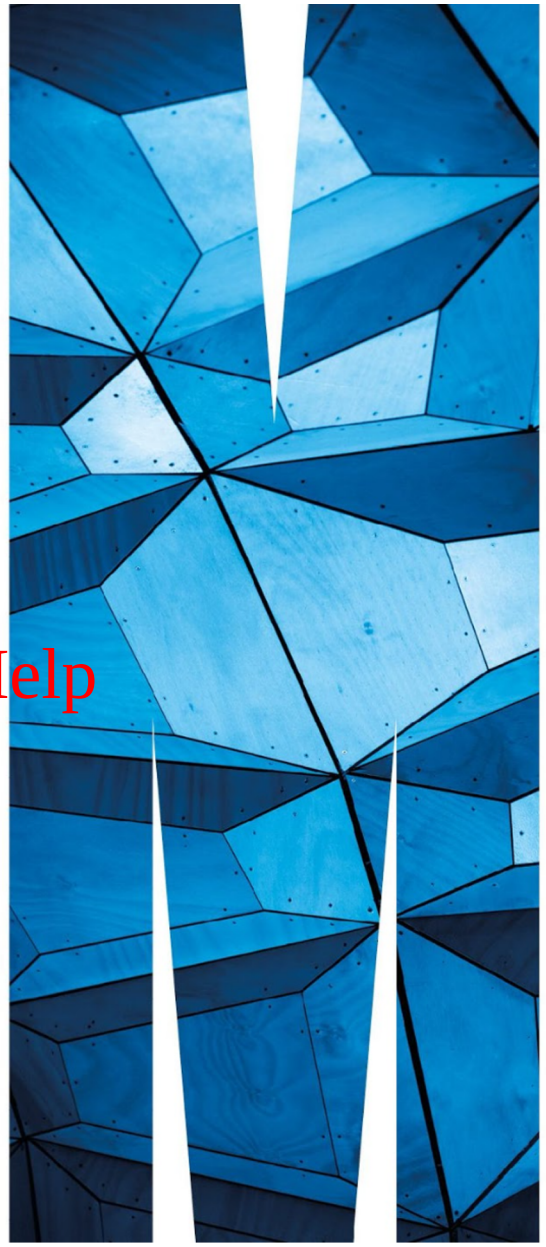
MONASH University

MONASH
INFORMATION
TECHNOLOGY



# Machine Learning: Collaborative Filtering



Prajwol Sangat
Updated by Chee-Ming Ting (30 April 2021)

GROUP OF EIGHT AUSTRALIA

## Last week



Clustering:

- K-Means
- Parallel K-Means

## This week



Collaborative Filterin

# Recommender System



A recommender system makes a prediction based on users' historical behaviours. More specifically, it predicts user preference for a set of items based on past experience. This system is personalizing user's web experience – e.g. telling what to buy (Amazon), which movies to watch (Netflix), whom to be friends with (Facebook), which songs to listen (Spotify) etc.

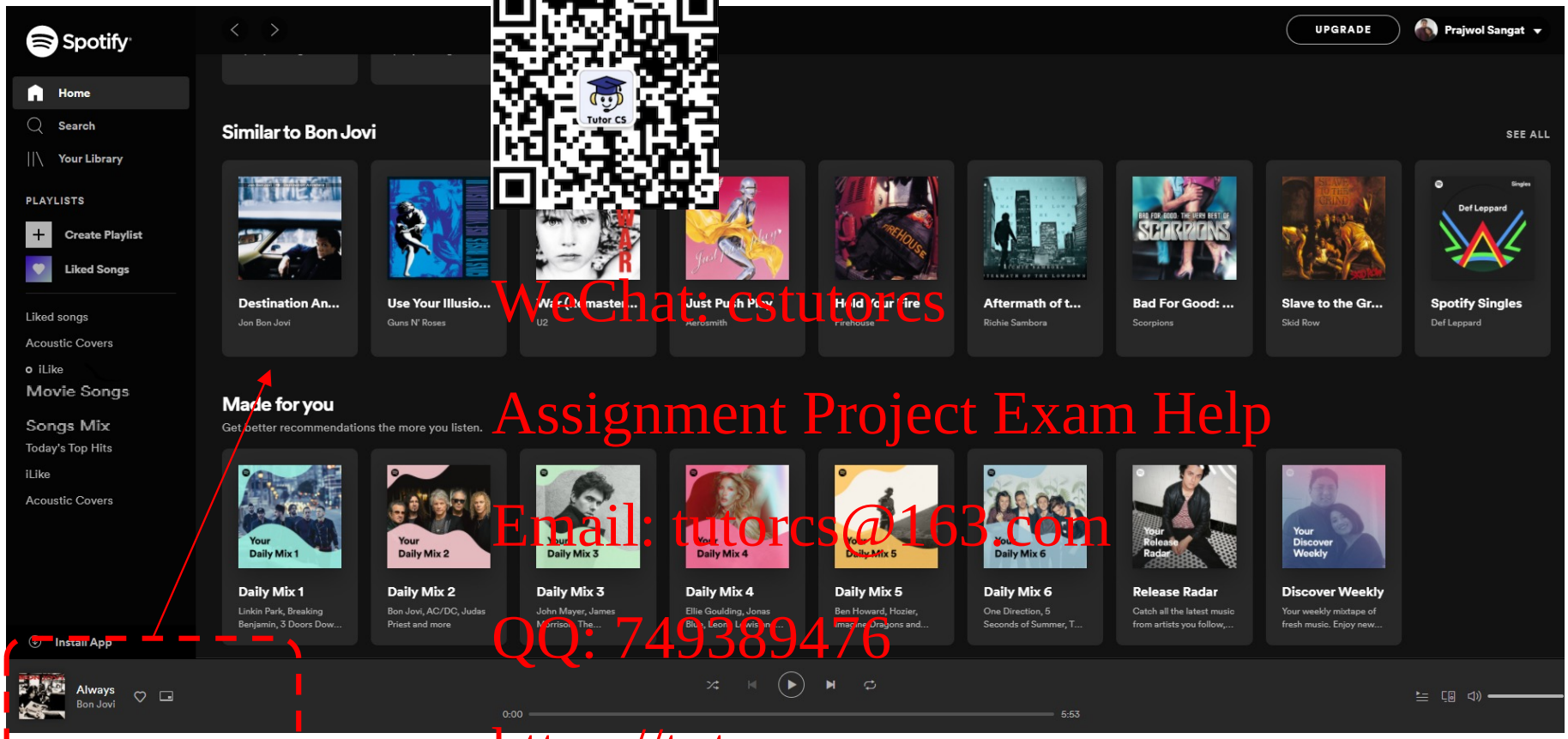


Two common approaches:

- Content based
- **Collaborative Filtering**

# Everyday Examples of Collaborative Filtering





# User feedback



- **Explicit feedback:** Di rences given by the user to the item (e.g., user rating)

- **Implicit feedback :** Indirect feedback, gathered from user behaviour (e.g. number of views, clicks, shares, likes, visited/brought objects etc.).



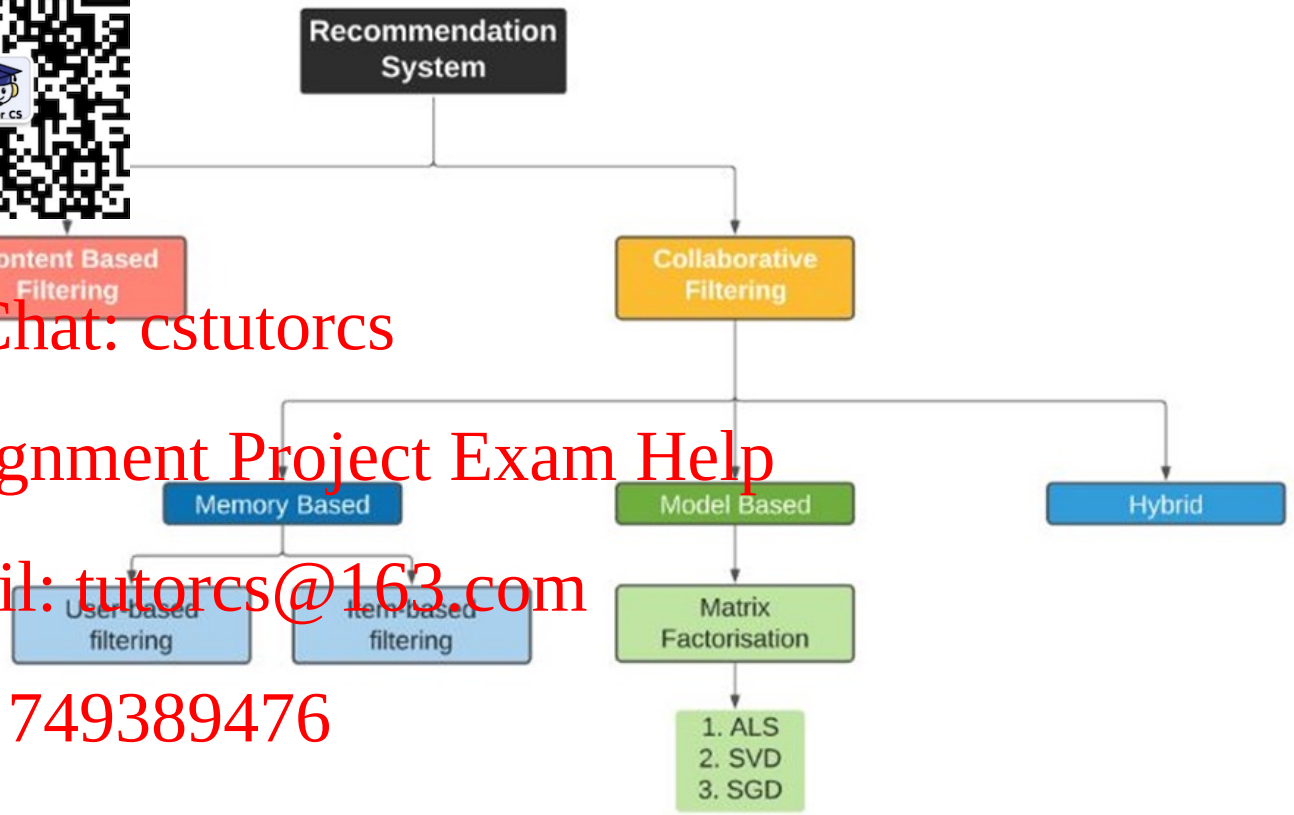



Image from: https://www.slideshare.net/LadislavPeska/using-the-context-of-user-feedback-in-recommender-systems

# Recommender System



- Two common approaches:
  - Content based
  - **Collaborative Filtering**





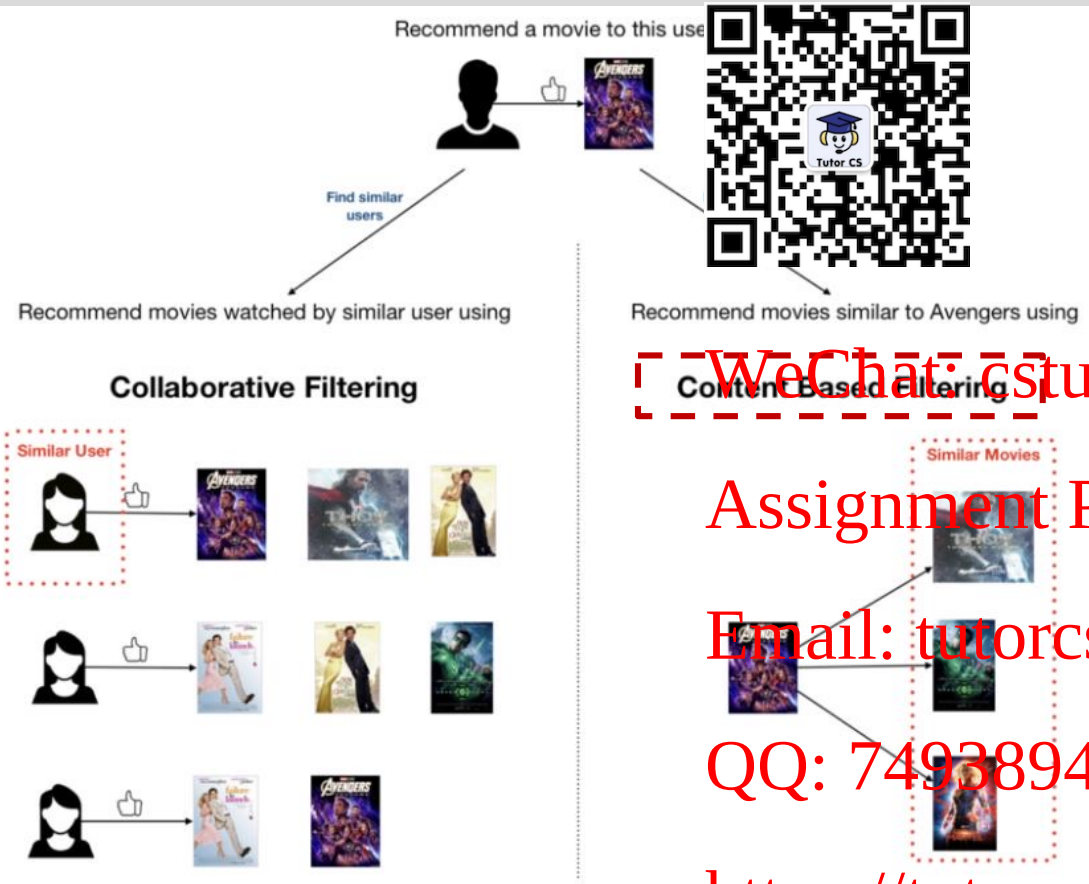

**Collaborative filtering** is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences from many users (collaborating)



https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679

# Content based Filtering






- Main Idea: Recommend items similar to the items previously liked by the user
- Example:
  - **Movie recommendations:** Recommend movies with same actor(s), director, genre.
  - **Websites, blogs, news:** Recommend other sites with "similar" content
- It requires a sufficient amount of information about the content of the items

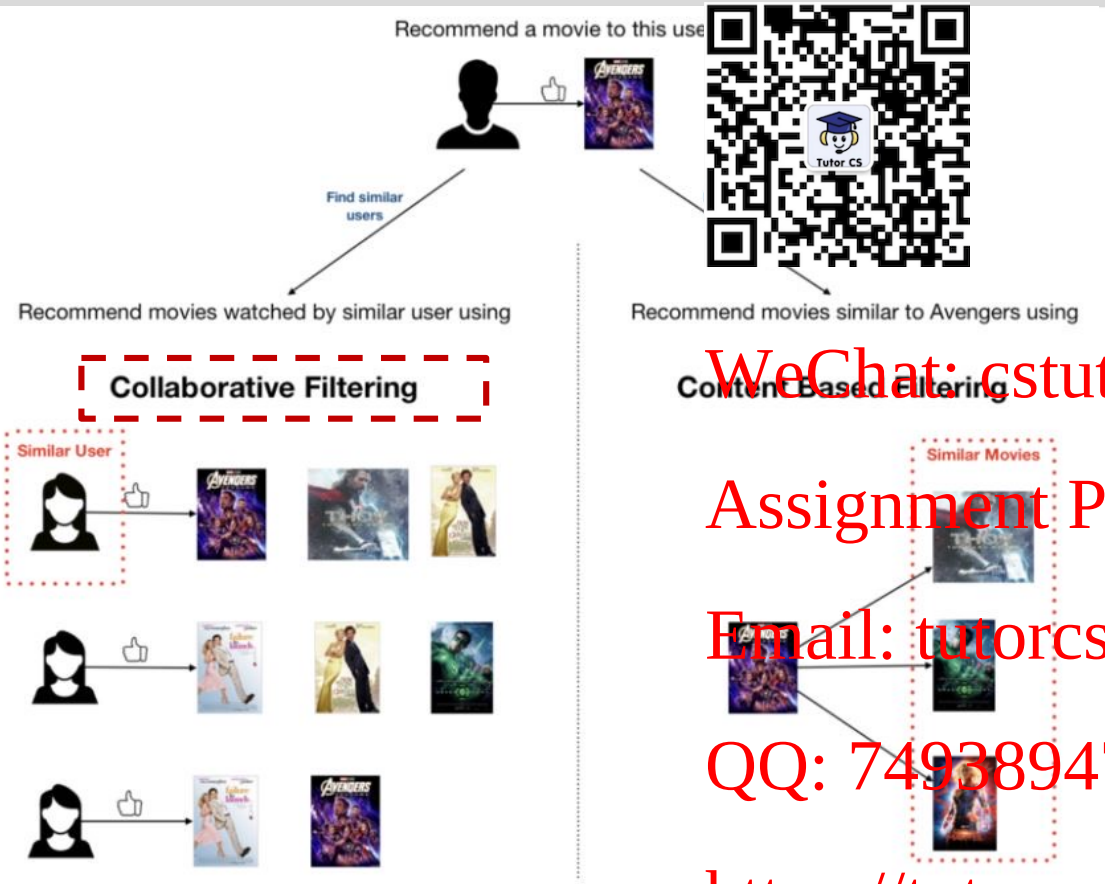




https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679

# Collaborative Filtering (CF)



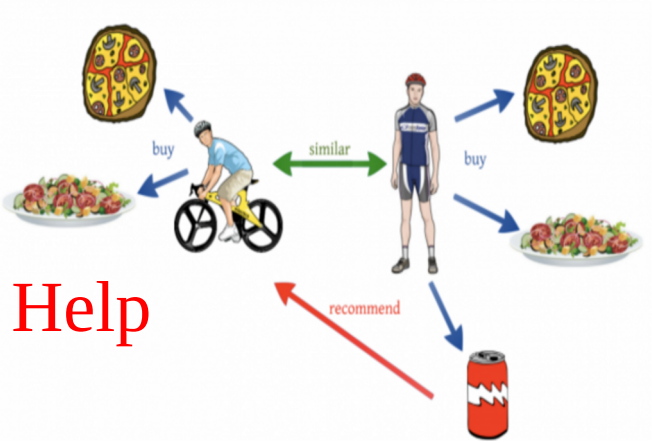



- Main Idea: Use input/behavior of all previous users to make future recommendation
- Recommend items to a user based on the items liked by another set of users whose rating pattern (like & dislike) are similar to the user
- Example:
  - **Movie recommendations:** Recommend movies watched by similar user
- It's domain-free - It does not look at the details of content, only looks at who is rating the content & what is the rating
- Make use of **similarity between users** past feedback/preferences (***user*-based CF**)

# Collaborative Filtering



**Collaborative filtering** is a ma[illegible] method/formula to find the predictions about hov[illegible]ter can rate a particular item by comparing t[illegible] all other users.




*For example*:
To predict *PersonA* rating on a particular item, it would compute the similarity between *PersonA* with all users. Take the top users who are most similar to the *PersonA*, then it would compute the predicted ratings for *PersonA* items with respective to all similar users.

# Why Collaborative Filtering?



- It benefits from large user bases.
- It's flexible across different domains.
- It produces the level of recommendations required.
- It can capture more nuance around items.

# Collaborative Filtering Process


**Data Collection -> Data Processing -> Calculate Referrals -> Derive Results**

- **Data collection**: Collecting user behaviour and associated data items
- **Data processing**: Processing the collected data
- **Recommendation Calculation**: The recommended calculation method used to calculate referrals
- **Derive the result**: Extract the similarity, sort it, and extract the top N to complete

# Memory-based Collaborative Filtering



**Memory-based (neighbourh ch) CF** recommends items by finding similarity between users or items

- **User-based CF**: To recommend items to a user based on another set of users with similar rating pattern to the user
- **Item-based CF**: To recommend items with the most similarity with user's other favourite items.



# Collaborative Filtering – User based

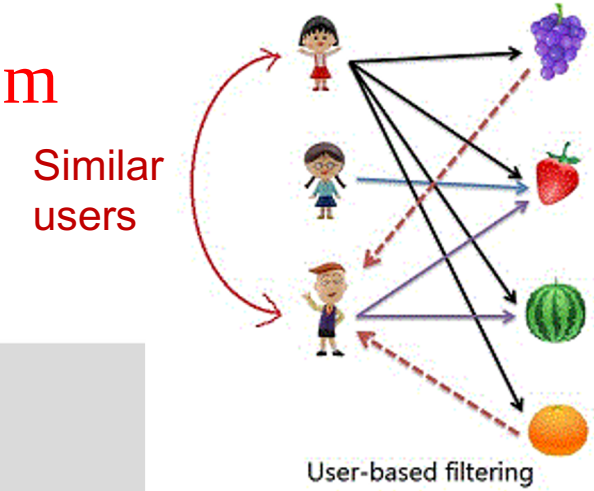


It calculates the **similarity between users** to make implicit recommendation.

Steps:

1. Calculate the similarity between *PersonA* and all other users.
2. Predict the ratings of items for *PersonA* based on similar users.
3. Select top-N rated items for *PersonA*.





# Collaborative Filtering – Item based



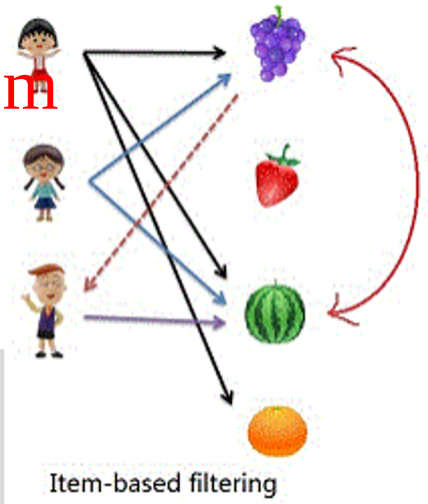
It calculates the **similarity between items** to make implicit recommendation

.

Steps:

1. Calculate the similarity between any two items to get item-item similarity matrix.
2. Predict the ratings of items for *PersonA* based on similar items.
3. Select top-N rated items for *PersonA*.

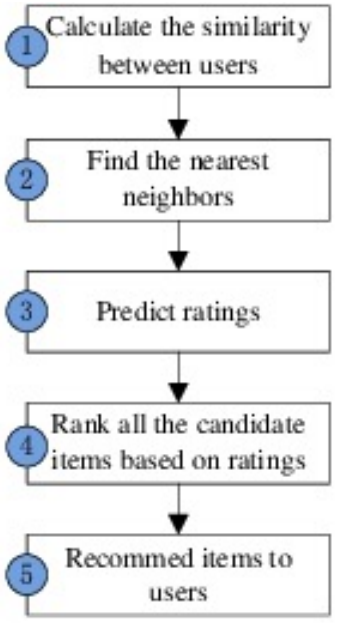




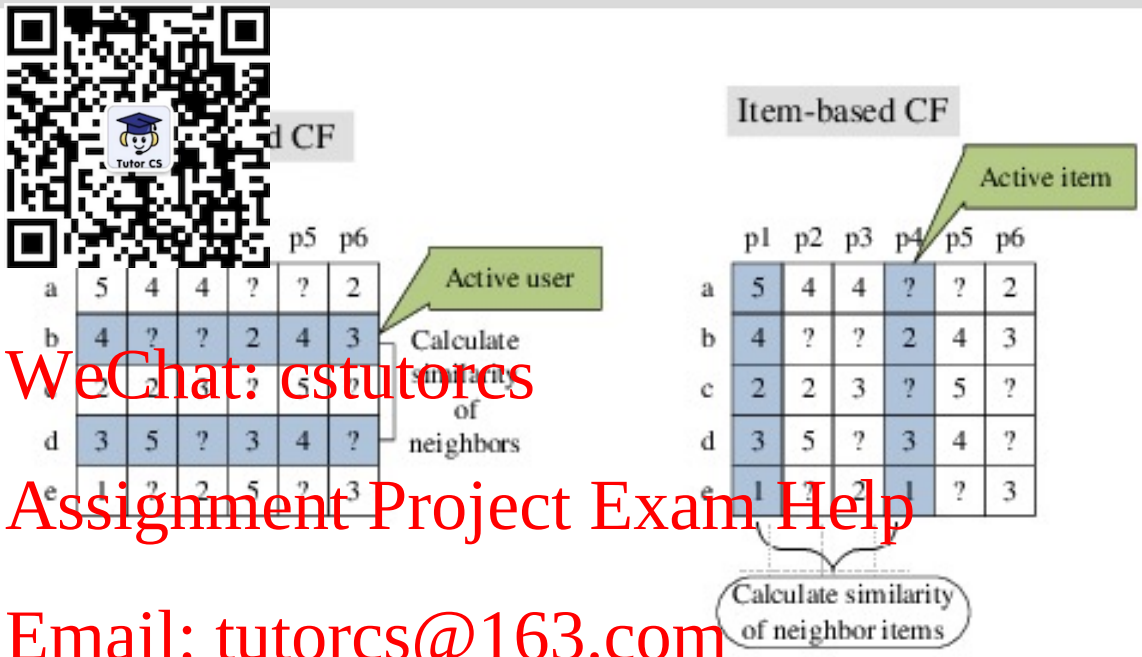

Similar items

# User-Based Vs Item-Based CF






The procedure of memory-based collaborative filtering RS






For both user-based or item-based CF, the computation of similarity is based on the preference for the item.

The features used to calculate similarity can be user's purchase frequency, user's preference rating, number of product clicks, or a combination of all of these.



Adopted from paper 'A Survey of Collaborative Filtering-Based Recommender Systems: From Traditional Methods to Hybrid Methods Based on Social Networks'

# How to get similarity?

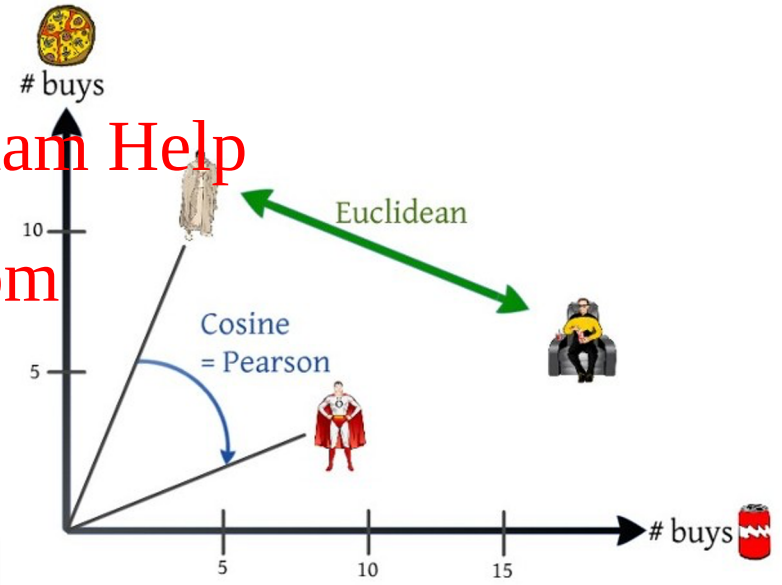


- **Cosine similarity:**
  - Measures the cosine of the [illegible] between ratings of two users
  - If cosine value is 1, two users are completely similar in their preference, if cosine value is -1, they are completely dissimilar
  - Similar to Pearson correlation, which measures correlation between two users

$$sim(u, u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\| \|\mathbf{r}_{u'}\|}$$



- **Euclidean distance:**
  - Measures distance in rating/preference between two users
  - If the distance is small, two users have a similar preference (i.e., the similarity is high).



# Collaboration Filtering: Walkthrough Example

| Name | Star Trek | [illegible] | Superman | Batman | Hulk |
|---|---|---|---|---|---|
| Harry | 4 | 2 | ? | 5 | 4 |
| John | 5 | 3 | 4 | ? | 3 |
| Rob | 3 | 2 | 4 | 4 | 3 |



Aim: Recommend top-2 movies to Harry

# Collaboration Filtering: Walkthrough Example (user-based)

Step 1: Calculate the si[illegible]etween Harry and all other users

| Name | Star Trek | [illegible]s | Superman | Batman | Hulk |
|---|---|---|---|---|---|
| Harry | 4 | 2 | ? | 5 | 4 |
| John | 5 | 3 | 4 | ? | 3 |
| Rob | 3 | 2 | 4 | 4 | 3 |



Cosine similarity

$$sim(u,u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\|\|\mathbf{r}_{u'}\|} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2}\sqrt{\sum_i r_{u'i}^2}}$$

$r_{ui}$ - value of ratings user *u* gives to item *i*

# Collaboration Filtering: Walkthrough Example (user-based)

Step 1: Calculate the similarity between Harry and all other users

| Name | Star Trek | Star wars | Sup | | Hulk |
|---|---|---|---|---|---|
| Harry | 4 | 2 | ? | 5 | 4 |
| John | 5 | 3 | 4 | ? | 3 |
| Rob | 3 | ? | 4 | 4 | 3 |

$$sim(u, u') = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

$r_{ui}$ - value of ratings user *u* gives to item *i*

Cosine similarity

$$\text{Sim(Harry, John)} = \frac{(4*5)+(2*3)+(4*3)}{sqrt(4^2+2^2+4^2)*sqrt(5^2+3^2+3^2)} = 0.97$$

$$\text{Sim(Harry, Rob)} = \frac{(4*3)+(5*4)+(4*3)}{sqrt(4^2+5^2+4^2)*sqrt(3^2+4^2+3^2)} = 1.00$$

# Collaboration Filtering: Walkthrough Example (user-based)

Step 2: Predict the ratings c[illegible]or Harry

| Name | Star Trek | Star wars | Superm | | Hulk |
|---|---|---|---|---|---|
| Harry | 4 | 2 | ? | 5 | 4 |
| John | 5 | 3 | 4 | ? | 3 |
| Rob | 3 | ? | 4 | 4 | 3 |

Predicted rating is calculated based on aggregation of some similar users' rating of the item

$$r_{u,i} = k \sum_{u' \in U} \text{simil}(u, u') r_{u',i}$$

with normalising factor

$$k = 1 / \sum_{u' \in U} |\text{simil}(u, u')|,$$

Calculate *k* as a normalising factor $k = \frac{1}{(0.97+1)} = 0.51$

R(Harry, Superman) = k*$((sim(Harry, John) * R(John, Superman)) + (sim(Harry, Rob) * R(Rob, Superman)))$

$= 0.51((0.97 * 4) + (1 * 4)) = 4.02$

## Collaboration Filtering: Walkthrough Example (user-based)

Step 3: Select top-2 ra[illegible]es for Harry

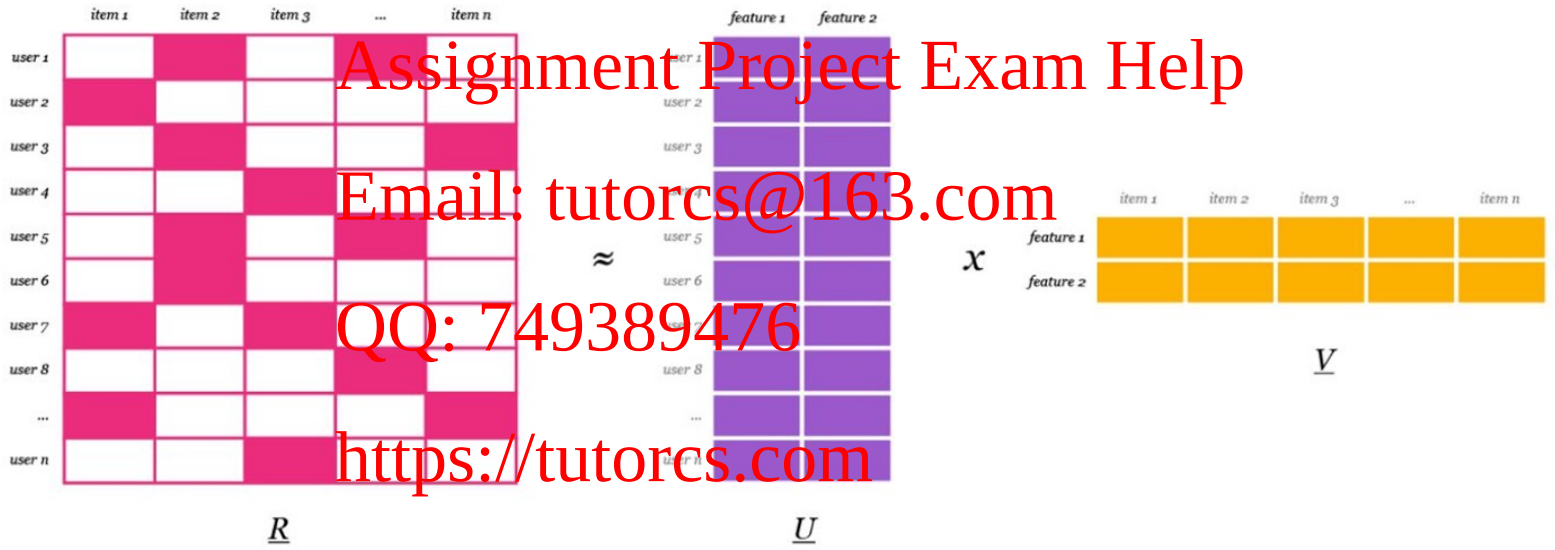
| Name | Star Trek | [illegible] | Superman | Batman | Hulk |
|---|---|---|---|---|---|
| Harry | 4 | | ***4.02*** | 5 | 4 |
| John | 5 | 3 | 4 | ? | 3 |
| Rob | 3 | ? | 4 | 4 | 3 |

*Top-2(Harry, movies)*= Batman, Superman

# Model-based Collaborative Filtering

- Latent factor model based CF (latent) user and item profiles through **matrix factorization**
- **Matrix factorization:** Factor a matrix into some smaller representation of the original matrix through alternating least squa product of lower dimensional matrices equals the original one
- Example: Factor the rating matrix **R** into user matrix **U** and item matrix **V**





# Alternating least squares



- ALS method aims to estimate th…em factor matrices (**U & V**) such tha…t will approximate the original rating …
- This is achieved by minimizing …square error (RMSE) between the original ratings and the predicted values

**ALS Procedure:**

**Optimizing alternately to find U, V**

- Randomly initialize U and V
- Iterating the following steps:
  - Fixing U → Optimizing V
  - Fixing V → Optimizing U
- Each iteration will minimize the square errors


Rating matrix, **R** | User factor matrix, **U** | Item factor matrix, **V** | Predicted rating matrix $\hat{R}$





https://developers.google.com/machine-learning/recommendation/collaborative/matrix

arthouse <-> blockbuster
preference for arthouse <-> blockbuster
children's <-> adult's
preference for children's <-> adult's

# Collaborative filtering in Spark



- `spark.ml` currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries.
- `spark.ml` uses the Alternating Least Squares (ALS) algorithm to learn these latent factors.

# Demo



MovieLens dataset consisting movie, a rating and a timestamp.

Lets train an **ALS model** whic by default, that the ratings are explicit.

And evaluate the recommendation model by measuring the root-mean-square error of rating prediction.

## What have we learnt today?

Collaborative Filteri
Walkthrough examp mplementation in Apache Spark with Python