Item-Item Collaborative Filtering

Recommender Systems

Dimitris Sacharidis

memory-based collaborative filtering

- some history
- User-User Collaborative Filtering
- **Item-Item Collaborative Filtering**
- dealing with Implicit Feedback
- cold start problems

Item-Item Collaborative Filtering

item similarities

- how can we compute similarity between items?
  - without **content**, i.e., item descriptions

- item similarities can be quantified by comparing the set of users that like them
  - or more precisely, by their ratings

- two items are **similar** if they *receive similar ratings* by users
  - John likes items 1 and 2; Mary dislikes items 1 and 2
  - then, items 1 and 2 are similar to each other

item similarities

- item similarities are more **stable** than user similarities

- why?
  - typically, number of users much greater than number of items
  - hence, an average item has more ratings than an average user
  - the description of items does not change much with new ratings

- addresses shortcomings of U-U CF

Item-Item Collaborative Filtering

To predict rating of target user to target item

- idea: (pivot what we did in U-U CF)

- compute **similarity** of target item to each other item
- create a **neighborhood** of the target item
- predict rating as the **weighted average rating** the target user gives to the neighbors of the target item

[2001 WWW B. Sarwar et al] *Item-based collaborative filtering recommendation algorithms*

starting from a baseline Item-Item CF

- we want to predict a rating for the target item
  - baseline only considers the target user

- we can take the **average rating** of the target user…
  - (in U-U CF, we took the average rating of the target item)

$$s(u,i) = \frac{\sum_{j \in I_u} r_{uj}}{|I_u|}$$

- … across all items rated by the target user

$$I_u = \{i \in I | r_{ui} \in R\}$$

incorporate additional ideas

- **weighted average**: not all items contribute equally
- **neighborhood**: only highly similar items should contribute
- **normalize (mean-center) item ratings**: predict deviations from the target item's mean rating
- Item-Item CF prediction formula:

$$s(u,i) = \overline{r_i} + \frac{\sum_{j \in N(i)} w_{ij}(r_{uj} - \overline{r_j})}{\sum_{j \in N(i)} |w_{ij}|}$$

- $N(i)$ is the neighborhood of the target item
- w_{ij} is the similarity between items i, j

how to compute weights

- weights (pairwise item similarities) are computed using **Pearson correlation** (= mean-centered cosine similarity):

$$w_{ij} = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \overline{r_i})(r_{uj} - \overline{r_j})}{\sqrt{\sum_{u \in U_i} (r_{ui} - \overline{r_i})^2} \sqrt{\sum_{u \in U_j} (r_{uj} - \overline{r_j})^2}}$$

- where $U_i = \{u \in U | r_{ui} \in R\}$ is the subset of users that have rated item i

item neighborhood selection

- to predict the rating of target item to the target user
  - create a neighborhood of the target item that includes *only items that the target user has rated*

- neighborhood may be *limited* according to standard ideas
  - limit neighborhood to top-k most similar items
  - limit neighborhood so that neighbors have similarity above a threshold
  - include items with high negative (anti-) similarity

- standard practice: take 20 neighbors

Item-Item CF algorithm

To recommend an item to target user u

- for each item $i \notin I_u$ compute its **similarity** to every other item $j \in I_u$

$$w_{ij} = \frac{\sum_{u \in I_i \cap I_j} (r_{ui} - \overline{r_i})(r_{uj} - \overline{r_j})}{\sqrt{\sum_{u \in I_i} (r_{ui} - \overline{r_i})^2} \sqrt{\sum_{u \in I_j} (r_{uj} - \overline{r_j})^2}}$$

- create a **neighborhood** $N(i)$ of the target item by selecting items with the highest w_{ij}
- for each item $i \notin I_u$ compute its **predicted rating**

$$s(u, i) = \overline{r_i} + \frac{\sum_{j \in N(i)} w_{ij} (r_{uj} - \overline{r_j})}{\sum_{j \in N(i)} |w_{ij}|}$$

- finally, recommend the item with the highest predicted rating

computational complexity of I-I CF

- how fast is it? recall:
  - m is the number of users, ~ millions
  - n is the number of items, ~ thousands

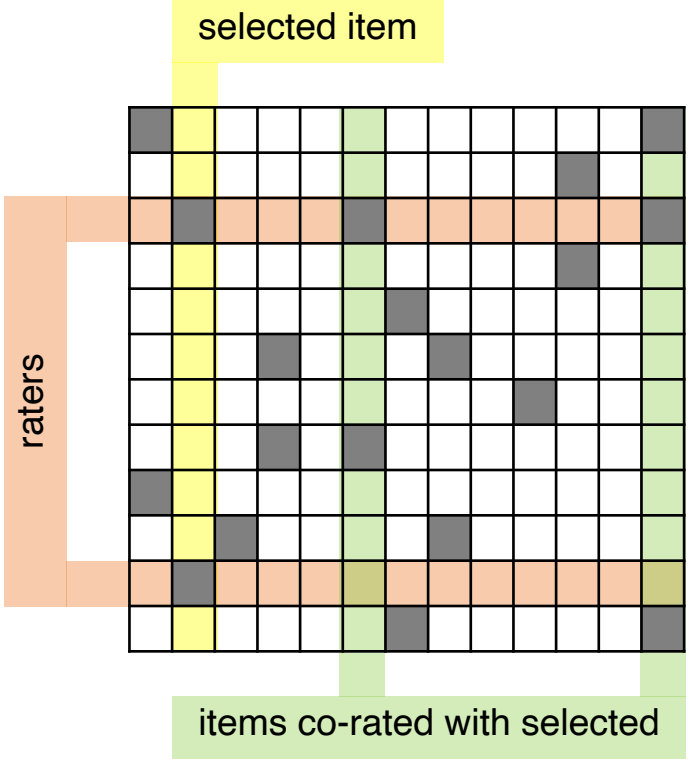
- computing similarity between two items takes $O(m)$
- computing all pairwise item similarities takes $O(n^2 m)$
- creating a neighborhood of fixed size k takes $O(n)$
- predicting rating for a single item takes $O(k)$
- selecting the item with the highest predicted rating takes $O(n)$

- costs dominated by similarity computations
  - but quadratic on number of items

optimizations to I-I CF

- Optimization 1: similarities computation
  - exploit the sparsity of the ratings

- for each item, called **selected**
- identify its **raters**
- identify other **items co-rated** by those users
- compute **similarity** between selected and each of the co-rated items



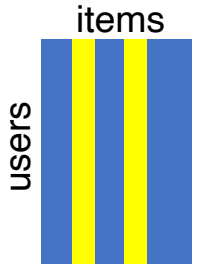
[2003 IEEE G. Linden et al] *Amazon.com Recommendations*

optimizations to I-I CF

- Optimization 2: build an offline model
- works because item similarities are more stable
- **offline phase**:
  - for each item, identify its K neighbors
  - only store similarities for the selected item-neighbor pairs
- **online phase**:
  - to create the item neighborhood, only consider items for which the similarity is stored
  - the neighbors selected may be less similar than those selected by the standard I-I CF,
    - why? we only store a subset of all possible pairwise similarities;
  - but in practice works as well

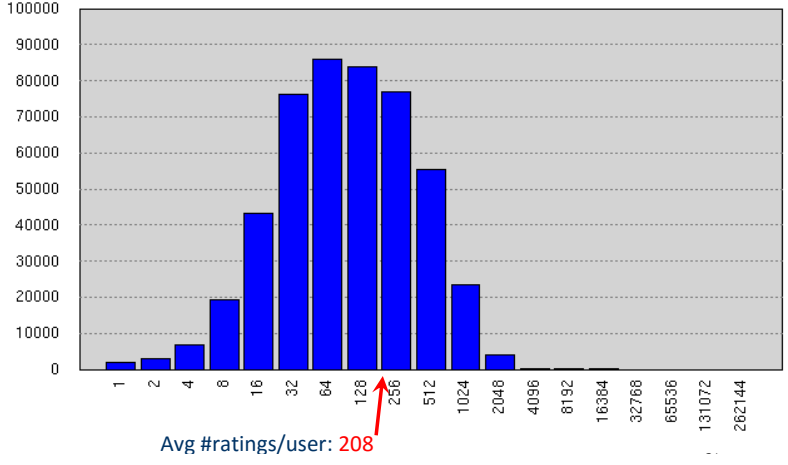
[2004 TOIS M. Deshpande, G. Karypis] *Item-Based Top-N Recommendation Algorithms*

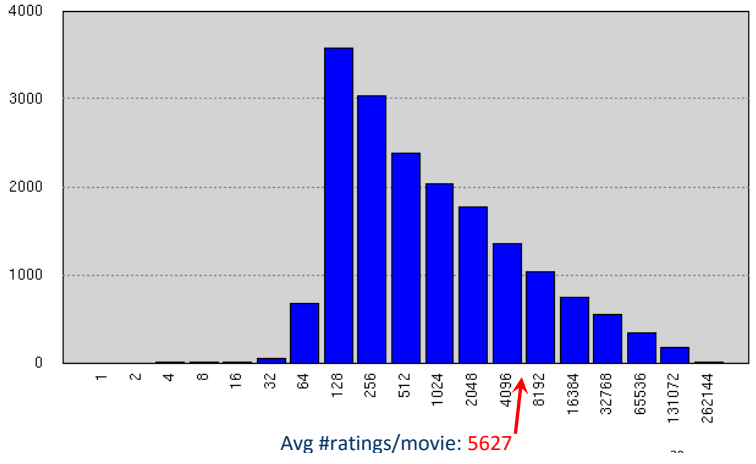
strengths of I-I CF



+ more **efficient** than U-U CF
  + when there are more users than items
  + comparing columns pairwise

+ item-item similarities more **stable** than user-user
  + item profile (a column) contains more ratings than a user profile (a row)
  + can precompute similarities offline





[plots by M. Zanker]

strengths of I-I CF

+ more **effective** than U-U CF

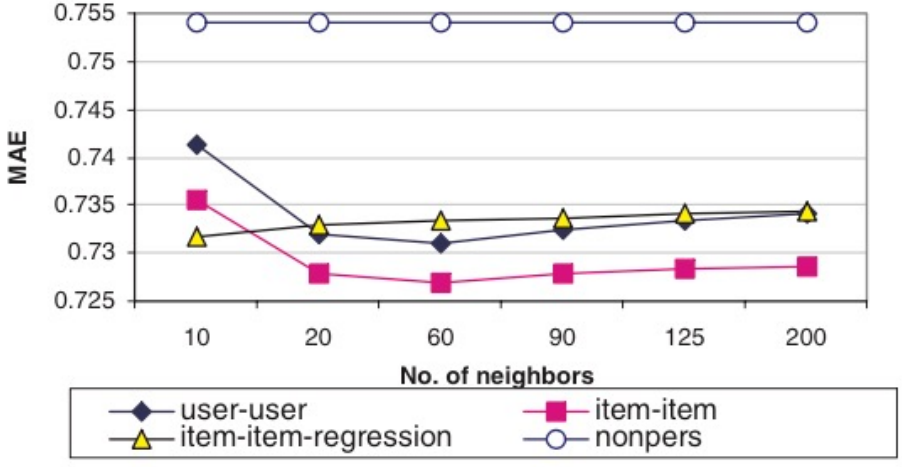
y-axis is error: lower values are better

x-axis is neighborhood size

U-U is black diamond

I-I is red square

non-personalized **average rating** is empty circle



[2001 WWW B. Sarwar et al] *Item-based collaborative filtering recommendation algorithms*

weaknesses of I-I CF

- recommendations **too obvious**
  - why? recommended items are similar to the ones a user already rated
  - less chance to recommend unexpected items
    - in U-U CF you might get a bold item recommendation (e.g., for an unknown movie) because someone similar to you enjoyed it
  - fix: consider metrics besides accuracy: serendipity, novelty, diversity

Acknowledgements

some ideas from:

- Joseph A. Konstan, Michael D. Ekstrand. Recommender Systems Specialization on coursera