Recommender Systems
lecture 2: neighbourhood-based models

Alexey Grishanov

Moscow Institute of Physics and Technology

Spring 2023

Recap: taxonomy

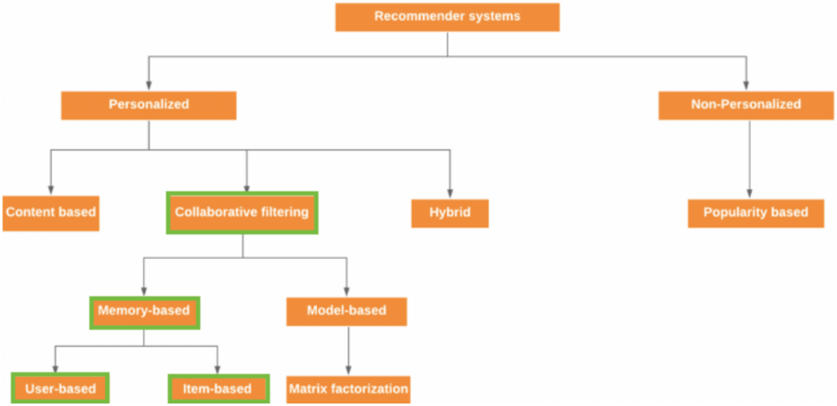
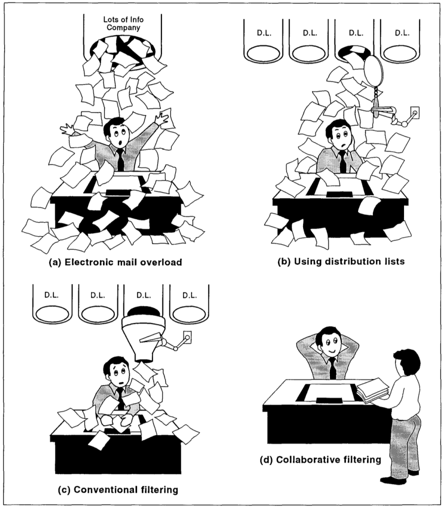


image credit: https://thingsolver.com/introduction-to-recommender-systems

Collaborative filtering



Lots of Info Company
D.L.
D.L.
D.L.
D.L.
(a) Electronic mail overload
(b) Using distribution lists
(c) Conventional filtering
(d) Collaborative filtering

Idea:

«Collaborative filters help people make choices based on the opinions of other people.»

[Paul Resnick et al. "Grouplens" 1994]

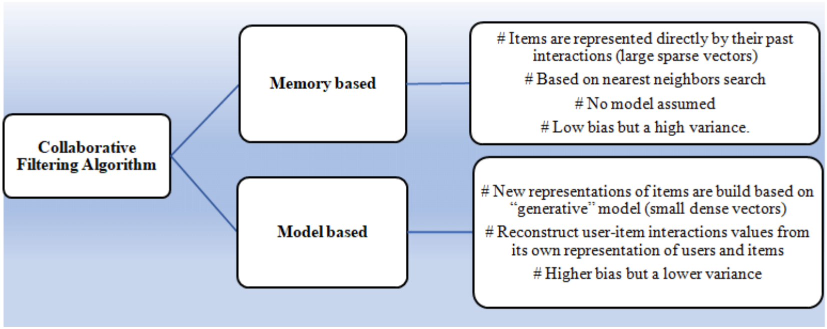
«Collaborative filtering simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read. Such reactions may be that a document was particularly interesting (or particularly uninteresting).

These reactions, more generally called annotations, can be accessed by others' filters.»

[David Goldberg et al. "Tapesty" 1992]

Memory-based vs Model-based

- memory-based (neighbourhood-based, heuristic-based) algorithms make recommendations based on the entire collection of previously rated items by the users.
- model-based algorithms use the collection of ratings to learn a model which is then used to make recommendations.



Collaborative Filtering Algorithm
Memory based
# Items are represented directly by their past interactions (large sparse vectors)
# Based on nearest neighbors search
# No model assumed
# Low bias but a high variance.
Model based
# New representations of items are build based on "generative" model (small dense vectors)
# Reconstruct user-item interactions values from its own representation of users and items
# Higher bias but a lower variance

image source: https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012057

User-based vs Item-based

a **User-based:** compute similarity between pairs of **users** based on thair past interactions with items

b **Item-based:** compute similarity between pairs of **items** based on thair past interactions with users. The similarity between items is typically more stable than the similarity between the users.

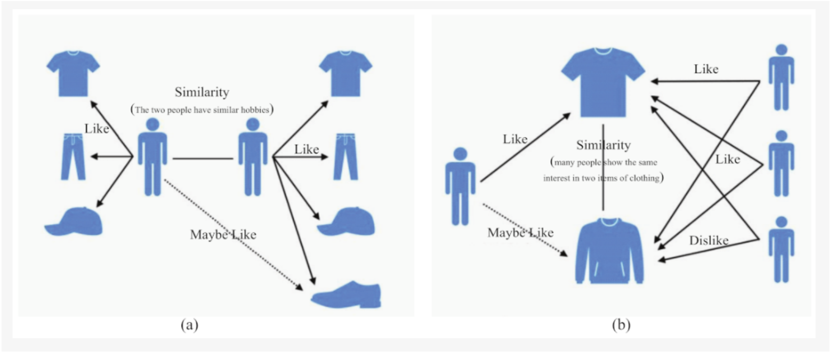

Similarity
(The two people have similar hobbies)
Like
Like
Maybe Like
(a)
Like
Similarity
(many people show the same interest in two items of clothing)
Like
Like
Maybe Like
Dislike
(b)

image source: https://www.mdpi.com/2076-3417/11/20/9554

Recap: problem statement

Given:

- $U = \{u_j|,\ j \in 1, \dots n_{users}\}$ — set of users
- $I = \{i_j|\ ,\ j \in 1, \dots n_{items}\}$ — set of items.
- $R = \|r_{ui}\|$ — relation matrix of shape $n_{users} \times n_{items}$,

$$r_{ui} \in \left[\begin{array}{c} \text{typically } \{0,1\}\text{— implicit feedback} \\ \text{typically } \{1,2,3,4,5\}\text{— explicit feedback} \end{array} \right.$$

Possible tasks:

Rating prediction

- Predict unknown r_{ui} — regression or (multi-class) classification

Top-k ranking

- Rank top-k recommendations for users — **item2user (course focus)**

Neighbourhood-based recommendations

$$r_{ui} = \left[\begin{array}{l} \underset{u' \in U_i}{\text{aggr}}\ r_{u'i} \text{ — user-based} \\ \underset{i' \in I_u}{\text{aggr}}\ r_{ui'} \text{ — item-based} \end{array} \right.$$

Next we will focus on **item-based** (if not specified explicitly). Examples of the aggregation functions are:

$$r_{ui} = \frac{1}{N} \sum_{i' \in I_u} r_{ui'}$$

$$r_{ui} = k \sum_{i' \in I_u} \text{sim}(i, i') \times r_{ui'}$$

$$r_{ui} = \overline{r_i} + k \sum_{i' \in I_u} \text{sim}(i, i') \times (r_{ui'} - \overline{r_i}),$$

k serves as a normalization factor and usually used as $k = \frac{1}{\sum_{i' \in I_u} |\text{sim}(i,i')|}$

Similarity weight computation

Possible variants for sim(i, j):

$$Cosine(i,j) = \frac{\sum\limits_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_{ij}} r_{ui}^2}\sqrt{\sum\limits_{u \in U_{ij}} r_{uj}^2}}$$

$$PC(i,j) = \frac{\sum\limits_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{ui} - \bar{r}_j)}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2 \sum\limits_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

For user-based (additional variants):

$$MSD(u,v) = \frac{|I_{uv}|}{\sum\limits_{i \in I_{uv}} (r_{ui} - r_{vi})^2}; \quad SRC(u,v) = \frac{\sum\limits_{i \in I_{uv}} (k_{ui} - \bar{k}_i)(k_{ui} - \bar{k}_j)}{\sqrt{\sum\limits_{i \in I_{uv}} (k_{ui} - \bar{k}_i)^2 \sum\limits_{i \in I_{uv}} (k_{uj} - \bar{k}_j)^2}},$$

where k_u is the average rank of items rated by u

Additional weighting (optional)

IDF, BM25, etc.

To reduce the influence of popular items in the similarityy measure and to give more weight to less popular items that a more relevant to a particular user, one may use common weighting from NLP. For example reweight cosine similarity with IDF:

$$\lambda_u = \log \frac{|I|}{|I_u|}; \quad Cosine(i,j) = \frac{\sum\limits_{u \in U_{ij}} \lambda_u r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_{ij}} \lambda_u r_{ui}^2} \sqrt{\sum\limits_{u \in U_{ij}} \lambda_u r_{uj}^2}}$$

Shrink coefficient

When weight is computed using only a few ratings, one may use shrinkage where a weak or biased estimator can be improved if it is “shrunk” toward a null-value (typical value for β is 100):

$$\text{sim}'(i,j) = \frac{|U_{ij}|}{|U_{ij}| + \beta} \text{sim}(i,j)$$

ItemKNN

$$r_{ui} = \frac{\sum\limits_{i' \in J} \text{sim}(i, i') \times r_{ui'}}{\sum\limits_{i' \in J} \text{sim}(i, i')},$$

where J — set of k most similar items.

Note

Consider $W \in \mathbb{R}^{|I| \times |I|}$ — item-item weight-matrix, $R_{u,\cdot}$ refer to row u; $W_{\cdot,i}$ — to column i.

r_{ui} might also be computed using $R_{u,\cdot}$ and $W_{\cdot,i}$

Exercise

The UserKNN algorithm also exists. Derive its formulas.

SLIM: Sparse LInear Methods

r_{ui} is calculated as a sparse aggregation of items that have been purchased/rated by u:

$$r_{ui} = R_{u,\cdot} \cdot W_{\cdot,i}$$

Optimization problem

Learning W for SLIM:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & ||R - RW||_F^2 + \frac{\beta}{2}||W||_F^2 + \lambda||W||_1 \\ \text{subject to} \quad & W \geq 0 \\ & \text{diag}(W) = 0, \end{aligned}$$

where $||W||_1 = \sum_{i=1}^{n}\sum_{j=1}^{n} |w_{ij}|$ is the entry-wise l_1-norm of W, and $||\cdot||_F$ is the matrix Frobenius norm.

EASE

r_{ui} is calculated similar to SLIM:

$$r_{ui} = R_{u,\cdot} \cdot W_{\cdot,i}$$

Optimization problem

Learning W for EASE:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & ||R - RW||_F^2 + \lambda ||W||_F \\ \text{subject to} \quad & \text{diag}(W) = 0 \end{aligned}$$

- Square loss allows for a closed-form solution (next slide). Training with other loss functions, however, might result in improved ranking accuracy.
- The constraint of a zero diagonal, $\text{diag}(W) = 0$, is crucial for ranking accuracy (in contrast to $W \geq 0$).

EASE: closed-form solution derivation

We start by forming the Lagrangian:

$$L = ||R - RW||_F^2 + \lambda ||W||_F + 2 \cdot \gamma^T \cdot \text{diag}(W),$$

where $\gamma = (\gamma_1, \dots, \gamma_{|I|})^T$ — Lagrangian multipliers.
Next, set its derivative to zero, which yields the estimate of the weight matrix:

$$W = \left(R^T R + \lambda E\right)^{-1} \left(R^T R - \text{diagMat}(\gamma)\right)$$

Defining (for sufficiently large λ)

$$P = \left(R^T R + \lambda E\right)^{-1}$$

this can be substituted into the previous equation:

$$\begin{aligned} W &= \left(R^T R + \lambda E\right)^{-1} \left(R^T R - \text{diagMat}(\gamma)\right) \\ &= P\left(P^{-1} - \lambda E - \text{diagMat}(\gamma)\right) \\ &= E - P\left(\lambda E + \text{diagMat}(\gamma)\right) \\ &= E - P \cdot \text{diagMat}(\tilde{\gamma}), \ \text{(where } \tilde{\gamma} = \lambda \vec{1} + \gamma) \end{aligned}$$

EASE: closed-form solution

$$0 = \mathrm{diag}(W) = \vec{1} - \mathrm{diag}(P) \odot \tilde{\gamma}$$

$$\Rightarrow \tilde{\gamma} = \vec{1} / \mathrm{diag}(P),$$

where / — elementwise division.

Substituting into previous slide gives

$$W = E - P \cdot \mathrm{diagMat}(\vec{1}/\mathrm{diag}(P))$$

EASE closed-form solution

$$P = \left(R^T R + \lambda E\right)^{-1}$$

$$W_{ij} = \begin{cases} 0, \text{if } i = j \\ -\frac{P_{ij}}{P_{jj}}, \text{otherwise} \end{cases} .$$

Literature

1. *F. Ricci, L. Rokach, B. Shapira.* (2011). Recommender Systems Handbook.
2. [David Goldberg et al. "Tapesty" 1992]
   D. Goldberg, D. Nichols, B. Oki, D. Terry (1992). Using Collaborative Filtering to Weave an Information Tapestry. Commun. ACM 35(12): 61-70 (1992)
3. [Paul Resnick et al. "Grouplens" 1994]
   P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Working Paper Series 165, MIT Center for Coordination Science.
4. *X. Ning and G. Karypis* (2011). SLIM: Sparse Linear Methods for Top-N Recommender Systems, ICDM
5. *H. Steck* (2019). Embarrassingly Shallow Autoencoders for Sparse Data, The World Wide Web Conference