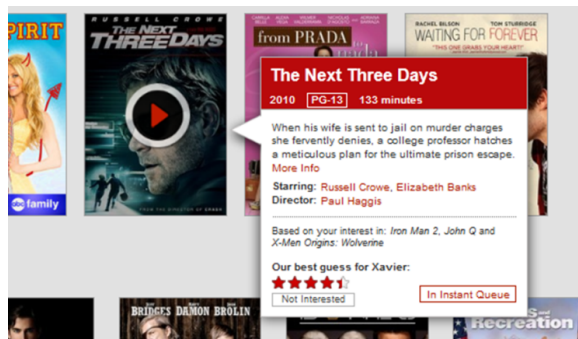


Chapter 6

Recommender Systems

- 6.1 Introduction to Recommender Systems
- 6.2 Neighborhood-based Collaborative Filtering
- 6.3 Model-based Collaborative Filtering
- 6.4 Content-based Recommender Systems
- 6.5 Knowledge-based Recommender Systems
- 6.6 Evaluating Recommender Systems
- 6.7 Advanced Topics in Recommender Systems

What is a recommender system? (And why do we need it?)



Source: <http://techblog.netflix.com>, 2012

Recommender System

A **recommender system** aims at **predicting** the **rating** or **preference** that a **user** would give to an **item** that she has not yet considered.



¹Image source: Yasir72.multan [CC BY-SA 3.0], via Wikimedia Commons

Items and Ratings

Examples for the **items**:

- ▶ Songs, movies, books, news, blogs, products, people, venues,...

A **rating** can be expressed in various ways:

- ▶ **Interval-based ratings**: for example, by using a five-point (or five-star) scale. An important assumption is that the values explicitly define the distance between the ratings.
- ▶ **Ordinal ratings**: for example *disagree*, *neutral*, *strongly agree* etc. It is not assumed that the distance between any pair of adjacent rating values is the same.
- ▶ **Binary ratings**: only two options are present.
- ▶ **Unary ratings**: the user is allowed to specify a positive preference for an item, but there is no mechanism to specify a negative preference.
- ▶ **Continuous ratings**: the user can specify a rating on a continuous scale (for example, any real number between -10 and 10). This approach is used relatively rare.

Goals of recommender systems

General goals of recommender systems:

- ▶ Provide recommendations that are **relevant** for the user.
- ▶ Recommend items that are **novel** for the user, i.e., the user has not seen them in the past.
- ▶ In some cases, **serendipity** is also desirable. Such recommendations are truly surprising to the user and not just something she did not know before.
- ▶ Provide recommendations that are **diverse**.

In general, the total number of items is large, but the number of items that have already been rated explicitly (or implicitly) by a user is small.

The central question is, how can we assess the utility of unrated items based on existing ratings of other items? As soon as the utility for all items has been computed, we can recommend the user the items with the "best" predicted rating.

Types of Recommender Systems

There are several **types of recommender systems**:

1. **Collaborative Filtering** Recommender System: recommend items to the user which have been preferred by other users with similar preferences in the past.
 - ▶ **Memory-based methods** (also called Neighborhood-based Collaborative Filtering)
 - ▶ **User-based Collaborative Filtering**
 - ▶ **Item-based Collaborative Filtering**
 - ▶ **Model-based methods**
2. **Content-based** Recommender Systems: recommend items to the user which are similar to the ones the user preferred in the past.
3. **Knowledge-based** Recommender Systems
4. **Hybrid** Recommender Systems

Collaborative Filtering

Collaborative filtering recommender systems can be subdivided into **memory-based** and **model-based** approaches.

- ▶ Memory-based approaches are also known as **neighborhood-based collaborative filtering**.

There are two primary types of neighborhood-based approaches:

1. **User-based collaborative filtering**: ratings provided by users that are similar to a target user u are used to make recommendations for u .
2. **Item-based collaborative filtering**: in order to predict the rating a user u would give to a particular item i , we first determine a set of other items S , which are most similar to i . Then a weighted average of the ratings in S is used to predict the rating that u will likely give to i .

The question is: how to determine the similarity between users and between items?

Neighborhood-based Collaborative Filtering

- ▶ The user-item ratings are usually expressed in an $m \times n$ **ratings matrix** \mathbf{R} , where m is the number of users in the system and n is the number of items.
- ▶ The ij -th element of \mathbf{R} indicates the rating of user i for item j .
- ▶ Typically the ratings matrix is incomplete because not all users have rated all items.
- ▶ The goal of the recommender system is to predict the values for the empty cells.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	7	6	7	4	5	4
User 2	6	7	?	4	3	4
User 3	?	3	3	1	1	?
User 4	1	2	2	3	3	4
User 5	1	?	1	2	3	3

User-based Collaborative Filtering

The idea behind user-based collaborative filtering is to identify other users that are **similar** to our **target user** (the user we want to predict ratings for). To determine the **neighborhood** of the target user, we need to compute her similarity to all other users.

- ▶ For the $m \times n$ ratings matrix \mathbf{R} with m users and n items, let I_u denote the set of item indices for which ratings have been specified by user (row) u .
- ▶ For example¹, if the ratings of the first, third, and fifth items (columns) of user (row) u are specified (i.e. the user has rated the items before) and the remaining are missing, then we have $I_u = \{1, 3, 5\}$. Therefore, the set of items rated by both users u and v is given by $I_u \cap I_v$. For example, if user v has rated the first four items, then $I_v = \{1, 2, 3, 4\}$, and $I_u \cap I_v = \{1, 3, 5\} \cap \{1, 2, 3, 4\} = \{1, 3\}$. It is possible (and quite common) for $I_u \cap I_v$ to be an empty set because ratings matrices are generally sparse.
- ▶ The set $I_u \cap I_v$ defines the **mutually observed ratings**, which are used to compute the similarity between the u th and v th users for neighborhood computation.

¹Example adopted from: Recommender Systems The Textbook, C. C. Aggarwal, Springer, 2016

User-based Collaborative Filtering

Based on the set $I_u \cap I_v$ we can now calculate the **similarity** between users u and v .

- First, we compute the **mean rating** μ_u for every user u :

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|} \forall u \in \{1 \dots m\}$$

- Then, the **Pearson correlation coefficient** between the rows (users) u and v is defined as follows:

$$Sim(u, v) = Pearson(u, v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

Repetition: Pearson's Correlation

Pearson's correlation coefficient ρ_{xy} is a measure of the linear dependence between two variables x and y , giving a value between -1 and $+1$ inclusive. It is defined as the *covariance* of the two variables divided by the product of their *standard deviations*. Writing the covariance and standard deviation in full gives

$$\rho_{xy} = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2}}$$

where \bar{x} and \bar{y} denote the mean of x and y respectively. $1/n$ can be dropped:

$$\rho_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

User-based Collaborative Filtering

Let $P_u(j)$ be the set of users who are similar to the target user u (i.e. the users who are in the **neighborhood** of u) and who have rated item j before.

- The **predicted rating** \hat{r}_{ui} of user u for item j is then given by

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{sim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{sim}(u, v)|}$$



User-based Collaborative Filtering: Example

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Mean rating	Pearson(i,3)
User 1	7	6	7	4	5	4	5.5	0.894
User 2	6	7	?	4	3	4	4.8	0.939
User 3	?	3	3	1	1	?	2	1.0
User 4	1	2	2	3	3	4	2.5	-1.0
User 5	1	?	1	2	3	3	2	-0.817

$$\text{sim}(1, 3) = \frac{(6 - 5.5)(3 - 2) + (7 - 5.5)(3 - 2) + (4 - 5.5)(1 - 2) + (5 - 5.5)(1 - 2)}{\sqrt{0.5^2 + 1.5^2 + (-1.5)^2 + (-0.5)^2} * \sqrt{1^2 + 1^2 + (-1)^2 + (-1)^2}} \approx 0.894$$

- For user 3, the users 1 and 2 are most similar.

User-based Collaborative Filtering

In case we applied just *raw* ratings in the prediction function:

$$\hat{r}_{31} = \frac{0.894 * 7 + 0.939 * 6}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{0.894 * 4 + 0.939 * 4}{0.894 + 0.939} = 4$$

Taking into consideration the mean-centered ratings we obtain

$$\hat{r}_{31} = 2 + \frac{0.894 * 1.5 + 0.939 * 1.2}{0.894 + 0.939} \approx 3.35$$

$$\hat{r}_{36} = 2 + \frac{0.894 * (-1.5) + 0.939 * (-0.8)}{0.894 + 0.939} \approx 0.86$$

User-based Collaborative Filtering: Discussion

Improving the metrics / prediction function:

- ▶ Not all neighbor ratings might be equally "valuable"
 - ▶ Agreement on commonly liked items is not so informative as agreement on controversial items
 - ▶ Possible solution: give more weight to items that have a higher variance
- ▶ Value of number of co-rated items
 - ▶ Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- ▶ Case amplification
 - ▶ Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- ▶ Neighborhood selection
 - ▶ Use similarity threshold or fixed number of neighbors

User-based Collaborative Filtering

- ▶ Scalability issues can arise with user-based collaborative filtering if there are many more users than items ($m \gg n$, $m = |users|$, $n = |items|$):
 - ▶ Space complexity $O(m^2)$ when pre-computed
 - ▶ Time complexity for computing Pearson $O(m^2n)$
- ▶ High sparsity leads to few common ratings between two users.
- ▶ In this case we can apply item-based collaborative filtering, which exploits relationships between items first, instead of relationships between users.

Item-based Collaborative Filtering

Another type of memory-based approach is called **item-based collaborative filtering**, which utilizes the similarity between *items* for calculating a rating prediction.

- ▶ The rating prediction for a target item of a target user is based on a set of other *items* that are similar to the target item.
- ▶ Item-to-item similarity is commonly based on measuring the **cosine similarity** between the rating vectors. The (raw) cosine similarity measures the similarity between two non-zero vectors in terms of the cosine of the angle between them.

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

or, using a different notation:

$$\cos(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^n a_i * b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

where a_i denotes the i -th element of vector \vec{a} .

Item-based Collaborative Filtering

- Considering only the raw cosine usually leads to inaccurate predictions because of the rating bias.

A more common approach is therefore to apply an **adjusted cosine similarity** measure in order to calculate the similarity between two items i and j .

Let U_i be the set of users who have rated item i before, and let μ_u be the mean rating of a user u .

$$\text{AdjustedCosine}(i, j) = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \mu_u)(r_{uj} - \mu_u)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \mu_u)^2} * \sqrt{\sum_{u \in U_i \cap U_j} (r_{uj} - \mu_u)^2}}$$

The cosine similarity is bounded between $[-1, 1]$. If both vectors are oriented in positive space then the outcome is bounded between $[0, 1]$.

Item-based collaborative filtering

Let $Q_t(u)$ be the set of top- k items most similar to item t for which user u has specified ratings.

- ▶ Then we can apply, for example, the following prediction function:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} \text{AdjustedCosine}(j, t) * r_{uj}}{\sum_{j \in Q_t(u)} |\text{AdjustedCosine}(j, t)|}$$

Item-based Collaborative Filtering: Example

Consider the ratings as before for the user-based approach, however, this time with **mean-centered** ratings:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	1.5	0.5	1.5	-1.5	-0.5	-1.5
User 2	1.2	2.2	?	-0.8	-1.8	-0.8
User 3	?	1	1	-1	-1	?
User 4	-1.5	-0.5	-0.5	0.5	0.5	1.5
User 5	-1	?	-1	0	1	1
Cosine(1,j)	1	0.735	0.912	-0.848	-0.813	-0.990
Cosine(6,j)	-0.990	-0.622	-0.912	0.829	0.730	1

$$AdjustedCosine(1, 3) = \frac{1.5 * 1.5 + (-1.5) * (-0.5) + (-1) * (-1)}{\sqrt{1.5^2 + (-1.5)^2 + (-1)^2} * \sqrt{1.5^2 + (-0.5)^2 + (-1)^2}} \approx 0.912$$

Items 2 and 3 are most similar to item 1. Items 4 and 5 are most similar to item 6.

Item-based Collaborative Filtering: Example

Using the AdjustedCosine similarity measure, we can now calculate the rating predictions, for example, for user 3:

- For predicting the rating of user 3 for item 1, we consider the two items 2 and 3 (because of high AdjustedCosine similarity with the target item 1):

$$\hat{r}_{31} = \frac{3 * 0.735 + 3 * 0.912}{0.735 + 0.912} = 3$$



- For predicting the rating of user 3 for item 6, we consider items 4 and 5 (because they have high similarity with target item 6):

$$\hat{r}_{36} = \frac{1 * 0.829 + 1 * 0.730}{0.829 + 0.730} = 1$$



Note that the predicted values are within the range of the original rating scale (1 to 7).

Item-based Collaborative Filtering


- ▶ Item-based collaborative filtering does not solve the scalability problem itself.
- ▶ However, we can calculate all pair-wise item similarities in advance, because item similarities are supposed to be more stable than user similarities.
- ▶ The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated.

Memory requirements:

- ▶ In theory, up to n^2 pair-wise similarities need to be memorized (n = number of items).
- ▶ In practice, this is significantly lower (items with no co-ratings).
- ▶ Further reductions possible, for example:
 - ▶ Minimum threshold for co-ratings (items, which are rated at least by t users).
 - ▶ Limit the size of the neighborhood (might affect recommendation accuracy).

Discussion

General issues to consider:

- ▶ **New user problem:** the system first needs to learn the preferences of the user from his ratings.
- ▶ **New item problem:** items can only be recommended when a substantial number of users have rated it before.
- ▶ **Sparsity:** many items that have been rated by only few users each. Even if they got a high ranking, they would be recommended rarely. Also, a user whose preferences are unusual compared to all other users will get poor recommendations. 

A solution to the problem of rating sparsity is to segment users based on their profile, for example age, gender, country, education (a.k.a. demographic filtering).