MATH 60629A
Machine Learning I

Recommender Systems

Agenda

What is a Recommender System?

- Quick overview

Some basic and mainstream methods

- Direct method
- User-to-User filtering
- Item-to-Item filtering
- Matrix factorization
- RNN

There are tens of other methods.

What is a Recommender System?

When you log in to Netflix, Amazon, Expedia, etc.,

What movies/products/hotels do you see?

Why do you see them, but not others? What does differentiate these movies/products/hotels from others?

They are determined based on a general recommender system:

- Past behavior (e.g. cookies)
- User preferences
- Product characteristics

Question Overview

What should we replace “?” with?

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

The answer to this question is the output of a recommender system.

Direct Method

- The simplest and most direct method to set up a recommendation system is to ask direct questions.
- In this method, first, each item is characterized by a set of features. For example, a movie can be characterized by genre, actors/actresses, director, etc. These characteristic sets are represented by a vector for each item.
- Subsequently, users are asked about their preferences towards each of those item characters. Their preferences are also represented by vectors.
- Multiplying each item's characteristic vector with each user's preference vector returns a value associated with how likely that user is attracted to that item.

User-to-User Filtering

- A class of collaborative filtering algorithms used in recommender systems.
  - predict the preferences of a user by collecting preferences from many users (collaborating)
  - Underlying assumption: if user A and user B rate a movie similarly, it is more likely that they rate similarly other another movie as well.
- This method is based on similarity of different users with regards to similar items (movies in our case).
- Here are the steps:
  1. Calculate Cosine Similarity among users
  2. Rank users based on their similarity
  3. Take the (weighted) average of top N users

User-to-User Filtering

Example 1 (2 nearest):

	User 1	User 2	User 3	User 4
Movie 1	2	2	5	3
Movie 2	?	1	2	3
Movie 3	4	5	1	3

$$U_1 = [2, 4]$$
$$U_2 = [2, 5]$$
$$U_3 = [5, 1]$$
$$U_4 = [3, 3]$$

$$Cosine(U_1, U_2) = 0.996$$
$$Cosine(U_1, U_3) = 0.614$$
$$Cosine(U_1, U_4) = 0.949$$

User 1-Movie 2$= \frac{(0.996 \times 1 + 0.949 \times 3)}{(0.996 + 0.949)} = 1.97 \approx 2$

What if the matrix is sparse? Cosine similarity does not work with null values!

User-to-User Filtering

	User 1	User 2	User 3	User 4
Movie 1	2		5	3
Movie 2	?	1	2	
Movie 3	4	5		3
Average	**3**	**3**	**3.5**	**3**

	User 1	User 2	User 3	User 4
Movie 1	-1	0	1.5	0
Movie 2	?	-2	-1.5	0
Movie 3	1	2	0	0

Two possible strategies:

1. ***Replace each value with the average of the user subtracted from that value, and then fill in missing values for other users with 0***
   - **This is to account for the user bias.**
2. Replace each value by the logical average value (3) subtracted from that value, and then replace the missing values with zero
   - Does not account for individual biases. Assumes the average rating of 3 of everyone which is a strong assumption.

$$U_1 = [-1, 1]$$
$$U_2 = [0, 2]$$
$$U_3 = [1.5, 0]$$
$$U_4 = [0, 0]$$

$$Cosine(U_1, U_2) = 0.707$$
$$Cosine(U_1, U_3) = -0.707$$
$$Cosine(U_1, U_4) = 0$$

$$\text{User 1-Movie 2} = \frac{(-2 \times 0.707 + 0 \times 0)}{(0.707 + 0)} = -2 \rightarrow \mathbf{1}$$

Centered Cosine Similarity = Pearson Correlation Coefficient

User-to-User Filtering

	User 1	User 2	User 3	User 4
Movie 1	2		5	3
Movie 2	?	1	2	
Movie 3	4	5		3
Average	**3**	**3**	**3**	**3**

	User 1	User 2	User 3	User 4
Movie 1	-1	0	2	0
Movie 2	?	-2	-1	0
Movie 3	1	2	0	0

Two possible strategies:

1. Replace each value with the average of the user subtracted from that value, and then replace missing values in other users with 0
   - This is to account for the user bias.
2. ***Replace each value by the logical average value (3) subtracted from that value, and then fill in the missing values with zero***
   - **Does not account for individual biases. Assumes the average rating of 3 of everyone which is a strong assumption.**

$$U_1 = [-1, 1]$$
$$U_2 = [0, 2]$$
$$U_3 = [2, 0]$$
$$U_4 = [0, 0]$$

$$Cosine(U_1, U_2) = 0.707$$
$$Cosine(U_1, U_3) = -0.707$$
$$Cosine(U_1, U_4) = 0$$

$$\text{User 1-Movie 2} = \frac{(-2 \times 0.707 + 0 \times 0)}{(0.707)} = -2 \rightarrow \mathbf{1}$$

Note: The rating prediction results are usually different when we use non-centerd or centered ratings. We expect to obtain more accurate predictions when we take the bias into account. However, experimentation is recommended.

Some Points to Consider

- It is easy to understand.
- It is not efficient for sparse matrices: there should be enough ratings per user in order to find similarity between users.
- Scalability is an issue: for each new rating estimation, all similarity values need to be calculated.
- Grey sheep/black sheep: a user may be dissimilar to all other users, or a user group may not have interacted with an item at all (stemming from sparsity issue)

Some Points to Consider

What if a user is dissimilar to others?

What if all users who are similar to a user have not interacted with the item (movie, article, etc.)?

A possible way to deal with this situation is to use a baseline approach. For example:

- A user's average rating of all items is 2.5
- Average rating of all items is 3.5
- Average rating of this item is 4

Baseline (best guess): 3.5 (average of all items) + 0.5 (how this item outperforms/underperforms all other items on average) -1 (to adjust for this user's average rating which is 1 point below the population) = 3

Some Points to Consider

- If we are dealing with ratings, *Cosine Similarity* is only one way to measure similarity.

- The distance function can be anything, such as *Euclidean distance*.

- If the interaction between user and item is defined as a binary variable (such as purchasing or not, clicking on the ad or not, etc.), *Jaccard Similarity* is also an option.

Item-to-Item Filtering

- A class of collaborative filtering algorithms used in recommender systems.
- The described method was based on user preferences. The same exact method can be developed based on item similarity.
- Depending on the context, one of these methods could outperform the other one.
- Example 1, revisited:

	User 1	User 2	User 3	User 4
Movie 1	2	2	5	3
Movie 2	?	1	2	3
Movie 3	4	5	1	3

$M_1 = [2, 5, 3]$ $\quad Cosine(M_1, M_2) = 0.910$

$M_2 = [1, 2, 3]$ $\quad Cosine(M_2, M_3) = 0.658$

$M_3 = [5, 1, 3]$

User 1-Movie 2$= \frac{(0.91 \times 2 + 0.658 \times 4)}{(0.91 + 0.658)} = 2.84 \approx 3$

But which one is the right answer? 2 or 3?

Matrix Factorization

- If you remember from the first slides, a direct recommender method is to describe each item or movie as a set of characteristics (e.g., movie genres), and then ask users how they value those characteristics (i.e., user preference). The recommendation then was the result of combining (i.e., multiplying) user preferences and item characteristics.

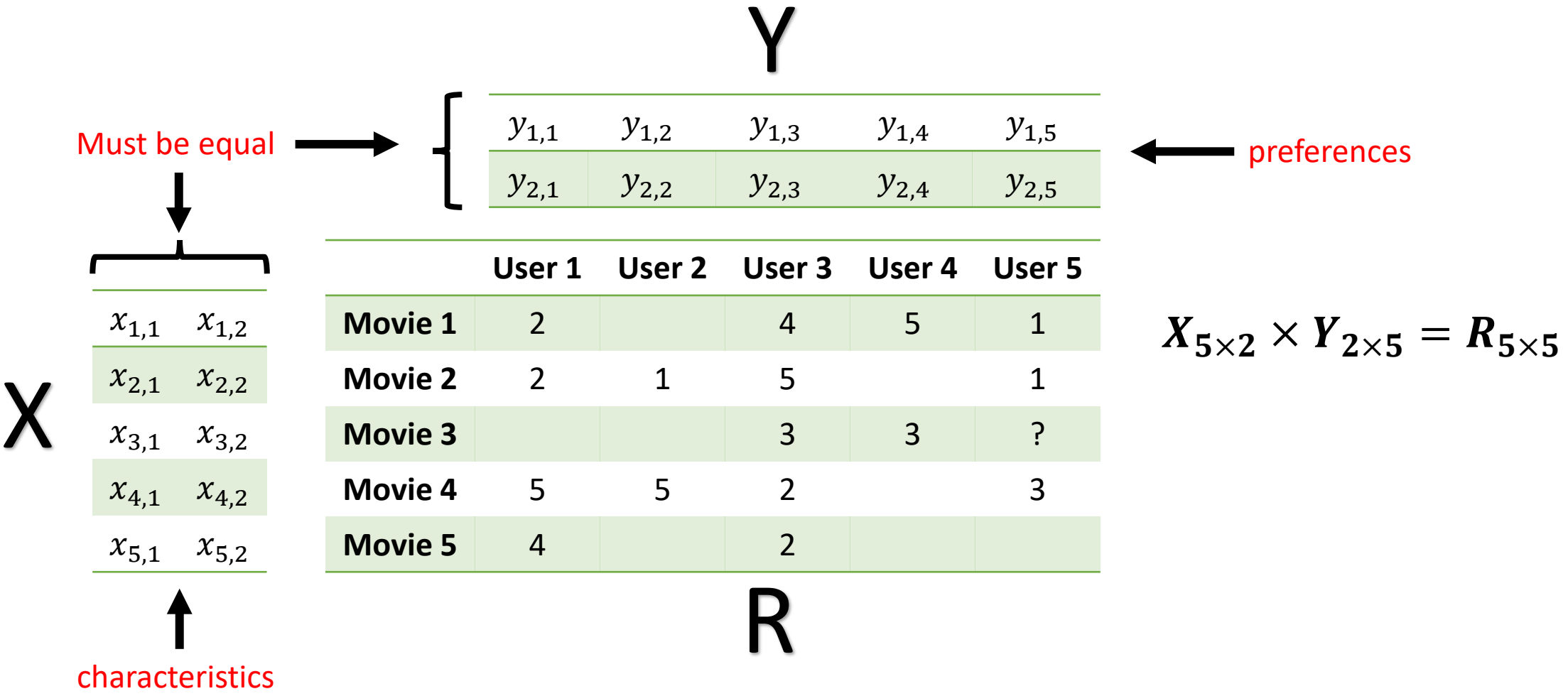
- Matrix Factorization does that! It tries to estimate ratings based on the characterization each user and item.

- In this method, preferences and characteristics are not necessarily meaningful. They are abstract representations learned from data.

Matrix Factorization

- A class of collaborative filtering algorithms used in recommender systems.
- The main idea is to approximate the observed ratings in a large, sparse user-item matrix by factorizing it into two lower-dimensional matrices that represent
  - Latent features of items: genre, cast, director
  - Latent feature of users: preferences for genre, actor/actress, director

Matrix Factorization



	User 1	User 2	User 3	User 4	User 5
Movie 1	2		4	5	1
Movie 2	2	1	5		1
Movie 3			3	3	?
Movie 4	5	5	2		3
Movie 5	4		2		

$$\boldsymbol{X_{5\times 2} \times Y_{2\times 5} = R_{5\times 5}}$$

Matrix Factorization

- **Learning Objective:** Find matrices X and Y such that the squared error function is minimized:

$$\sum_{i,j} \left(R_{i,j} - X_i Y_j\right)^2$$

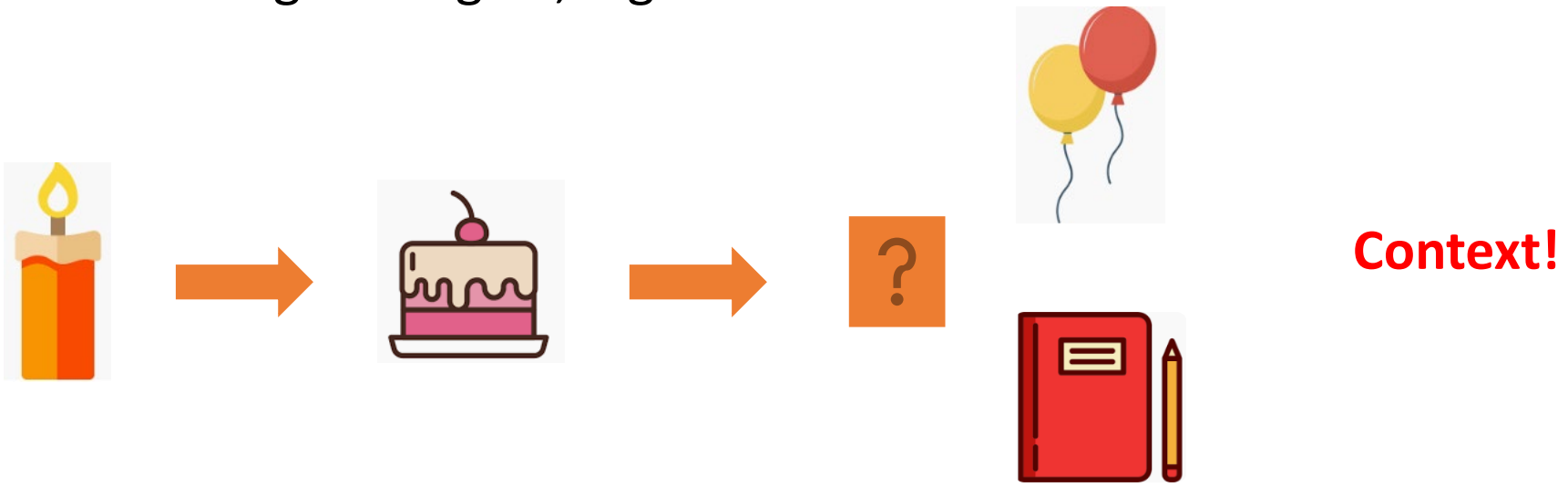
where $R_{i,j}$ represents entry (i, j) of matrix R, X_i represents the ith row of X and Y_j represents the jth column of Y.

  - **Optimization Method**: (typically) gradient descent

- **Prediction:** Rating for user i and item j is predicted by computing the dot product of i-th column of Y and j-th column of X^T.

RNN

- RNN is designed to handle sequential data.
- Variations of RNN such as LSTM are better at capturing long-term memory.
- We can build an RNN-based recommender when the sequence of user interactions with the catalog items matters. In other words, when user interactions with items are temporally dependent or the sequence of interactions provides meaningful insights, e.g. for session data.



Final Notes

- There are tens of other methods and several variations for the methods we described in this session.

- Remember: More complicated does not always mean better!