Recommender Systems

Lecture 4

Previous lecture

Lifecycle of a recsys experiment

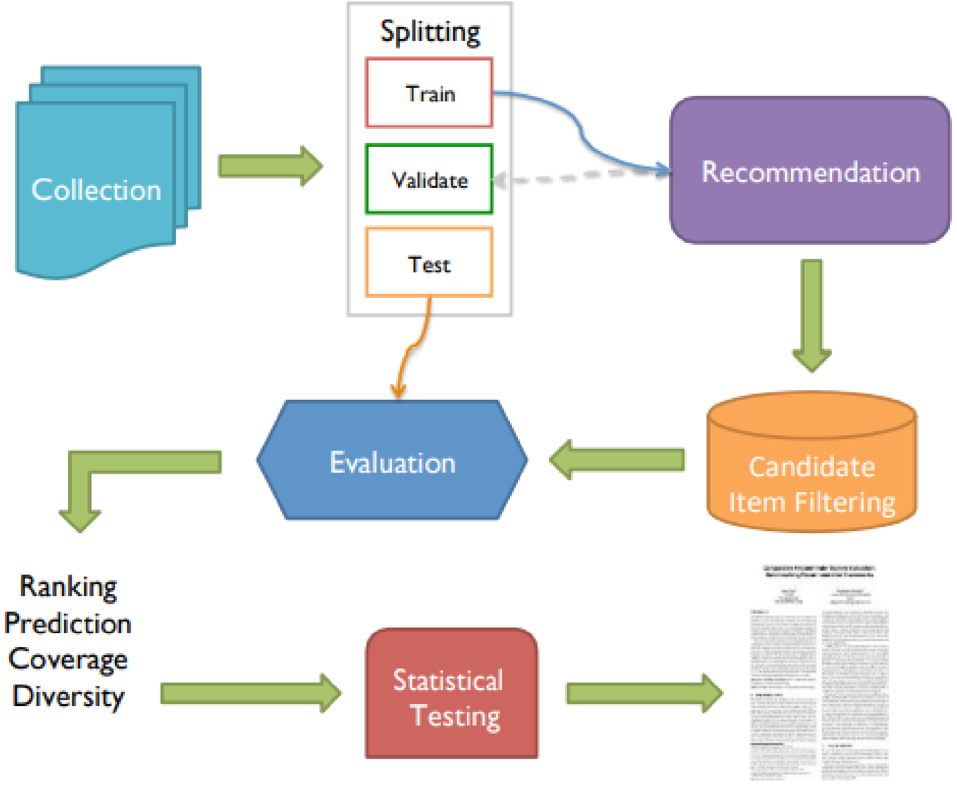
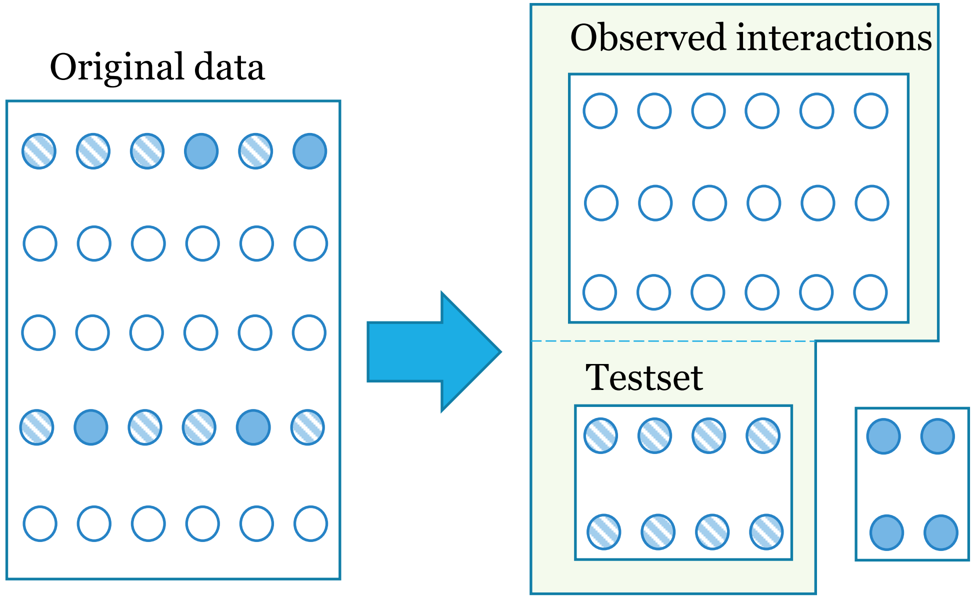


Image source: Bellogín, Alejandro, and Alan Said. "*Improving accountability in recommender systems research through reproducibility*." User Modeling and User-Adapted Interaction (2021): 1-37.

Data splitting



Splitting options:

- entry- or user-wise
- warm start
    - strong/weak generalization

Holdout sampling

Strategies

- Random
- Rating-based (e.g., top-rated)
- Temporal (e.g., most recent)

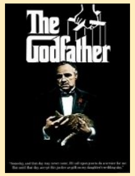
Sample size

- Fixed number of items (e.g., 1)
- Fixed percentage of items

Previous lecture

User

known user preferences



holdout

recommendations

Algorithm:

$$\mathrm{HR} = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test}\\\text{users}}} \text{hit}$$

$$\text{hit} = \begin{cases} 1 & \text{if holdout item is in recommended items,} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathrm{MRR} = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test}\\\text{users}}} \frac{1}{\text{hit rank}}$$

hit rank = position of the item in the recommendations list

Typically computed: metric@n, where n = #recommended items, e.g. Recall@n, MRR@n, etc.

Task

Consider top-2 recommender for 10 users from 20-items (2 items per user).

What's better:

1) Correctly recommend 2 items to each of 5 users?

or

2) 1 item to each of 10 users?

Why?

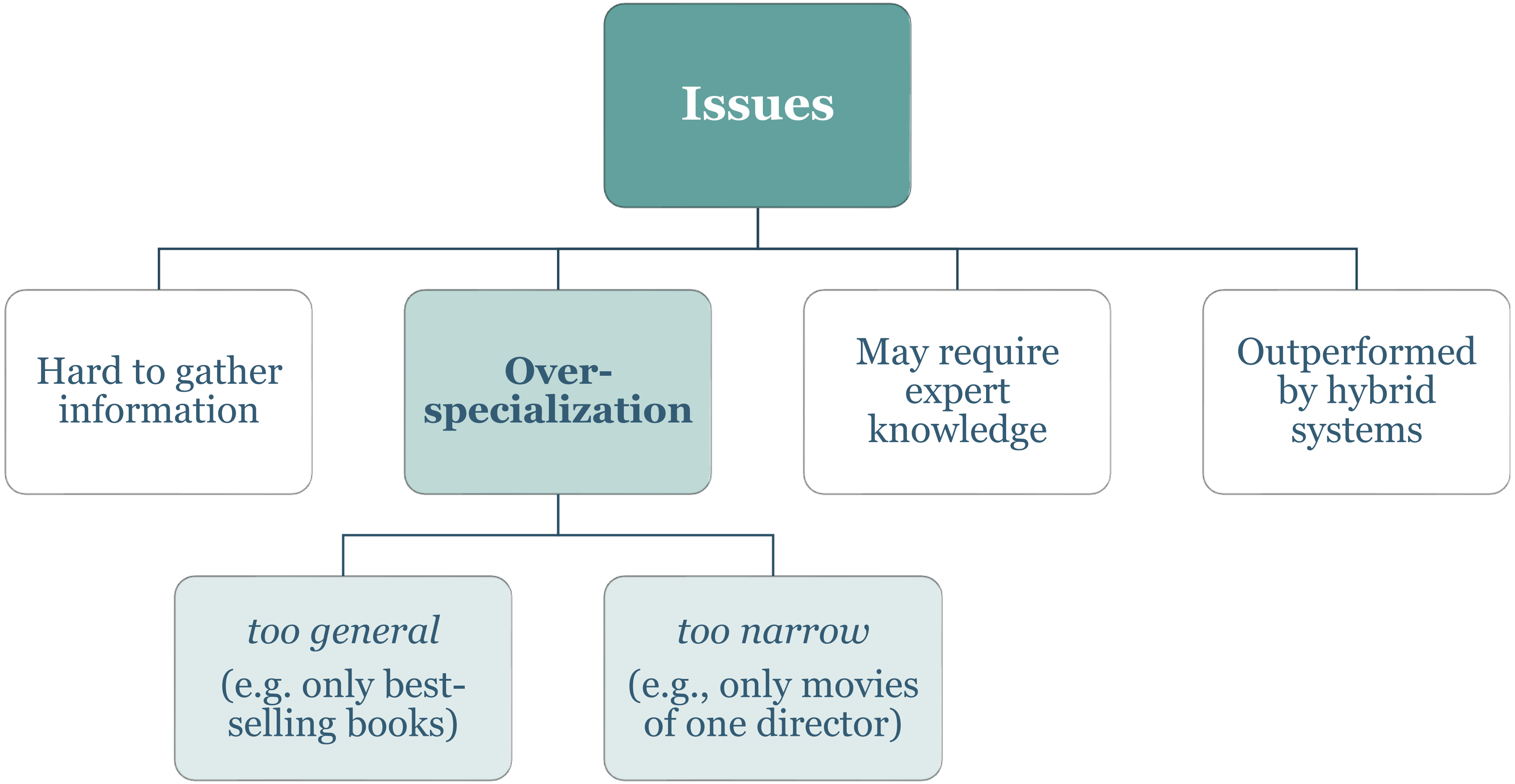
Use standard definition of precision and recall.

Today's lecture

Collaborative filtering

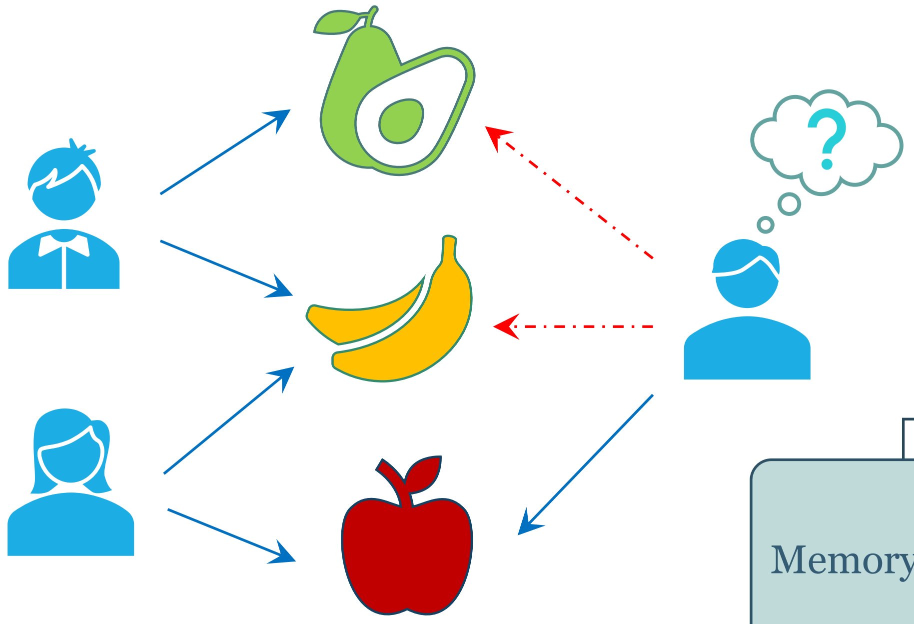
- memory-based approach
  - frequent pattern mining
  - nearest neighbors models

Previously: content-based approach



Collaborative Filtering: “wisdom of crowds”

- **Non-personalized**
  - popularity models
  - baseline predictors



- **Personalized**
  - Memory-based
    - item-based
    - user-based
  - Model-based
    - latent factors models

General workflow

Goal: predict user preferences based on prior user feedback and collective user behavior.

collect data

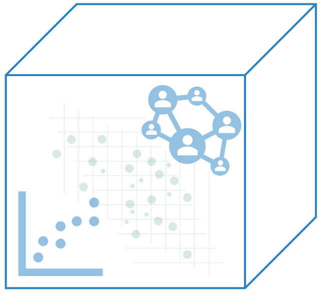
user-movie matrix A of size $M \times N$

a_{ij} is a rating of i^{th}user for j^{th} movie

? - missing (unknown) values

build model

f_U: User × Item → Relevance



generate recommendations

unknown user

“Customers who like … also like …”

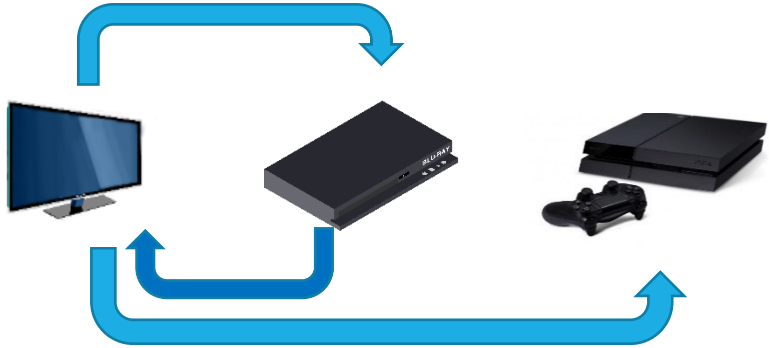
How do we implement that logic?

Pure item-to-item (I2I)

Typical transactions log:

user id	item id	transact.
0	575	view
0	1881	view
0	846	basket
1	1878	purchase
1	576	view
…	…	…

Count co-occurrence of items:



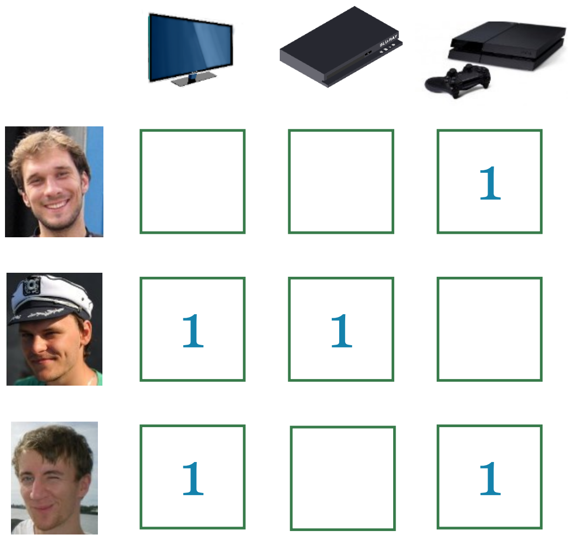
$$\mathrm{score}_{\mathrm{I2I}}(u,i) = \sum_{\substack{j \in I_u \\ j \neq i}} \mathrm{pairCount}(i,j)$$

Item-to-item is a strong baseline on very “sparse” datasets.

Simplest item-to-item approach

Convenient representation of logs– sparse matrix

user id	item id	transact.
0	575	view
0	1881	view
0	846	basket
1	1878	purchase
1	576	view



- Can be efficiently stored in CSR or CSC formats.
- Also enables efficient computations (especially useful for experiments).

Computing I2I scores

- How to compute item-to-item co-occurrence matrix in symmetric case?
- How to compute similarity scores in that case?

A – user-item interactions $M \times N$

$$C = A^T A - \mathrm{diag}(\mathrm{diag}(A^T A))$$

$$p = [0\ 1 \ldots 1 \ldots 0 \ldots 1 \ldots 0\ 1]^T \in \mathbb{B}^N$$

$$z = Cp$$

Computing I2I scores

$$C = A^\top A - \operatorname{diag}\left(\operatorname{diag}(A^\top A)\right)$$

If p is a vector of known user preferences,
then the vector of predicted relevance scores is:

$$r = Cp$$

Recommendations:

$$\operatorname{toprec}(n) \coloneqq \arg\max_{j}^{n} r_j$$

Complexity analysis

$C = A^T A$ $\quad M \times N$

$O(MN^2)$ - compute

$O(N^2)$ - storage

$O(1)$

$O(\log N)$

$O(MN)$

$O(N)$

$O(MN \log N)$

$O(N \log N)$

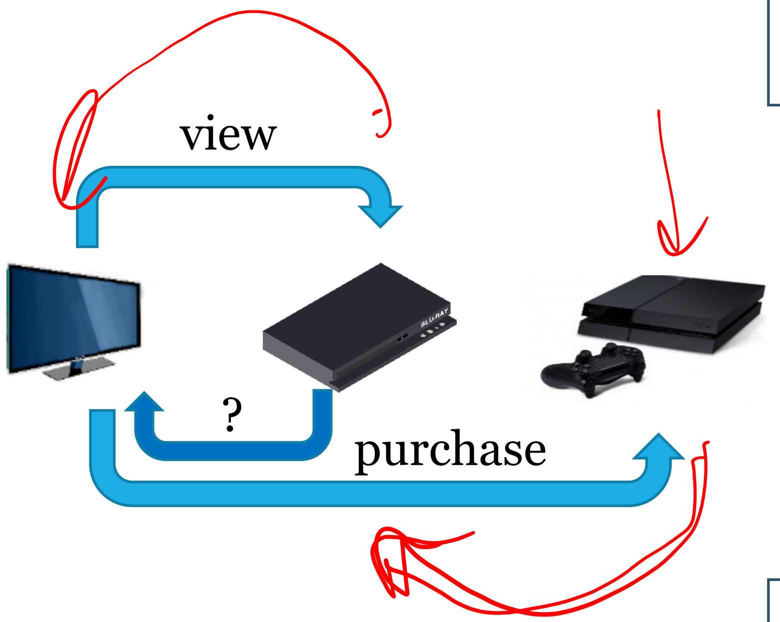
Item-to-item issues

- somewhat obvious recommendations
  - high influence of popular items

- i2i matrix can also become dense if there are too many interactions per user

$$c_{ij} \sim a_{:i}^T a_{:j}$$

Item-to-item variants



- Pair count strategies:
  - symmetric / asymmetric
  - next-only / list-wise
  - by type / category
- Value processing:
  - thresholding
  - weighting
  - smoothing

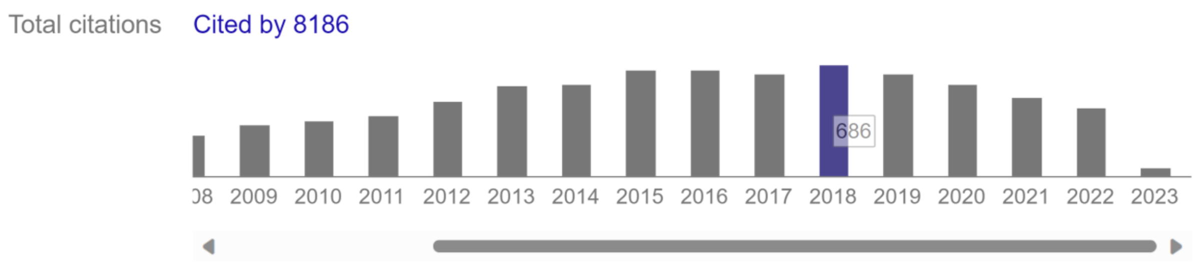
$\log(c_{ij}+1)$

$(t-t_0)^{-1}$

Case study: Amazon item-to-item

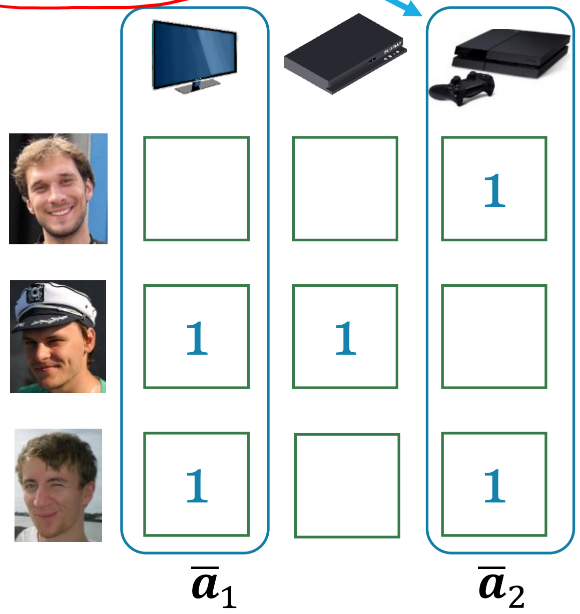
```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003.



Iterative algorithm

Computes similarity of items based on user purchases.



$$\mathrm{sim}(I_1, I_2) = \cos(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{a}}_2) = \frac{(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{a}}_2)}{\|\overline{\boldsymbol{a}}_1\|\|\overline{\boldsymbol{a}}_2\|}$$

$\overline{\boldsymbol{a}}_k$ - "one-hot" representation of item k

Scalability trick: incremental updates in binary case

$$\text{sim}(i,j) = \frac{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{a}}_j}{\|\bar{\boldsymbol{a}}_i\| \cdot \|\bar{\boldsymbol{a}}_j\|} = \frac{\text{pairCount}(i,j)}{\sqrt{\text{itemCount}(i)} \cdot \sqrt{\text{itemCount}(j)}}, \qquad j \neq i$$

$$\|\bar{\boldsymbol{a}}_i\|^2 = \sum_u a_{ui}^2 = \sum_u a_{ui} = \text{itemCount}(i) \qquad \bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{a}}_j = \sum_u a_{ui}\, a_{uj} = \text{pairCount}(i,j)$$

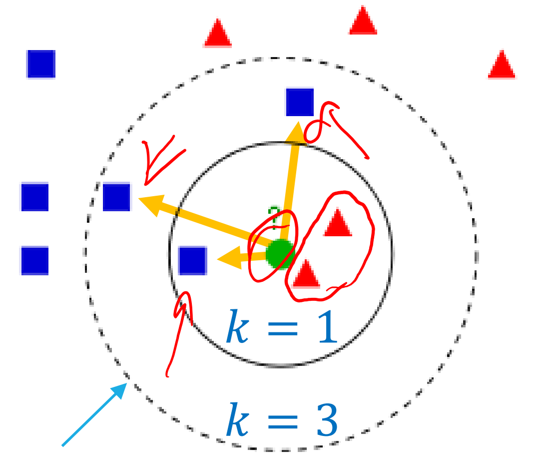
After observing ΔA new interactions s.t. $A' = A + \Delta A$, the updated similarity is:

$$\text{sim}'(i,j) = \frac{\text{pairCount}(i,j) + \sum_u [\Delta A]_{ui}\, [\Delta A]_{uj}}{\sqrt{\text{itemCount}(i) + \sum_u [\Delta A]_{ui}} \cdot \sqrt{\text{itemCount}(j) + \sum_u [\Delta A]_{uj}}}$$

Source: Huang, Yanxiang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. "Tencentrec: Real-time stream recommendation in practice." *In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 227-238. 2015.

Nearest neighbors models

kNN-based approach



neighborhood $\mathcal{N}_i(u)$

- user u
- neighbors of user u, who rated item i
- other users, who have not rated item i

User-based approach

- aggregated opinion of like-minded users:

$$\mathrm{score}_{\mathrm{uKNN}}(u,i) = \operatorname*{agg}_{v\in\mathcal{N}_i(u)} a_{vi}$$

Item-based approach:

- aggregated rating of similarly rated item

$$\mathrm{score}_{\mathrm{iKNN}}(u,i) = \operatorname*{agg}_{j\in\mathcal{N}_u(i)} a_{uj}$$

Simple user-based kNN

$$\text{score}_{\text{uKNN}}(u,i) = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in \mathcal{N}_i(u)} a_{vj}$$

Potential issues:

- users may have very different interests
- neighborhood size is unlimited

Improved user-based kNN

$$\text{score}_{\text{uKNN}}(u,i) = \frac{1}{z} \sum_{v \in \mathcal{N}_i(u)} \text{sim}(u,v) \cdot a_{vi}$$

$$\mathcal{N}_i(u) = U_i \setminus \{u\}, \qquad z = \sum_{v \in \mathcal{N}_i(u)} |\text{sim}(u,v)|$$

$\sum_i w_i a_i$

$w_i \in [0,1]$

Potential issues:

- ~~other users may have very different interests~~
- large neighborhood size

Dealing with large neighborhood size

- Storing similarities or on-the-fly computations?

- Aggressive subsampling

randomly sample from $N_u(i)$

recency-based $N_i(u)$

- Approximate nearest neighbors
  - e.g., NMSLib, Faiss, Annoy

- Dimensionality reduction

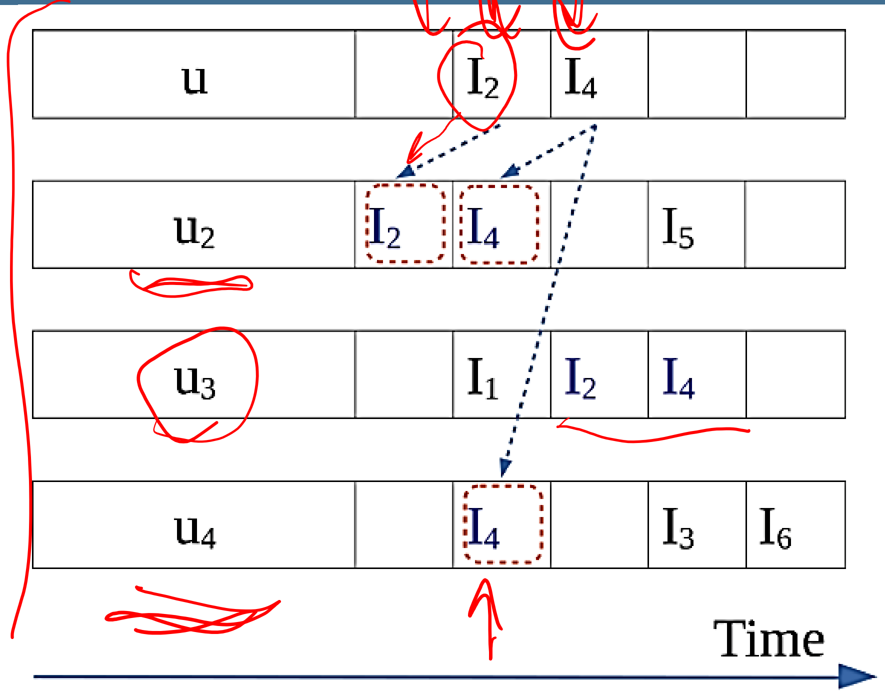
Reducing neighborhood

- from N total entities sample $n \ll N$
- select top-k most similar among n samples, $k \ll n$

Possible sampling strategies (must be fast):

- randomly
- most recent only
- most ratings in common (turns into fast MIPS problem)

Example: local time-aware sampling



Sampling strategy:
- select users that have items in common with a target user u
  - each item of a neighbour-user must precede the corresponding item in the target user profile
  - filter out neighbours with too few items in common

Additional weighting:
- users with no recent ratings → lower weights
- active neighbour-user but old rating on a target item → lower weights

	recent user ($t_0 \approx t_{u'1}$)	old user ($t_0 \gg t_{u'1}$)
recent item ($t_{u'1} \approx t_{u'i}$)	≈ 0	$t_0 - t_{u'1}$
old item ($t_{u'1} \gg t_{u'i}$)	$t_{u'1} - t_{u'i}$	$t_0 - t_{u'1}$

Images source and more details: Anelli, V. W., Di Noia, T., Di Sciascio, E., Ragone, A., & Trotta, J. (2019). Local popularity and time in top-N recommendation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11437 LNCS*, 861–868

Centered kNN

- baseline estimators contain most of useful signal
- every user may have individual "rating scale"

$$\text{score}_{\text{uKNN}}(u,i) = \bar{a}_u + \frac{1}{z}\sum_{v \in \mathcal{N}_i(u)} \text{sim}(u,v)\cdot(a_{vi} - \bar{a}_v)$$

$\bar{a}_u$ - average rating of user u

item-based kNN:

$$\text{score}_{\text{iKNN}}(u,i) = \bar{a}_i + \frac{1}{z}\sum_{j \in \mathcal{N}_u(i)} \text{sim}(i,j)(a_{uj} - \bar{a}_j)$$

Centered kNN

- baseline estimators contain most of useful signal
- every user may have individual "rating scale"

user-based kNN:

$$\mathrm{score}_{\mathrm{uKNN}}(u,i) = \bar{a}_u + \frac{1}{z}\sum_{v\in\mathcal{N}_i(u)} \mathrm{sim}(u,v)\cdot(a_{vi} - \bar{a}_v)$$

$\bar{a}_u$ - average rating of user u

item-based kNN:

$$\mathrm{score}_{\mathrm{iKNN}}(u,i) = \bar{a}_i + \frac{1}{z}\sum_{j\in\mathcal{N}_u(i)} \mathrm{sim}(i,j)\cdot(a_{uj} - \bar{a}_j)$$

$\bar{a}_i$ - average rating of user i

Similarity measures

- Cosine Similarity
- Pearson Correlation
- Adjusted Cosine Similarity
- Jaccard Index
- Weighted Jaccard Index
- Asymmetric Similarities
- ...
- Spearman's Rank Correlation
- Kendall Tau

insensitive to ranking of "bad" items vs "good " items

Baseline-adjusted similarity

- **Pearson correlation** (adopted for CF):

$$\text{score}_{\text{Pearson}}(u,v) = \frac{\sum_{i \in I_u \cap I_v} (a_{ui} - \bar{a}_u)(a_{vi} - \bar{a}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (a_{ui} - \bar{a}_u)^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (a_{vi} - \bar{a}_v)^2}}$$

$\bar{a}_u$ - average rating of user u

- **Adjusted Cosine Similarity**:

$$\text{score}_{\text{AC}}(u,v) = \frac{\sum_{i \in I_u \cap I_v} (a_{ui} - \bar{a}_i)(a_{vi} - \bar{a}_i)}{\sqrt{\sum_{i \in I_u \cap I_v} (a_{ui} - \bar{a}_i)^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (a_{vi} - \bar{a}_i)^2}}$$

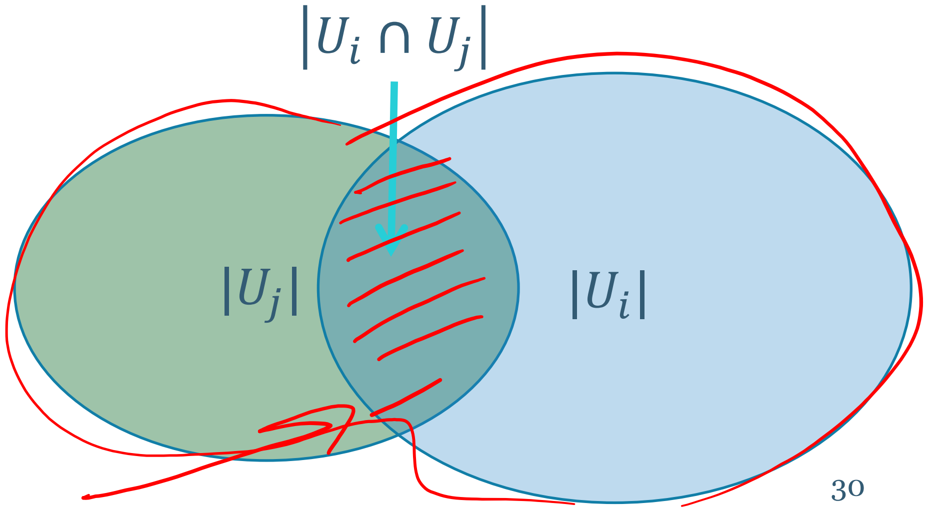
$\bar{a}_i$ - average rating of item i

Jaccard Index

Item-based similarity:

$$\mathrm{sim}_{\mathrm{JI}}(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

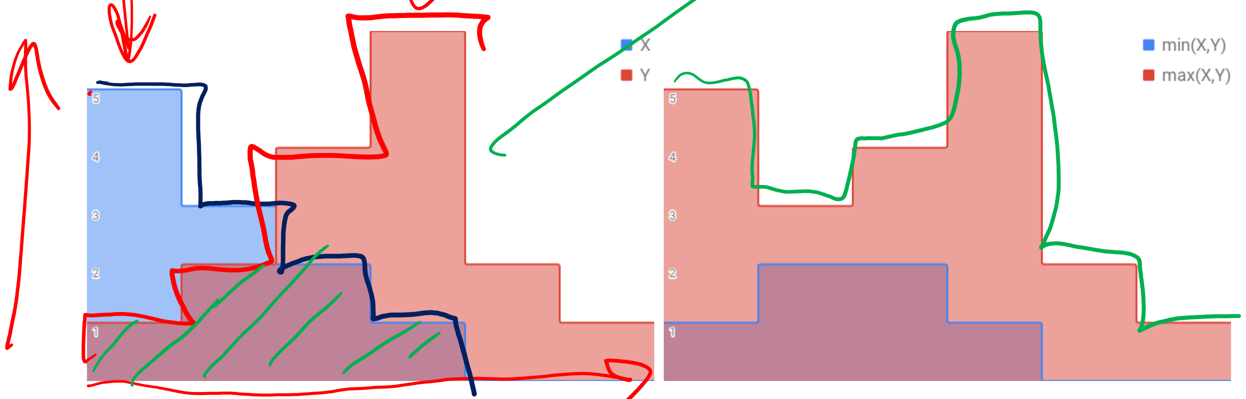
$$|U_i \cup U_j| = |U_i| + |U_j| - |U_i \cap U_j|$$

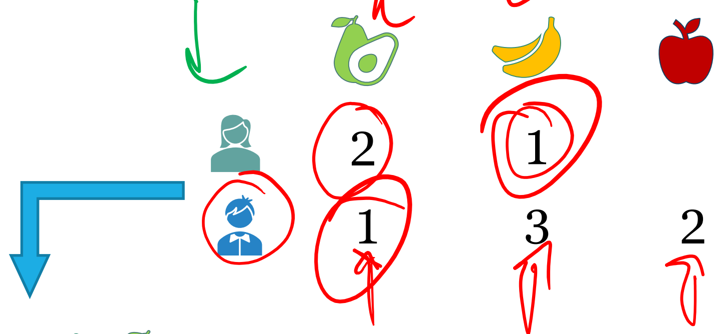


Weighted Jaccard Index

- Jaccard Index only operates on sets
- often some values are associated with interactions (e.g., ratings, frequencies)

$$\mathrm{sim}_{\mathrm{WJI}}(u,v) = \frac{\sum_{i=1}^{N} \min\{w_i(a_u), w_i(a_v)\}}{\sum_{i=1}^{N} \max\{w_i(a_u), w_i(a_v)\}}, \qquad w_i(a_u) = f(a_{ui})$$





- intersection = []
- union = []
- $score_{WJI}($, $) = \frac{2}{7}$

Images from: https://moultano.wordpress.com/2018/11/08/minhashing-3kbzhsxyg4467-6/

kNN in matrix form

Element-wise weighting for user-based KNN:

$$r_{ui} = \frac{1}{z_{ui}} \cdot \sum_{v \in \mathcal{N}_i(u)} \mathrm{sim}(u, v) \cdot a_{vi}\,, \qquad \mathcal{N}_i(u) = U_i \setminus \{u\}$$

$M \times M$

K - zero-diagon. similarity matrix

$A \in M_{\times N}$

we impute A with 0's

elementwise division

$$R = KA \oslash KB$$

$$z_{ui} = \sum_{v \in \mathcal{N}_i(u)} |\mathrm{sim}(u,v)|$$

$$[B]_{ij} = \begin{cases} 1 & a_{ij} \text{ known} \\ 0 & \text{otherwise} \end{cases}$$

kNN in matrix form

Row-wise weighting for user-based KNN:

$$r_{ui} = \frac{1}{z_{ui}} \cdot \sum_{v \in \mathcal{N}_i(u)} \text{sim}(u, v) \cdot a_{vi}\,, \qquad \mathcal{N}_i(u) = U \backslash \{u\}$$

i.e., "explicit" 0's

$$z_{ui} = \sum_{v \in U \backslash \{u\}} |\text{sim}(u,v)| \qquad [D]_{uu} = z_{ui}$$

$$R = D^{-1} K A$$

kNN weighting schemes

K – user similarity, $k_{ii} = 0, k_{ij} \geq 0, i \neq j$; S –item similarity matrix, $s_{ii} = 0, \; s_{ij} \geq 0, i \neq j$.

element-wise weighting:

- User-based:

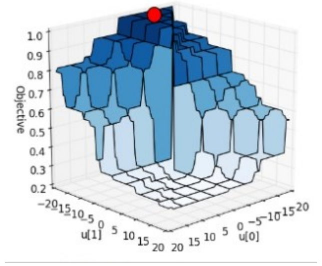
$$R = KA \oslash (KB)$$

$$b_{ui} = \begin{cases} 1, & \text{if } a_{ui} \text{ is known} \\ 0 & \text{otherwise} \end{cases}$$

- Item-based:

$$R = AS^{\top} \oslash (BS^{\top})$$

- filters known ratings only
- better for rating prediction



row-wise / no weighting:

- User-based:

Sim(i,j)

$$R = D_K^{-1} KA$$

$$D_K = \mathrm{diag}(K\boldsymbol{e}) \;\text{ or }\; D_K = I$$

$$\boldsymbol{e} = [1, 1 \ldots, 1]^{\top}$$

- Item-based

$$R = AS^{\top} D_S^{-1}$$

$$D_S = \mathrm{diag}(S\boldsymbol{e}) \;\text{ or }\; D_S = I$$

- assumes 0-imputation of unknowns
- better for top-n recommendations

Let’s implement simple KNN models

User-based kNN for top-n recommendations

- Row-wise weighting:

$$R = D^{-1}KA$$

$$r_{ui} = w_u \cdot \sum_{v \in \mathcal{N}(u)} \text{sim}(u,v) \cdot a_{vi}$$

- Alternative (column-wise) weighting:

$$R = KD^{-1}A$$

$$r_{ui} = \sum_{v \in \mathcal{N}(u)} \text{sim}(u,v) \cdot w_v \cdot a_{vi}$$

- Is it different from the unweighted case?

kNN with asymmetric similarity

kNN similarity (e.g., item-based):

row-wise weighted symmetric → unweighted asymmetric

$$S_{\text{asym}} = D^{-\alpha} S$$

$$R = A S_{\text{asym}}^{\top}$$

$S = S^{\top}$

$S_{asym} \neq S_{asym}^{\top}$

Example: cosine similarity, assuming $d_{ii} = \|\bar{\boldsymbol{a}}_i\|$:

$$\text{sim}(i, j) = \left[S_{\text{asym}}\right]_{ij} = \frac{\bar{\boldsymbol{a}}_i^{\top} \bar{\boldsymbol{a}}_j}{\|\bar{\boldsymbol{a}}_i\|^{1+\alpha} \cdot \|\bar{\boldsymbol{a}}_j\|}$$

For binary data, $\alpha = -1$ gives a simple conditional probability $p(i|j)$

Popularity effect in asymmetric similarity

$$\mathrm{sim}(i,j) = [D^{-\alpha}S]_{ij} = \frac{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{a}}_j}{\|\bar{\boldsymbol{a}}_i\|^{1+\alpha} \cdot \|\bar{\boldsymbol{a}}_j\|}$$

What do we recommend?

$$\underset{i}{\text{popular item}} \xrightarrow{\mathrm{sim}} \underset{j}{\text{unpopular}} \quad \text{vs.} \quad \underset{i}{\text{unpopular}} \xrightarrow{\mathrm{sim}} \underset{j}{\text{popular item}}$$

Popularity effect in asymmetric similarity

$$\text{popular item} \xrightarrow{\text{sim}} \text{unpopular} \qquad \text{vs.} \qquad \text{unpopular} \xrightarrow{\text{sim}} \text{popular item}$$

$\alpha < 0$

- Popular products → too trivial recommendations.
- Easy to guess but low value for users + low diversity.

$\alpha > 0$

- Recommending niche products increases diversity.
- For users with generic tastes may not fit well.

- <u>Observation</u>: popular items are not very descriptive of users interests.
- Suggest a normalization that would improve item-KNN recommendations.

New scheme – emphasizing contribution of specific user tastes:

$$S_{\text{asym}} = SD^{-\beta}, \qquad \text{sim}(i,j) = \frac{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{a}}_j}{\|\bar{\boldsymbol{a}}_i\| \cdot \|\bar{\boldsymbol{a}}_j\|^{1+\beta}}$$

Let’s implement new weighting

Useful reading on similarity in kNN

Herlocker, Jon, Joseph A. Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." Information retrieval 5 (2002): 287-310.

Deshpande, Mukund, and George Karypis. "Item-based top-n recommendation algorithms." ACM Transactions on Information Systems (TOIS) 22, no. 1 (2004): 143-177.

Koenigstein, Noam, and Yehuda Koren. "Towards scalable and accurate item-oriented recommendations." In Proceedings of the 7th ACM conference on Recommender systems, pp. 419-422. 2013.

Naumov, Sergey, Marina Ananyeva, Oleg Lashinin, Sergey Kolesnikov, and Dmitry I. Ignatov. "Time-Dependent Next-Basket Recommendations." In European Conference on Information Retrieval, pp. 502-511. Cham: Springer Nature Switzerland, 2023.

Summary

Key advantages:

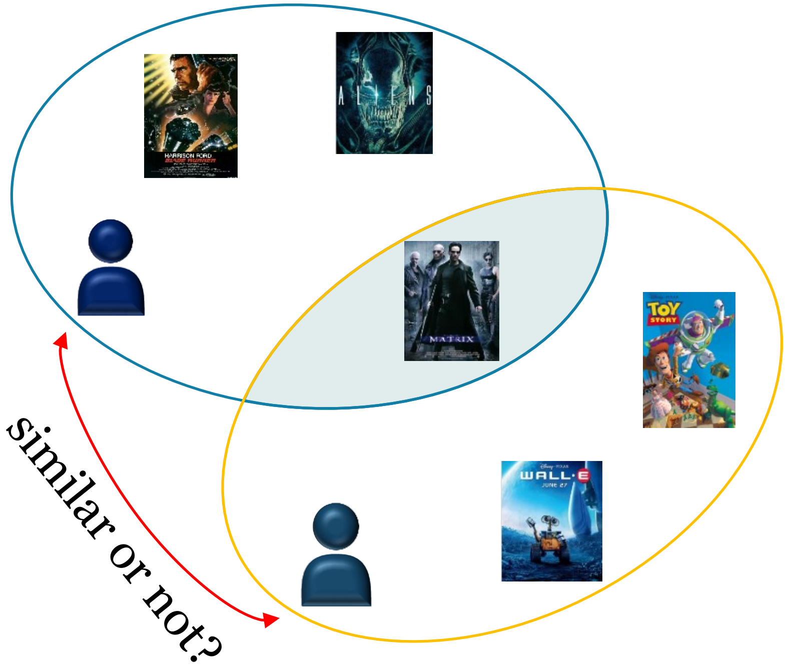
- easy to implement
- intuitive explanations
- good baseline
- suitable for very sparse datasets

Scalability:

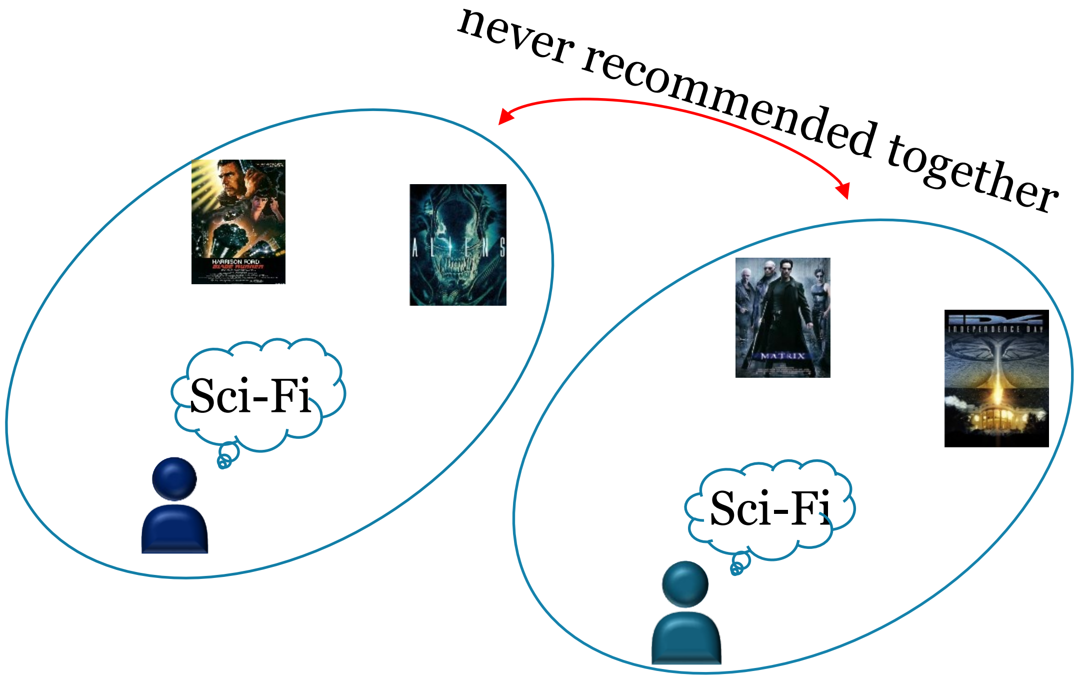
- computional complexity in the worst case: $O(MN^2)$ or $O(NM^2)$
- storage complexity in the worst case: $O(M^2)$ or $O(N^2)$
- due to sparsity real complexity can be significantly decreased
- could additionally limit the number of neighbors
- sometimes incremental updates are possible

Limited coverage problems

Unreliable correlations



Weak generalization



When to use user-based vs item-based?

- Depends on # users and # items
- Depends on dynamics

- Item-based recommendations are easier to explain.
- User-based recommendations increase serendipity.