User-User Collaborative Filtering

Recommender Systems

Dimitris Sacharidis

Collaborative Filtering (CF)

- currently **the most well-known approach** to generate recommendations
  - thanks to successes at the Netflix prize

Characteristics

- based on **User-Item interactions** alone
- extract and exploit patterns from the “wisdom of the crowd”, the collective user behavior
- **no content information** necessary about items
  - difference with e.g., content-based recommenders
  - domain agnostic: ideas apply to other types of items

Collaborative Filtering (CF)

Input Data

- **users** give **ratings** to **items**

Assumption

- users' behavior (i.e., ratings) is guided by their **preferences**
- users' preferences remain stable and consistent over time
- thus, we can expect similar users to prefer similar items

Classification

- **Memory-based Methods**
  - remember the entire *history* of user-item interactions
  - e.g., Neighborhood methods: User-User and Item-Item

- **Model-based Methods**
  - build a model that *describes* the history, and make recommendations from that
  - e.g., Matrix Factorization

Memory-based Collaborative Filtering

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

some history

the Tapestry System

- CF term introduced in the context of the **Tapestry** system at Xerox PARC to manage e-mails from newsgroups, mailing lists

- system goal: identify emails that are interesting to the user

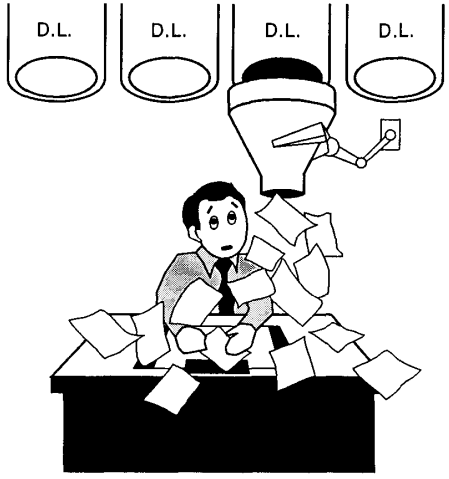
Existing options

- mailing lists: subscription to a topic
- conventional filtering: specify criteria an email must satisfy

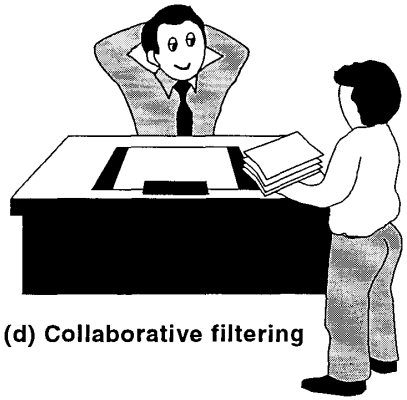
[1992 CACM D. Goldberg et al] *Using collaborative filtering to weave an information tapestry*

(a) Electronic mail overload

(b) Using distribution lists



(c) Conventional filtering



(d) Collaborative filtering

the Tapestry System

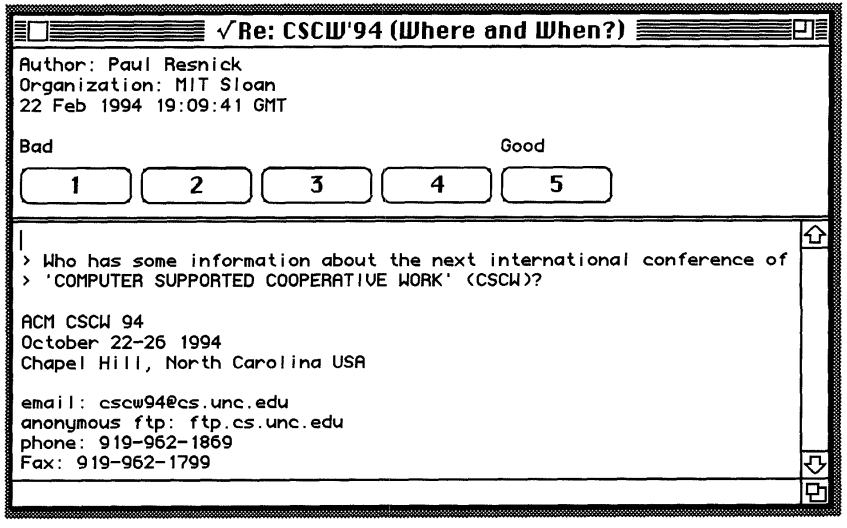
- Tapestry records user interactions with emails
  - **ratings** = explicit feedback, e.g., this email was interesting
  - **implicit feedback**, e.g., John has replied to this email

- and allows users to do “collaborative filtering”, i.e., filter emails based on other user’s interactions
  - e.g., show me emails that John has replied to

- Tapestry was *not a recommender system* per se, but included all the concepts necessary to build one

the GroupLens System

- Building on the idea of Tapestry, the GroupLens system collected ratings of users to emails
- but also attempted to ***predict the rating*** a user might give to an email
  - with a user-based CF method we discuss later



[1994 CSCW P. Resnick et al.] *GroupLens: an open architecture for collaborative filtering of netnews*

User-User Collaborative Filtering

notation and terminology

- **Users** $U,\ |U| = m$
- **Items** $I,\ |I| = n$
- **Ratings** R , rating $r_{ui} \in R$ of user $u \in U$ to item $i \in I$
  - in some domain, say 1 to 5 stars

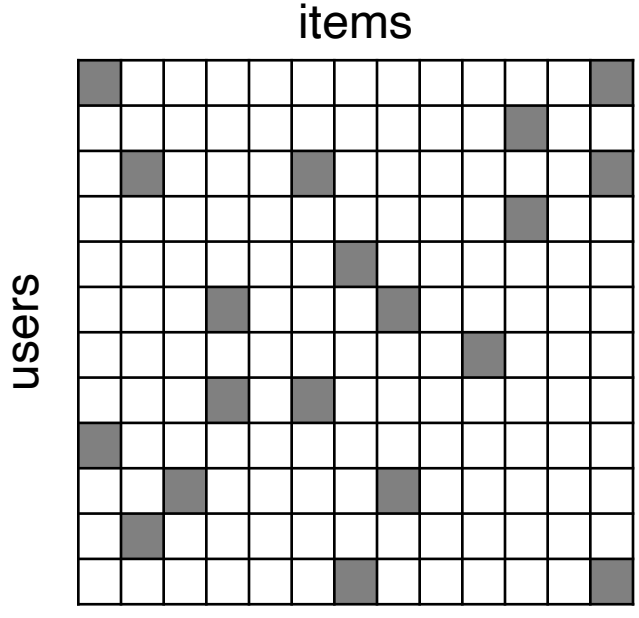
- Predicted rating $\hat{r}_{ui}$ or **score** $s(u, i)$
  - for some user-item pair that we have no rating
  - u is called the **target user**
  - i is called the **target item**

User-Item Ratings Matrix

Feedback/Ratings is all you got!

- **Ratings** seen as a sparse **matrix**
  - rows represent users
  - columns represent items

- grey cells indicate that a rating exists
- typically **very sparse**
- e.g., in MovieLens 20M:
  - 27,278 items
  - 138,493 users
  - 20,000,263 ratings = 0.5% of possible ratings



User-User Collaborative Filtering

- goal: predict the rating of a target user to a target item
- idea: **combine** the ratings of **similar users**
  - if like-minded users liked this movie a lot, then you most probably are going to like it

- three important questions to answer:
  - how to define user similarity (**similarity function**)
  - how many similar users to consider (**neighborhood selection**)
  - how to combine ratings (**prediction formula**)

(also called user-based CF, user-based neighborhood CF)

[1999 SIGIR J. L. Herlocker et al] *An algorithmic framework for performing collaborative filtering*

User-User Collaborative Filtering

- let's start simple and progressively add complexity
    - non-personalized
    - make it personal
    - remove rating bias
    - compute similarity weights
    - consider similarity neighborhood

non-personalized User-User CF

- we want to predict a rating for the target item
  - non-personalized = independent of the target user

- we can take the **average rating** for the item…

$$s(u,i) = \frac{\sum_{v \in U_i} r_{vi}}{|U_i|}$$

- … across all users that have rated the target item

$$U_i = \{u \in U | r_{ui} \in R\}$$

make it personal

- the target user is (probably) *not* like the average user
- her/his preferences may be more **similar** to some users and more **dissimilar** to others
- suppose we have a magical way to quantify user similarity
- w_{uv} denotes the **similarity** between users u, v
  - ranging from -1 (complete anti-similarity) to 1 (complete similarity)

make it personal

- instead of average, compute the **weighted average rating** for the target item
  - where a weight is the similarity of each user to the target user

- predicted rating becomes:

$$s(u,i) = \frac{\sum_{v \in U} w_{uv} r_{vi}}{\sum_{v \in U} |w_{uv}|}$$

rating here takes positive and negative values

- **in practice**: we sum over all other users who have rated the item
- why the absolute value in the denominator?
  - recall what a **negative value** represents

but remove rating bias

- users might have *different* interpretations of the **rating scale**
  - e.g., I'm positive and give bad movies a 3 out of 5
  - but you might be less forgiving and use the full scale

- how do we account for these differences in the personal rating scales?

- a better indication of a user's preference towards an item is how far a rating is from the user's mean (average) rating
  - e.g., if my mean rating is 4, then a rating of 4 is probably a **weak endorsement** by me
  - but if your mean rating is 2, then a rating of 4 is probably a **very strong endorsement** by you

but remove personal bias

- instead of ratings, use **deviations** (how far off ratings are) from users' mean ratings
  - a general normalization principle called *center around the mean*

- in previous formula replace ratings r_{vi} with deviations $r_{vi} - \overline{r_v}$
  - where $\overline{r_v}$ is the mean rating of user v
  - but don't forget to add back the mean rating of the target user!

- updated formula:

$$s(u,i) = \overline{r_u} + \frac{\sum_{v \in U} w_{uv}(r_{vi} - \overline{r_v})}{\sum_{v \in U} |w_{uv}|}$$

one small caveat

- predicted ratings can go **outside the rating scale**

- if my mean rating is 4 (out of 5) and all users give +2 above their average
- then my predicted rating is 6

- in practice, you can simply restrict the predicted ratings at a final step

computing user similarities

- remember those weights that capture user-user similarity?

- many ways to compute similarities

- typically you want a weight of
  - 1 to mean complete **similarity**,
  - 0 to mean **no similarity**, and
  - -1 to mean complete **anti-similarity** (I say love, you say hate)

- gold standard is the Pearson correlation (PCC)
  - with some small adaptation
  - similar to the mean-centered cosine similarity

Pearson correlation

Intuition

- correlation: to what extent **ratings** agree

- (absolute) correlation increases
  - with **frequency** of agreement: we like and dislike the same movies
  - and with **intensity** of agrement: we strongly like/hate movies

- correlation is close to zero
  - if there is no clear pattern

Pearson correlation

- the similarity of users u, v is:

$$w_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2}}$$

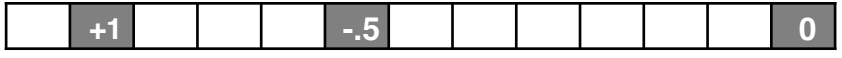
- where $I_u = \{i \in I | r_{ui} \in R\}$ is the set of items rated by u
- numerator sum is over **common items** (rated by both users) and computes **to what extent the rating deviations of one user agree with those of the other**
- denominator sums are over items rated by each user independently and normalizes similarity to $[-1, 1]$

Pearson correlation - example

$$w_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2}}$$

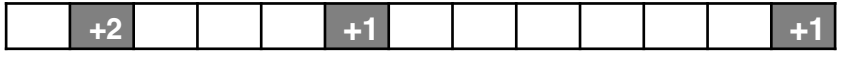
rating vectors: rows of the rating matrix (after mean-centering)

u



$$\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} = \sqrt{1^2 + (-0.5)^2 + 0^2} \approx 1.1$$

v



$$\sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2} = \sqrt{2^2 + 1^2 + 1^2} \approx 2.4$$

+2 **-.5** **0**

$$\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v}) = 2 - 0.5 + 0 = 1.5$$

$$w_{uv} \approx \frac{1.5}{1.1 \cdot 2.4} \approx 0.55$$

a note on Pearson correlation

$$w_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2}}$$

- some past approaches computed Pearson correlation **over common items**, i.e., all sums are over $I_u \cap I_v$
- this has a **counter-intuitive effect**
  - if users have just 1 common item, they are exactly similar
  - with 2 common items, they are either totally similar or totally negative similar
  - generally, users with few common items are more likely to be found similar
- to reverse it, several **elaborate methods** were proposed

- the formula given here nicely avoids such effects; **why**?

Pearson correlation and cosine similarity

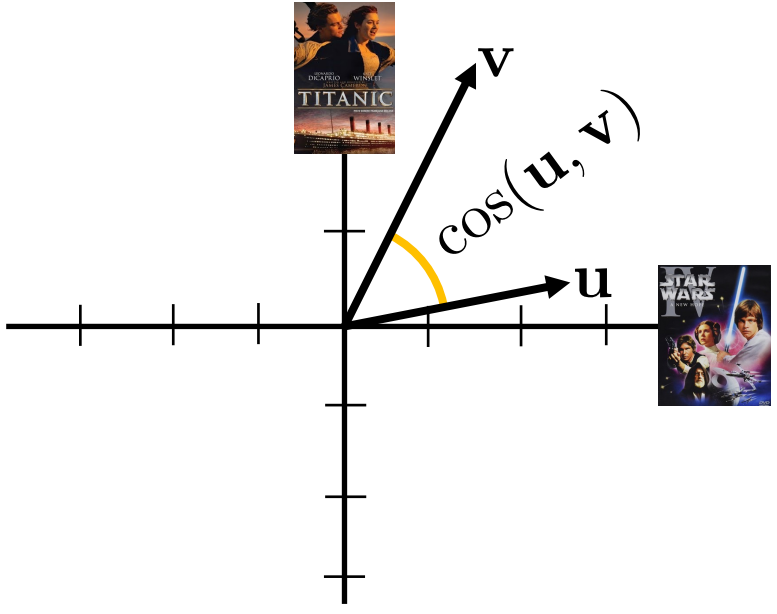
- they are related

- construct the mean-centered ratings **vector** $\mathbf{u}$ for user u
- so that each coordinate corresponds to an item and
  - has value $r_{ui} - \overline{r_u}$ if the user rated that item, i.e. the **deviation**
  - 0 otherwise

mean-centered cosine similarity

- the **cosine** of the angle between two (mean-centered) user vectors equals the **similarity** computed in the previous formula

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} = w_{uv}$$

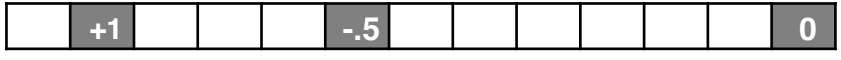


mean-centered cosine similarity

- if you look closely at the **Pearson correlation** example, you can see the **cosine similarity**

$$w_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2}}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} = w_{uv}$$



$$\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} = \sqrt{1^2 + (-0.5)^2 + 0^2} \approx 1.1 \qquad \|\mathbf{u}\|$$

$$\sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2} = \sqrt{2^2 + 1^2 + 1^2} \approx 2.4 \qquad \|\mathbf{v}\|$$

+2 -.5 0

$$\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v}) = 2 - 0.5 + 0 = 1.5 \qquad \langle \mathbf{u}, \mathbf{v} \rangle$$

last step to User-User CF

- so far, here is the formula for predicting ratings:

$$s(u,i) = \overline{r_u} + \frac{\sum_{v \in U} w_{uv}(r_{vi} - \overline{r_v})}{\sum_{v \in U} |w_{uv}|}$$

- considers *all* users to make predictions, including those who are not really similar (sim. around 0) to the target user
  - may introduce *noise* to the predictions

- standard approach is to limit the set of considered users, to a **neighborhood** of the target user containing highly similar users

user neighborhood

- here is the final User-User CF prediction formula:

$$s(u,i) = \overline{r_u} + \frac{\sum_{v \in N(u)} w_{uv}(r_{vi} - \overline{r_v})}{\sum_{v \in N(u)} |w_{uv}|}$$

- it computes the weighted average of rating deviations among the target user's **neighborhood** $N(u) \subseteq U$

- neighborhood contains users with the highest similarity to the target user
  - no reason to exclude a highly similar user in favor of a less similar one
  - in practice neighborhood contains only users that have rated the target item

neighborhood selection

- Q: Which users should go into the neighborhood of the target?
- A: application/domain dependent

- different options:
  - all users
  - limit neighborhood to top-k most similar users
  - limit neighborhood so that neighbors have similarity above a threshold
  - include (or not) users with high negative similarity (negative weights)
    - why? to learn what *not* to recommend

- best practice: use a neighborhood of 25-100 most similar users

User-User CF algorithm

To recommend an item to target user u

- for each user $v \neq u$ compute her/his **similarity** to the target user:

$$w_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_v} (r_{vi} - \overline{r_v})^2}}$$

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

$$s(u, i) = \overline{r_u} + \frac{\sum_{v \in N(u)} w_{uv} (r_{vi} - \overline{r_v})}{\sum_{v \in N(u)} |w_{uv}|}$$

- finally, recommend the item with the highest predicted rating

computational complexity of U-U CF

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

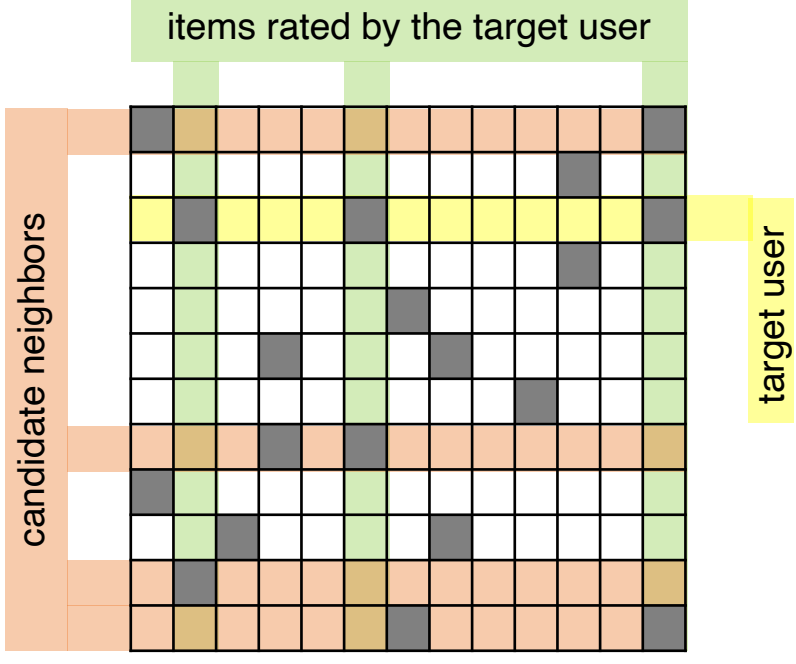
- computing similarity between two users takes $O(n)$
- computing all pairwise user similarities takes $O(nm^2)$
- creating a neighborhood of fixed size k takes $O(m)$
- predicting rating for a single item takes $O(k)$
- selecting the item with the highest predicted rating takes $O(n)$

- costs dominated by similarity computations

optimizations for U-U CF

Optimization 1: neighborhood creation

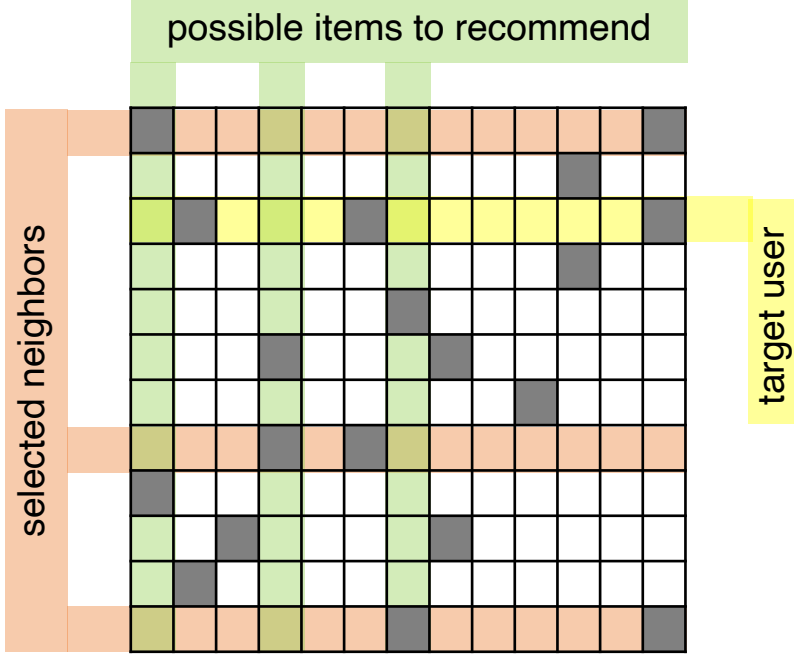
- consider the **target user**
- due to *sparsity*, has **rated few items**
- to compute similarity with other users, it suffices to **look among users that have rated at least one of those items**
  - why? if no common rated item exists, the similarity is zero



optimizations for U-U CF

Optimization 2: item recommendation

- now, assume the **neighbors** have been found
- only items that have been **rated by at least one neighbor** can be recommended
  - why? non-rated items cannot be recommended



problems with U-U CF

sparsity of ratings

- typically much more users than items
- relatively few ratings compared to the product of items and users
  - e.g., Netflix prize: 17K movies, 480K users, 100M ratings
  - density: 1.2% (considered relatively high compared to other domains)

- consequence:
  - two users might not have rated any common items
    - similarity cannot be computed

problems with U-U CF

computational performance

- most expensive part is computing *pairwise user similarities*
  - cost quadratic to number of users

- *precomputing* neighborhoods does not help
  - new ratings coming in might change similarities dramatically
  - e.g., if I only have a few ratings, an additional one carries a lot of weight

Acknowledgements

some ideas from:

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