    - Each score is a weight
  - Compute cosine(u, v), just as with tf-idf info-retrieval doc-ranking
- What is the equivalent of "inverse doc frequency"?

How can we estimate scores?

- Approach #1: average for film
- Approach #2: use rating of closest user
- One way to find user-closeness is with an approach similar to tf-idf
  - Consider u and v's vectors of ratings
    - Each movie is a dimension
    - Each score is a weight
  - Compute cosine(u, v), just as with tf-idf info-retrieval doc-ranking
- What is the equivalent of "inverse doc frequency"?
  - Inverse-user-frequency. Rarely-seen movies are more influential

User similarity

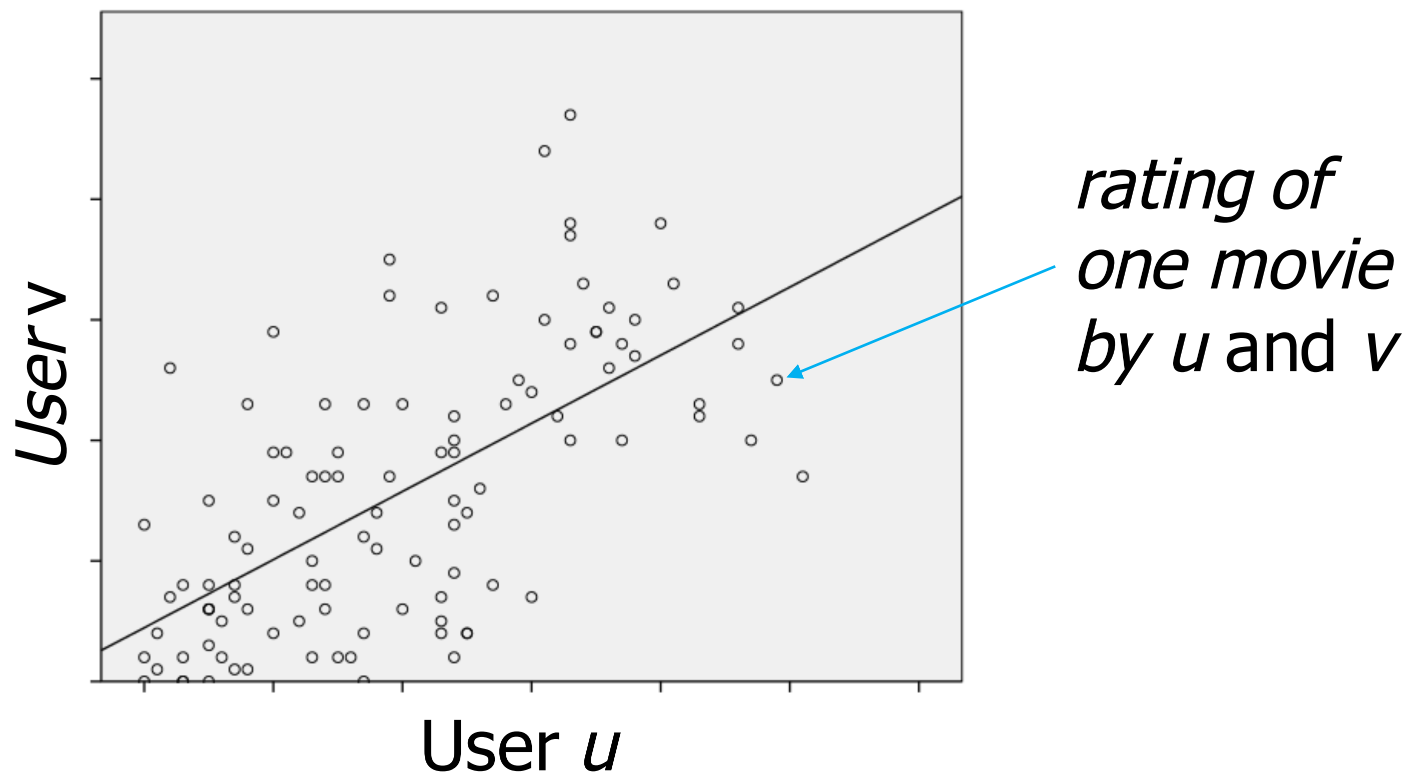
- Another method is Pearson correlation
  - S = set of movies
  - $r_{u,i}$ = rating of user u on movie *i*
  - $S_u = \{i \in S \mid u \text{ saw movie } i\}$
  - $S_{uv} = \{i \in S \mid \text{both } u \text{ and } v \text{ saw movie } i\}$

$$r_u = \sum_{i \in S_u} r_{u,i} / \left|S_u\right|$$

$$\frac{\sum_{i \in S_{uv}} (r_{u,i} - r_u)(r_{v,i} - r_v)}{\sqrt{\sum_{i \in S_{uv}} (r_{u,i} - r_u)^2 \sum_{i \in S_{uv}} (r_{v,i} - r_v)^2}}$$

User similarity

- Pearson correlation
  - How related are ratings from two users?



Example

- Chris' closest match is Alice, so...

	W.	Xanadu	Youngblood	Zorro
Alice	4	2	4	4
Bob	?	2	5	1
Chris	4	2	4	?
Donna	3	?	5	1

Example

- Chris' closest match is Alice, so...

	W.	Xanadu	Youngblood	Zorro
Alice	4	2	4	4
Bob	3	2	5	1
Chris	4	2	4	4
Donna	3	2	5	1

Can't Pick Just One

- Can also weight by similarity

$$r_{u,i} = k \sum_{v \in Top-Sim(u)} sim(u,v) r_{v,i}$$

- Where Top-Sim(u) is the n most-similar user neighbors to u
- k is a normalizer

$$k = 1 / \sum_{v \in Top-Sim(u)} \left| sim(u,v) \right|$$

Can't Pick Just One

- Can also re-add user average scores

$$r_{u,i} = r_u + k \sum_{v \in Top-Sim(u)} sim(u,v)(r_{v,i} - r_v)$$

Miscellaneous

- We've seen a collaborative filtering algorithm:
  - User-based
  - Neighborhood-based (top-N neighbors)
- What is the time-complexity of finding nearest-neighbor?
  - With locality-sensitive hashing, linear
  - Remember efficient shingling? The algorithm we saw is example of locality-sensitive hashing
- Many analogies between doc IR and collab. filtering
  - What would doc *d* say about term *t*, even though it is silent on *t* now?

Other Algorithms

- Problems so far?
  - What if data is sparse, i.e., a user can only rate a tiny number of products?
  - What if the # of users and products is millions each? Computationally difficult
- One solution: item-based filtering
  - Avoid nearest-neighbor operations on users
  - Recommend products similar to ones the user has liked in the past

Item-Based Filtering

- For test item *i*, find *k* most-similar items the user has rated previously
  - *i*'s score is a weighted combination of user's ratings on those *k* items
- How can we compute item similarity?
  - Can use cosine or correlation, as before
  - The vector for item *i* is the set of user-reviews associated with *i*

Model-Based Efficiency

- We can also skimp on the item-item model. New algorithm:
- For each item *j*, compute *k* nearest-items, where *k* << *n*
- When predicting *u*'s opinion of *i*, retrieve the *k* nearest-items for *i*. Predict score based on the subset of *k* that *u* has rated
- Tradeoff between quality and model size!

Netflix Prize

- Announced October 2, 2006
- $1M to whoever could improve NetFlix's own recommender by 10%
- Data from Netflix in the form:
    - <user, movie, date, grade>
    - Where grade is 1...5
    - That's it
- Training data: 100M examples from ~500k users on ~18k movies
- Ideas for how you might do this?

Netflix Prize

- How did they do it?
- Among many ideas:
  - Use IMDB for director, genre, etc
  - Some movies' grades change over time; they "age" well or poorly
  - User grades depend on day of week

Netflix Prize

- Dan Tillberg's progress
- Dropped out in 2008 while in 5th place