Recommender Systems

Content-based systems

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

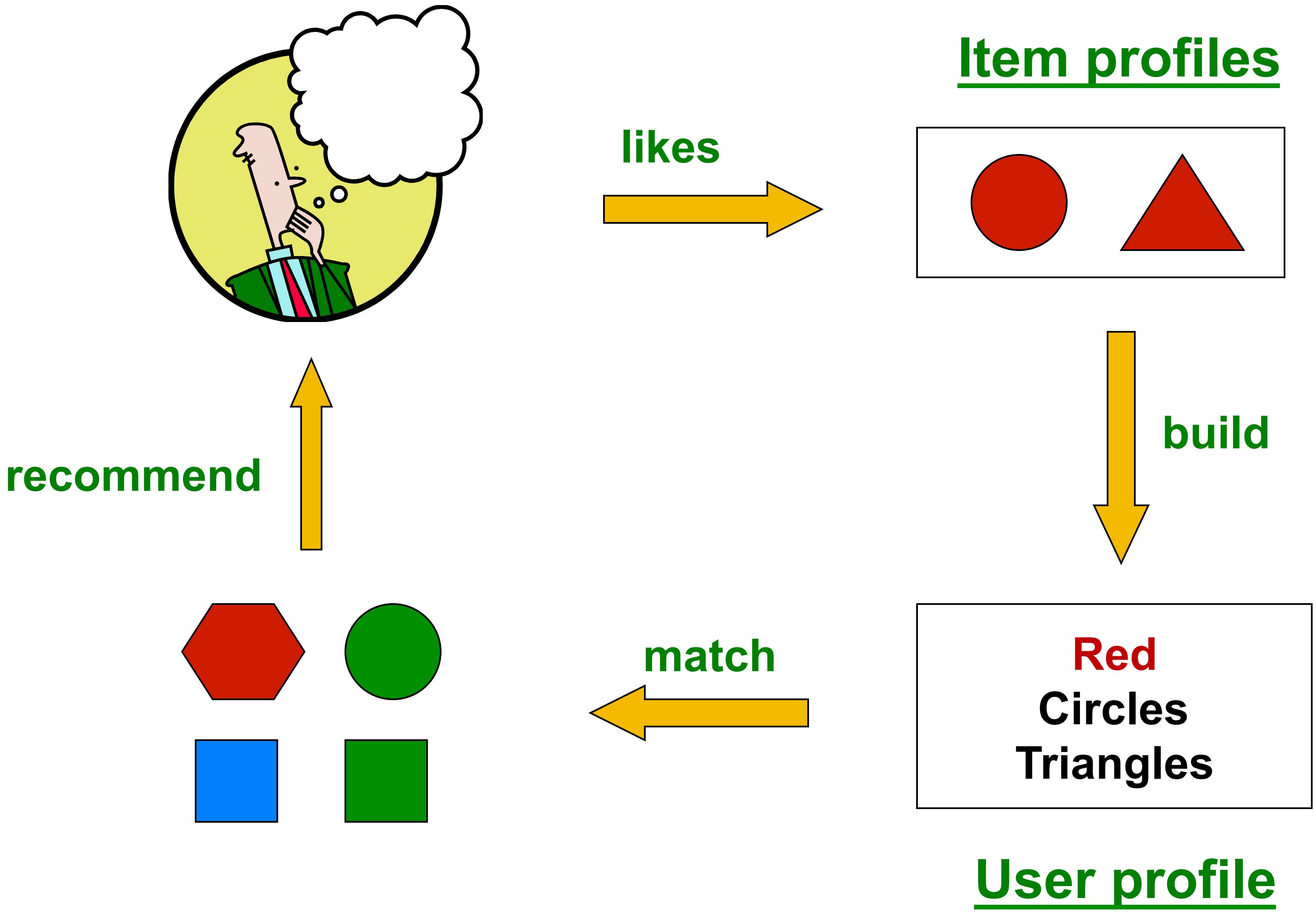
Content-based Recommendations

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

Examples:

- Movies
  - Same actor(s), director, genre, …
- Websites, blogs, news
  - Articles with "similar" content
- People
  - Recommend people with many common friends

Plan of Action



Item Profiles

- For each item, create an **item profile**
- Profile is a set of features
  - **Movies:** author, title, actor, director,…
  - **Images, videos:** metadata and tags
  - **People:** Set of friends
- Convenient to think of the item profile as a vector
  - One entry per feature (e.g., each actor, director,…)
  - Vector might be boolean or real-valued

Text features

- Profile = set of "important" words in item (document)

- How to pick important words?
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)

Sidenote: TF-IDF

f_{ij} = frequency of term (feature) i in doc (item) j

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Note: we normalize TF to discount for "longer" documents

n_i = number of docs that mention term i

N = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest **TF-IDF** scores, together with their scores

User Profiles

- User has rated items with profiles $\mathbf{i_1,\ldots,i_n}$
- Simple: (weighted) average of rated item profiles
- Variant: Normalize weights using average rating of user
- More sophisticated aggregations possible

Example 1: Boolean Utility Matrix

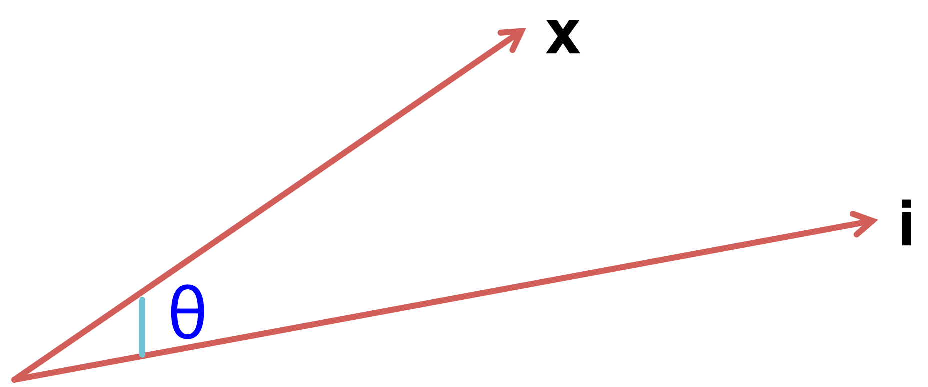
- Items are movies, only feature is "Actor"
  - Item profile: vector with 0 or 1 for each Actor
- Suppose user x has watched 5 movies
  - 2 movies featuring actor A
  - 3 movies featuring actor B
- User profile = mean of item profiles
  - Feature A's weight = 2/5 = 0.4
  - Feature B's weight = 3/5 = 0.6

Example 2: Star Ratings

- Same example, 1-5 star ratings
  - Actor A's movies rated 3 and 5
  - Actor B's movies rated 1, 2 and 4
- Useful step: Normalize ratings by subtracting user's mean rating (3)
  - Actor A's normalized ratings = 0, +2
    - Profile weight = (0 + 2)/2 = 1
  - Actor B's normalized ratings = -2, -1, +1
    - Profile weight = -2/3

Making predictions

- User profile **x**, Item profile **i**
- Estimate $U(\mathbf{x},\mathbf{i}) = \cos(\theta) = (\mathbf{x} \cdot \mathbf{i})/(|\mathbf{x}|\,|\mathbf{i}|)$



Technically, the cosine distance is actually the angle θ
And the cosine similarity is the angle $180-\theta$

For convenience, we use $\cos(\theta)$ as our similarity measure and call it the "cosine similarity" in this context.

Pros: Content-based Approach

- No need for data on other users
- Able to recommend to users with unique tastes
- Able to recommend new & unpopular items
  - No first-rater problem
- Explanations for recommended items
  - Content features that caused an item to be recommended

Cons: Content-based Approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
- Cold-start problem for new users
  - How to build a user profile?