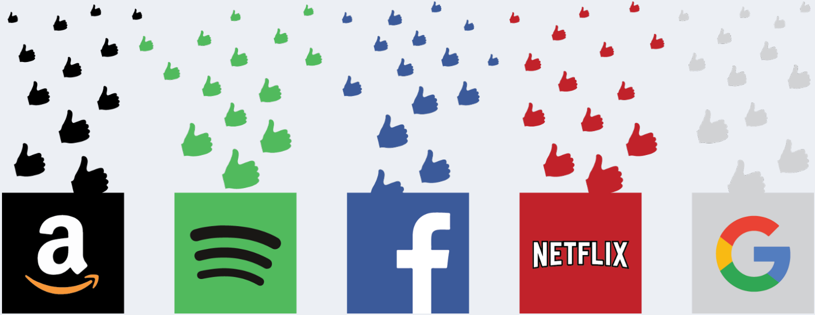
STAT3009 Recommender Systems

Lec1: Overview of Recommender Systems

by Ben Dai (The Chinese University of Hong Kong)
on Department of Statistics

» Big Picture: Recommender Systems

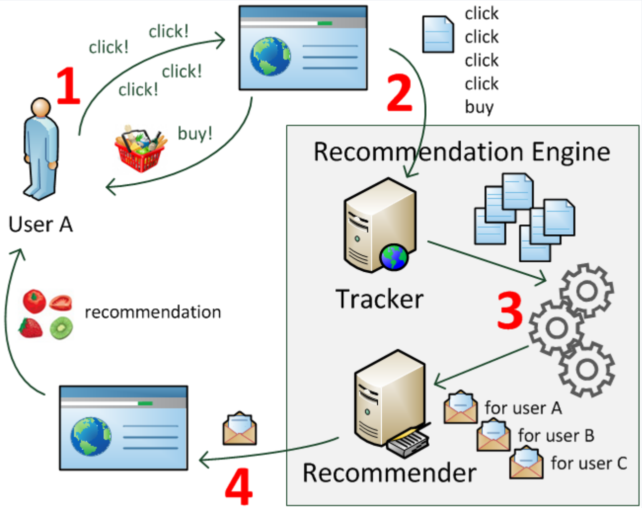
Netflix, YouTube, Taobao and Amazon are all examples of recommender systems (RSs) in use. The systems recommend users with relevant items (suggestions) based on **users' historical data**.



Source[1]

1 thedatascientist.com/right-way-recommender-system-startup

» Big Picture: Recommender Systems



Source[2]

2 medium.com/voice-tech-podcast/a-simple-way-to-explain-the-recommendation-engine-in-ai-

» Advantages: Recommender Systems

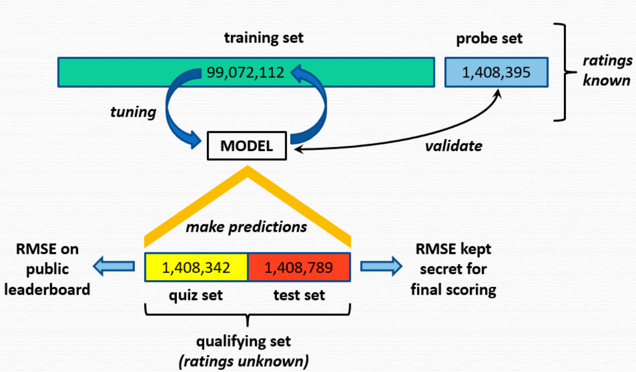
- **Revenue Increase.** The successful recommendation systems lead to the **29% annual sales increase** for Amazon[3].
- **User Satisfaction Increase.** A good RS could offer the personalized suggestions and understand the users' needs.
- **Other Application.** Financial products, medicine recommender systems, ...

That's why the RS is one of the **most valuable** applications of Machine Learning (ML).

[3] https://azati.ai/recommendation-systems-benefits-and-issues/

» Example: Netflix Prize

- **Netflix Prize Competition:** 51051 contestants on 41305 teams from 186 different countries
- $1 million prize for **10% improvement** on Netflix
- **Competition Structure**



Source[4]

[4] https://pantelis.github.io/cs301/docs/common/lectures/recommenders/netflix/

» Dataset: Netflix Prize

- **Netflix Prize Competition**
- **Training dataset**[5]: [**MovieIDs**, **CustomerIDs**, **Ratings**]
    - **MovieIDs** range from 1 to 17770 sequentially.
    - **CustomerIDs** range from 1 to 2649429, with gaps. There are 480189 users.
    - 100 million **Ratings** are on a five star (integral) scale from 1 to 5.
    - **InClass demo: load Netflix dataset via Python**
- **Testing dataset:** [**MovieIDs**, **CustomerIDs**]
- **Evaluation:** The root mean squared error (RMSE)

We need a **formal mathematical model** to formulate a RS problem.

[5] https://www.kaggle.com/netflix-inc/netflix-prize-data

» Data: Recommender Systems

	Gladiator	Space Jam	Pitch Perfect	Life of Pi	Dear Basketball
Rajon	?	?	3	?	?
James	5	5	?	?	5
Davis	?	?	?	4	5
Dwight	?	?	4	?	5
Bryant	?	?	5	4	5

* **Goal:** Can you predict the rating with question mark?

» Formal Model: Recommender Systems

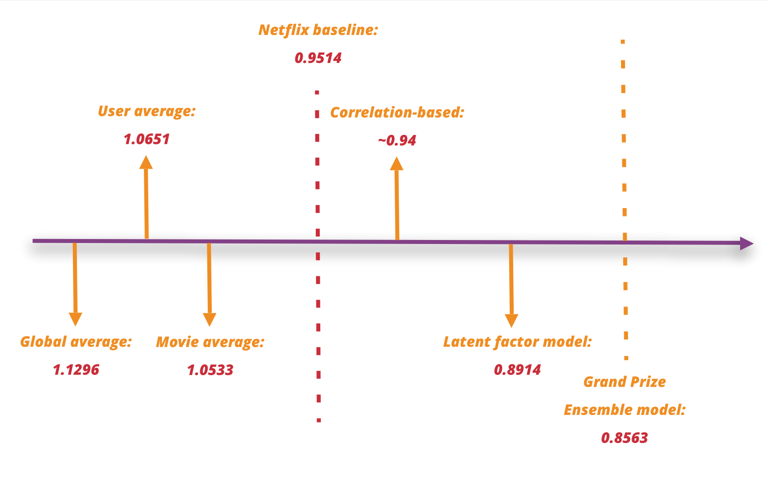
* **User:** $u = 1, \cdots, n$ with n is number of users
* **Item:** $i = 1, \cdots, m$ with m is number of items
* **Rating:** r_{ui} is the u-th user rating in the i-th item.
* **Obs Index set:** $(u,i) \in \Omega$ if the rating for (u,i) pair is observed.
* **Evaluation:** Given a testing index set Ω^{te} (set of user-item pairs we want to predict),

$$RMSE = \Big(\frac{1}{|\Omega^{\text{te}}|} \sum_{(u,i) \in \Omega^{\text{te}}} \big(\hat{r}_{ui} - r_{ui} \big)^2 \Big)^{1/2}.$$

* **Goal:** Find predicted ratings $(\hat{r}_{ui})_{(u,i) \in \Omega^{\text{te}}}$ such that **minimizes RMSE**

InClass demo: Data demonstration of RS datasets

» Methods: Netflix Prize



* We will cover all methods in this course.

» Global Average Method

* **Simple baseline method** that predicts new ratings by taking the average over all observed ratings.
* This method assumes that the overall rating distribution is a good representation of the user's preferences.

» Global Average Method: Formula

The global average rating is calculated as:

$$\bar{r} = \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} r_{ui}$$

where Ω is the set of all observed user-item ratings.

» Global Average Method: Example Calculation

Suppose we have the following ratings:

User ID	Item ID	Rating
1	1	4
1	2	3
1	3	5
2	1	2
2	2	4
2	3	3
3	1	3
3	2	2
3	3	4

The global average rating is:

$$\bar{r} = \frac{4+3+5+2+4+3+3+2+4}{9} = 3.44$$

» Global Average Method: Predictions

Now, if we want to predict the rating of user 1 for a new item 4, we would predict:

$$\hat{r}_{ui} = \bar{r} = 3.44$$

This method ignores user and item differences and provides a single average rating for all users and items.

» Implementation in Python: Rough Workflow

1. Load the rating data into a Pandas DataFrame with columns for user ID, item ID, and rating.
2. Calculate the global average rating by taking the mean of the rating column.
3. Define a function that takes in a user ID and item ID, and returns the global average rating as the predicted rating.
4. Use this function to make predictions for a test set of user-item pairs.

» Global Average Method: Advantages and Disadvantages

Advantages

- Extremely simple to implement
- Fast computation
- Can be used as a baseline for more complex models

Disadvantages

- Ignores user and item differences
- Can be biased towards the majority of ratings
- Does not provide personalized recommendations

» User Average Method

- **User baseline method** that predicts new ratings by taking the average of a user's past ratings.
- This method assumes that a user's past behavior is a good representation of their future preferences.

» User Average Method: Formula

The user average rating is calculated as:

$$\bar{r}_u = \frac{1}{|I_u|} \sum_{i \in I_u} r_{ui}$$

where I_u is the set of items rated by user u.

» User Average Method: Example Calculation

Suppose we have the following ratings for multiple users:

User ID	Item ID	Rating
1	1	4
1	2	3
1	3	5
2	1	2
2	2	4
2	3	3
3	1	3
3	2	2
3	3	4

The user average ratings are:

$$\bar{r}_1 = \frac{4+3+5}{3} = 4, \quad \bar{r}_2 = \frac{2+4+3}{3} = 3.67, \quad \bar{r}_3 = \frac{3+2+4}{3} = 3$$

» User Average Method: Predictions

Now, if we want to predict the rating of user 1 for a new item 4, we would predict:

$$\hat{r}_{1,4} = \bar{r}_1 = 4$$

Similarly, if we want to predict the rating of user 2 for a new item 4, we would predict:

$$\hat{r}_{2,4} = \bar{r}_2 = 3$$

This method takes into account a user's past behavior, but ignores item differences.

» Implementation in Python: Rough Workflow

1. Load the rating data into a Pandas DataFrame with columns for user ID, item ID, and rating.
2. Calculate the user average rating for each user by taking the mean of their ratings.
3. Define a function that takes in a user ID and item ID, and returns the user average rating as the predicted rating.
4. Use this function to make predictions for a test set of user-item pairs.

» User Average Method: Advantages and Disadvantages

Advantages

* Simple to implement
* Fast computation
* Takes into account a user's past behavior

Disadvantages

* Ignores item differences
* Can be biased towards a user's past ratings
* Does not provide personalized recommendations for new users

» Item Average Method

* **Item baseline method** that predicts new ratings by taking the average of an item's observed ratings.
* This method assumes that an item's past ratings are a good representation of its future ratings.

» Item Average Method: Formula

The item average rating is calculated as:

$$\bar{r}_i = \frac{1}{|\mathcal{U}_i|} \sum_{u \in \mathcal{U}_i} r_{ui}, \text{ for } i = 1, \cdots, m$$

where $\mathcal{U}_i = \{u : (u, i) \in \Omega\}$ is the index set for observed ratings of the i-th item.

» Item Average Method: Example Calculation

Suppose we have the following ratings for multiple items:

User ID	Item ID	Rating
1	1	4
2	1	3
3	1	5
1	2	2
2	2	4
3	2	3

The item *average* ratings are:

$$\bar{r}_1 = \frac{4+3+5}{3} = 4$$

$$\bar{r}_2 = \frac{2+4+3}{3} = 3$$

» Item Average Method: Predictions

Now, if we want to predict the rating of user 1 for item 2, we would predict:

$$\hat{r}_{1,2} = \bar{r}_2 = 3$$

This method takes into account an item's past ratings, but ignores user differences.

» Implementation in Python: Rough Workflow

1. Load the rating data into a Pandas DataFrame with columns for user ID, item ID, and rating.
2. Calculate the item average rating for each item by taking the mean of its ratings.
3. Define a function that takes in a user ID and item ID, and returns the item average rating as the predicted rating.
4. Use this function to make predictions for a test set of user-item pairs.

» Item Average Method: Advantages and Disadvantages

Advantages

* Simple to implement
* Fast computation
* Takes into account an item's past ratings

Disadvantages

* Ignores user differences
* Can be biased towards an item's past ratings
* Does not provide personalized recommendations for new items

» User-Item Average Method

* **User-Item hybrid method** that combines the user average and item average methods to predict new ratings.
* This method takes into account both a user's past behavior and an item's past ratings.

» User-Item Average Method: Formula

The predicted rating is calculated as:

$$\hat{r}_{ui} = \bar{r} + \mu_u + \mu_i$$

where:

$$\mu_u = \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} (r_{ui} - \bar{r})$$

and:

$$\mu_i = \frac{1}{|\mathcal{U}_i|} \sum_{u \in \mathcal{U}_i} (r_{ui} - \bar{r} - \mu_u)$$

» User-Item Average Method: Interpretation

The user-item average method can be interpreted as a multi-stage prediction process:

1. First, the global average rating is calculated.
2. Then, the user and item deviations from the global average are calculated.
3. Finally, the predicted rating is calculated by adding the global average and the user and item deviations.

» Implementation in Python: Rough Workflow

1. Load the rating data into a Pandas DataFrame with columns for user ID, item ID, and rating.
2. Calculate the global average rating.
3. Calculate the user deviations from the global average.
4. Calculate the item deviations from the **global and user average**.
5. Define a function that takes in a user ID and item ID, and returns the predicted rating.
6. Use this function to make predictions for a test set of user-item pairs.

» User-Item Average Method: Advantages and Disadvantages

Advantages

* Takes into account both a user's past behavior and an item's past ratings
* Can provide more accurate predictions than the user average or item average methods

Disadvantages

* Can be computationally expensive
* May not perform well for users or items with few ratings

We have now covered the basics of recommender systems, including:

* The global *average* method
* The user *average* method
* The item *average* method
* The user-item *average* method

These methods *are* simple to implement *and* can provide *a* good baseline for more complex models.

Thank you!