Learning from User-generated Data
Summer Term 2022

Learning from Explicit User Feedback: Memory-based Collaborative Filtering

Markus Schedl
markus.schedl@jku.at

http://www.cp.jku.at

JOHANNES KEPLER UNIVERSITY LINZ

INSTITUTE OF COMPUTATIONAL PERCEPTION

MULTIMEDIA MINING AND SEARCH GROUP

Recommender systems are omnipresent

For products:

amazon

For fashion:

STITCH FIX

For jokes:

Jester 5.0

Jokes for *your* sense of humor

For travel:

trivago

For movies/series:

NETFLIX

For books:

WHAT SHOULD I READ NEXT?

For music:

DEEZER

Spotify

last.fm

For users and user-generated content:

facebook

Pinterest

twitter

Retrieving, Browsing, Recommending

Definitions and Implications:

Information **retrieval** (IR) is the area of study concerned with **searching** for documents, for information within documents, and for metadata about documents [...] *[Wikipedia]*

Implication: user has a **specific information need**, typically represented as a query; system tries to find the most relevant items wrt. query

Retrieving, Browsing, Recommending

Definitions and Implications:

Browsing: interactive task in which the user is more interested in **exploring** the document collection than in retrieving documents which satisfy a specific information need *[Baeza-Yates and Ribeiro-Neto, 1999]*

Implication: user has an **undirected information need** (hard to express as a query); system should provide a good interface for discovering items

Retrieving, Browsing, Recommending

Definitions and Implications:

Recommender: form of a specific type of information filtering system technique that attempts to **recommend** information items [...] or social elements [...] that are likely to be of interest **to the user**. *[Wikipedia]*

Implication: the system searches for potentially relevant items based on user's actions or preferences that form a **user profile** (can, e.g., be observed during retrieval or browsing)

Retrieving, Browsing, Recommending

Required Involvement of the User:

Retrieval: user has a specific information need, typically represented as a query; system tries to find the most relevant items wrt. query

user needs to be active

Browsing: user has an undirected information need; system should provide a good interface for discovering items

Recommending: the system searches for potentially relevant items based on user's actions or preferences (e.g., observed during retrieval or browsing)

user can be (almost) passive

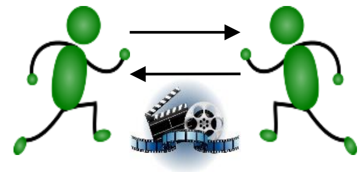
Characteristics of Recommender Systems: RS...

- provide a **personalized** view on the available collection (personalized filtering): every user sees a different list depending on her taste
- need to know something about the user and maintain some sort of **user model** or **user profile** - generated either **implicitly** (e.g., by observing behavior or preferences or keeping a list of purchased items) or **explicitly** (e.g., by asking to rate items)
- **find** those items that are most likely of interest to the user and (possibly) **rank** them according to prospective interest (utility)
- to this end often incorporate **additional knowledge** such as the user's characteristics (e.g., demographic filtering) or context (e.g., context-aware recommendation)
- **need to scale** well: typically deal with very large databases (millions of items) and communities (up to millions of users)

Main Flavors of Recommendation Systems

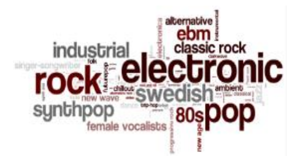
Collaborative filtering:

Recommend to target user items that other *similar users* liked in the past



Content-based filtering:

Recommend to target user *content similar* to what he or she liked in the past



Context-aware RS:

Recommend to target user items that he, she, or other users liked in a given *context or situation*

Hybrid RS: Any *combination* of the above

Multimedia Mining

User Modeling

Why People Use Recommender Systems

Herlocker et al. have identified 10 reasons why people use recommender systems (user tasks):

1. **Annotation in Context**. Predictions of potential items related to current context are displayed (historical background: recommending discussion postings, links while Web browsing)
2. **Find Good Items**. “Core recommendation task”; Interfaces suggests top-ranked list and shows predicted rating values (optional)
3. **Find All Good Items**. In contrast to 2., in this case, systems must not overlook a single relevant item, i.e., keep false negative rate low (scenario: law case precedents, patents)
4. **Recommend Sequence**. Whole sequence should be pleasant or meaningful, cf. automatic playlist generation in personalized radio streams or reading recommendations for scientific literature to learn about a field

Why People Use Recommender Systems

5. **Just Browsing**. Browsing (commercial systems) can be pleasant even without the goal of purchasing; pure entertainment
6. **Find Credible Recommender**. “Playing around” with the system to see whether recommendations match taste; checking system for *bias* by using multiple profiles
7. **Improve Profile**. Rating items to improve user profile to improve recommendations
8. **Express Self**. Contributing ratings and reviews is satisfactory for some people (sometimes even more important than getting recommendations)
9. **Help Others**. To let the community benefit from own contributions
10. **Influence Others**. Manipulate ratings to make others view or buy (or avoid) specific items, e.g., *hacking* and *vandalism*

Collaborative Filtering (CF)

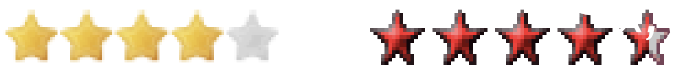
- Most widely used approach for recommendation
- Applicable to many domains (no explicit domain knowledge necessary)
- Main underlying assumption: users that had similar taste in the past, will have similar taste in the future
- Collect user feedback given as explicit (or implicit) ratings of items
- Use ratings of all or some users collectively to determine whether to recommend a new item to a user (e.g., by predicting the user’s rating for an item)

Gathering Explicit User Data: Rating Methods

Different types (scales) of rating data:

- "Continuous" slider and/or text box

- Likert scale: discrete steps (e.g., 5 or 10 stars, possibly in half steps)



- Binary ratings (thumbs up/down)

- Unary ratings ("Like")

(For investigation of impact of different rating scales, cf. [Sparling and Sen, 2011])

User-Item-Matrix / Rating Matrix

- Representation of ratings in a user-item matrix:
  $\mathrm{U} = \{u_1, \dots u_n\}$ … set of users
  $\mathrm{P} = \{p_1, \dots u_m\}$ … set of items
  R … rating matrix ($\mathrm{n} \times m$), $\mathrm{r}_{\mathrm{i,j}}$ represents rating of user i for item j

	Item 1	*Item 2*	*Item 3*	*Item 4*	*Item 5*
User 1	3		2	3	3
User 2	4	3	4	3	
User 3	3	2	1	5	4
User 4		5	4	3	1

- Task: predict missing rating (for item 5) for another active user

	Item 1	*Item 2*	*Item 3*	*Item 4*	*Item 5*
User a	5		3	4	**?**

CF: Model-based vs. Memory-based

Memory-based CF:

- item ratings made by users are stored
- no model is built, but predictions are made on the fly
- nearest neighbor approach (find most similar users or items)
- simple and well-established method
- results easy to interpret → transparency
- very memory-demanding

Model-based CF:

- user-item matrix is factorized
- ratings are explained by “latent factors” in a low-dimensional space
- users and items can be represented in the same space
- once model is created, rating prediction is very fast
- not so simple, but well-established method
- latent factors may not be easy to interpret (just describe variance in the data)

CF: User-based vs. Item-based

User-based CF:

- find similar users based on rated items (vectors over items)
- predict rating as weighted combination of most similar users' ratings

Item-based CF:

- find similar items based on user ratings (vectors over users)
- predict rating as weighted combination of most similar items' ratings

- In real-world applications, item-based methods can scale better.
- Rating biases should be accounted for.

User-Based CF Recommendation

- Idea:
  identify **similar users**, use their ratings to predict missing rating

- Algorithm outline:

1. Calculate similarity of active/target user to all users that have rated the item to predict

2. Select *k* users that have highest similarity (*k-neighborhood*)

3. Compute prediction for item from a weighted combination of the item’s ratings of users in neighborhood
   (weights correspond to similarity)

User-Based CF Recommendation

1. Calculate similarity (=weight) of active user to all users that have rated the item to predict

- Commonly used for user similarity: **Pearson's correlation**

$$sim(a,u) = \frac{\sum_{p \in P'} (r_{a,p} - \bar{r}_a)(r_{u,p} - \bar{r}_u)}{\sqrt{\sum_{p \in P'} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P'} (r_{u,p} - \bar{r}_u)^2}}$$

where *P'* is the set of items rated by both users and $\bar{r}_u$ is the mean rating of user *u*:

$$\bar{r}_u = \frac{1}{|P'|} \sum_{p \in P'} r_{u,p}$$

- Ranges from –1 to +1, requires variance in user ratings (else undefined), accounts for *users' rating biases* (general high or low ratings) by subtracting mean rating

User-Based CF Recommendation

1. Calculate similarity (=weight) of active user to all users that have rated the item to predict

- Pearson's correlation has shown to work best for this purpose
- Alternatives are *Spearman rank correlation (e.g., to deal with different numeric ranges of ratings)*, *Kendall's τ correlation*, *mean squared differences*, *entropy*, etc.

2. Select *k* users that have highest similarity (*neighborhood*)

- Predefine *k*, sort according to similarity scores, and select *k* highest (should not need any further explanation...)

User-Based CF Recommendation

3. Compute prediction for item from a weighted combination of the item's ratings of users in neighborhood

- Predict rating *r'* as weighted average of deviations from neighbors' mean rating on target item

$$r'_{a,p} = \bar{r}_a + \frac{\sum_{u \in K} sim(a,u) * (r_{u,p} - \bar{r}_u)}{\sum_{u \in K} sim(a,u)}$$

where K is the set of the k nearest neighbors and $\bar{r}_a$ the mean rating of the active user a (this time calculated over all of a's ratings)

- Starts from a's rating bias and adds deviations based on similarity

- After predicting all missing values of a, the items with highest prediction will be recommended to a

User-Based CF Recommendation – Example

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	3		2	3	3
~~*User 2*~~	~~4~~	~~3~~	~~4~~	~~3~~	
User 3	3	2	1	4	4
User 4		5	4	3	1
User a	5		3	4	**?**

Back to our example...

- User 2 hasn't rated item 5...

1. Calculate correlations

$$sim(a,u_1) = \frac{(5-4)(3-2.67)+(3-4)(2-2.67)+(4-4)(3-2.67)}{\sqrt{(5-4)^2+(3-4)^2+(4-4)^2}\sqrt{(3-2.67)^2+(2-2.67)^2+(3-2.67)^2}}$$

$$= \frac{0.33+0.67+0}{\sqrt{2}\sqrt{0.11+0.44+0.11}} = \frac{1}{1.15} = 0.87$$

$$sim(a,u_3) = \frac{(5-4)(3-2.67)+(3-4)(1-2.67)+(4-4)(4-2.67)}{\sqrt{(5-4)^2+(3-4)^2+(4-4)^2}\sqrt{(3-2.67)^2+(1-2.67)^2+(4-2.67)^2}}$$

$$= \frac{0.33+1.67+0}{\sqrt{2}\sqrt{4.66}} = \frac{2}{3.05} = 0.65$$

We will ignore all users that are negatively (or un-) correlated!

$$sim(a,u_4) = \frac{(3-3.5)(4-3.5)+(4-3.5)(3-3.5)}{\sqrt{(3-3.5)^2+(4-3.5)^2}\sqrt{(4-3.5)^2+(3-3.5)^2}} = \frac{-0.25-0.25}{0.5} = -1$$

User-Based CF Recommendation – Example

	Item 1	*Item 2*	*Item 3*	*Item 4*	*Item 5*
User 1	3		2	3	3
~~*User 2*~~	~~4~~	~~3~~	~~4~~	~~3~~	
User 3	3	2	1	4	4
User 4		5	4	3	1
User a	5		3	4	**?**

2. Sort and select neighbors (for the setting k=2):

i.e., $K = \{u_1, u_3\}$

3. Calculate prediction for item 5 for user *a*

$$r'_{a,i_5} = 4 + \frac{[0.87 * (3 - 2.75)] + [0.65 * (4 - 2.8)]}{0.87 + 0.65} = 4 + \frac{0.9975}{1.52} = 4.66$$

Thus, we predict a rating of **4.66** (or 4.5 or 5, depending on the scale)

Is this a good prediction?

What would be the predicted rating for item 2?

And which of the two would you recommend to user *a*?

→ optional homework, exercise :)

Advanced User-Based CF Recommendation

- Users' agreement on items that are "controversial" should have more impact than agreement on items everybody likes
  - *inverse user frequency*: items liked by all get less influence
  - *variance weighting factor*: items with high variance in ratings get more influence
- Neighborhood selection should also consider how many items have been co-rated (small overlap might lead to high correlation by chance)
  - *significance weighting*: devalue weights if less than 50 co-rated items
  - *default voting*: assume default values for missing ratings to calculate correlations on union of rated items rather than on intersection
- General problem: proper selection of *k* (or a similarity threshold)
  → *coverage* vs. *noise* (k<10 should be avoided, typically 20<k<50)

Item-Based CF Recommendation

- Idea: identify **similar items** based on all their user ratings; use ratings active user gave to those similar items to predict missing rating

- Algorithm very similar to user-based CF recommendation (instead of the rows in the rating matrix, we now look at the columns)

1. Calculate similarity of items by comparing rating vectors (over users)
   - instead of correlation, calculate *(adjusted) cosine similarity*

2. Select *k* items that have highest similarity (*k-neighborhood*)

3. Compute prediction for item from a weighted combination of the ratings of items in neighborhood (weights correspond to similarity)

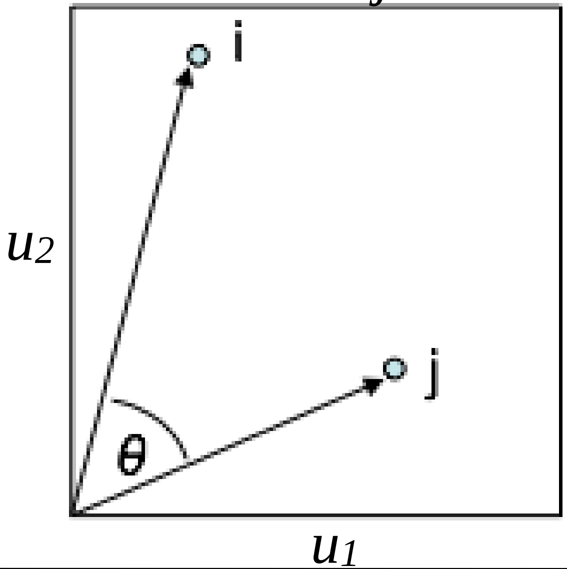
Item Similarity

Cosine similarity:

$$sim(i,j) = \cos\theta = \frac{\vec{i} \cdot \vec{j}}{\left|\vec{i}\right| * \left|\vec{j}\right|} = \frac{\sum_{u \in U} r_{u,i} * r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \sqrt{\sum_{u \in U} r_{u,j}^2}}$$

where $i, j \in P$, $\vec{i}, \vec{j}$ their corresponding rating vectors,
$\cdot$ the dot product of vectors, and $\left|\vec{i}\right|$ the Euclidian length of $\vec{i}$
θ is the angle between the vectors, U users that rated both i and j

- Range: 0 to 1 (1 indicates max. sim.) when all rating values are positive
- Can't deal with missing values
- Does account for different item popularities (due to normalization)
- Does not account for user rating bias!
  → adjusted cosine sim.



Item Similarity 2 and Rating Prediction

Adjusted cosine similarity:

$$sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

where U is the set of users that rated both i and j.

- cf. Pearson correlation; range also from –1 to +1
- Subtracts user u ‘s average from ratings given to items i and j (Not item average! This is not a Pearson correlation on R^T !)

Rating prediction: weighted sum of user’s ratings for similar items

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

where N is the set of the k most similar items

Item-Based Recommendation – Example

Again our example...

1. Subtract mean user rating (saves some time later)

2. User 2 won't help for item 5...

And user *a* hasn't rated item 2...

3. Calculate adjusted cosine similarities (i.e., cosine similarity on mean-adjusted matrix)

	Item 1	*Item 2*	*Item 3*	*Item 4*	*Item 5*
User 1	3		2	3	3
User 2	4	3	4	3	
User 3	3	2	1	4	4
User 4		5	4	3	1
User a	5		3	4	**?**

↓

	Item 1	~~*Item 2*~~	*Item 3*	*Item 4*	*Item 5*
User 1	0.25		-0.75	0.25	0.25
~~*User 2*~~	~~0.5~~	~~-0.5~~	~~0.5~~	~~-0.5~~	
User 3	0.2	~~-0.8~~	-1.8	1.2	1.2
User 4		~~1.75~~	0.75	-0.25	-2.25
User a	5		3	4	**?**

Item-Based Recommendation – Example

(Compacted matrix)

	Item 1	*Item 3*	*Item 4*	*Item 5*
User 1	0.25	-0.75	0.25	0.25
User 3	0.2	-1.8	1.2	1.2
User 4		0.75	-0.25	-2.25
User a	5	3	4	**?**

3. Adjusted cosine similarities

$$sim(i_5,i_1) = \frac{(0.25*0.25)+(1.2*0.2)}{\sqrt{0.25^2+1.2^2}\sqrt{0.25^2+0.2^2}} = 0.77$$

$$sim(i_5,i_3) = \frac{(0.25*-0.75)+(1.2*-1.8)+(-2.25*0.75)}{\sqrt{0.25^2+1.2^2+2.25^2}\sqrt{0.75^2+1.8^2+0.75^2}} = -0.68$$

$$sim(i_5,i_4) = \frac{(0.25*0.25)+(1.2*1.2)+(-2.25*-0.25)}{\sqrt{0.25^2+1.2^2+2.25^2}\sqrt{0.25^2+1.2^2+0.25^2}} = 0.82$$

4. Select $k=2$ most similar (i_1, i_4) and calculate rating prediction

$$r'_{a,i_5} = \frac{0.77*5+0.82*4}{0.77+0.82} = 4.48$$

Item-Based CF Recommendation

- Better suited for large-scale recommenders than user-based CF
- Preprocessing can be performed offline, i.e., all *item-to-item similarities* can be calculated in advance (need update after some time)
  (Could be done for user-to-user similarities too, but user profiles, esp. of new users, change very dynamically)
- More realistic: users rate only small number of items ($<<m$)
  To predict item i, find most similar (item-item similarity matrix lookup), and weight own ratings over these (commonly only few) items
  (In contrast, user-based CF needs to consider all users who rated the target item)
- For item-based CF, at runtime, recommendation in real-time possible

Popular Real-World Data Sets

Name	Domain	Users	Items	Ratings	**Sparsity**
BX	*Books*	278,858	271,379	1,149,780	**0.9999**
EachMovie	*Movies*	72,916	1,628	2,811,983	**0.9763**
Jester	*Jokes*	73,421	101	4.1M	**0.4471**
MovieLens 10M	*Movies*	71,567	10,681	10M	**0.9869**
Netflix	*Movies*	480K	18K	100M	**0.9884**
Yahoo! Music	*Music*	1M	625K	262M	**0.9995**

Sparsity: Fraction/percentage of missing ratings in R

Summary: Memory-based CF

- Memory-based CF recommenders store all ratings made by users and calculate a prediction on the fly
- Very memory-demanding; not the most effective (caching strategies required), item-based CF typically more effective
- Make use of nearest neighbor approaches (**“lazy learning”**)
- Memory-based CF recommenders are easily interpretable, well-established, and researched
- Simple method that is prone to several negative effects (data sparsity, “cold-start”, curse of dimensionality, etc.)
- Solutions to counteract these effect exist: graph-based transitivity, mutual proximity, etc.

References: Textbooks, Tutorials

[Ricci et al., 2015] *Recommender Systems Handbook*, 2nd ed., Springer.

[Aggarwal, 2016] *Recommender Systems: The Textbook*, Springer.

[Jannach et al., 2010] *Recommender Systems - An Introduction*, Cambridge Press, 2010.

[Baeza-Yates and Ribeiro-Neto, 1999] Modern Information Retrieval, Addison Wesley.

[Jannach and Friedrich, 2011] *Tutorial: Recommender Systems*, International Joint Conference on Artificial Intelligence, *http://ijcai-11.iiia.csic.es/files/proceedings/Tutorial%20IJCAI%202011%20Gesamt.pdf*

References: Original Research

[Herlocker et al., 2004] *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems, 22(1).

[Jannach and Friedrich, 2011] *Tutorial: Recommender Systems*, International Joint Conference on Artificial Intelligence, *http://ijcai-11.iiia.csic.es/files/proceedings/Tutorial%20IJCAI%202011%20Gesamt.pdf*

[Lemire and Maclachlan, 2005] *Slope One Predictors for Online Rating-Based Collaborative Filtering*, In SIAM Data Mining.

[Linden et al., 2003] *Amazon.com recommendations: item-to-item collaborative filtering*. In IEEE Internet Computing, 7(1).

[Melville and Sindhwani, 2010]. *Recommender Systems*, In Encyclopedia of Machine Learning, Springer.

[Resnick et al., 1994] *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*, In Proceedings of the CSCW.

[Sparling and Sen, 2011] *Rating: how difficult is it?*, In Proceedings of the 5th ACM Conference on Recommender Systems.