

# Machine Learning: Collaborative Filtering

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



# Last week

Clustering:

- K-Means
- Parallel K-Means

# This week

- Collaborative Filtering

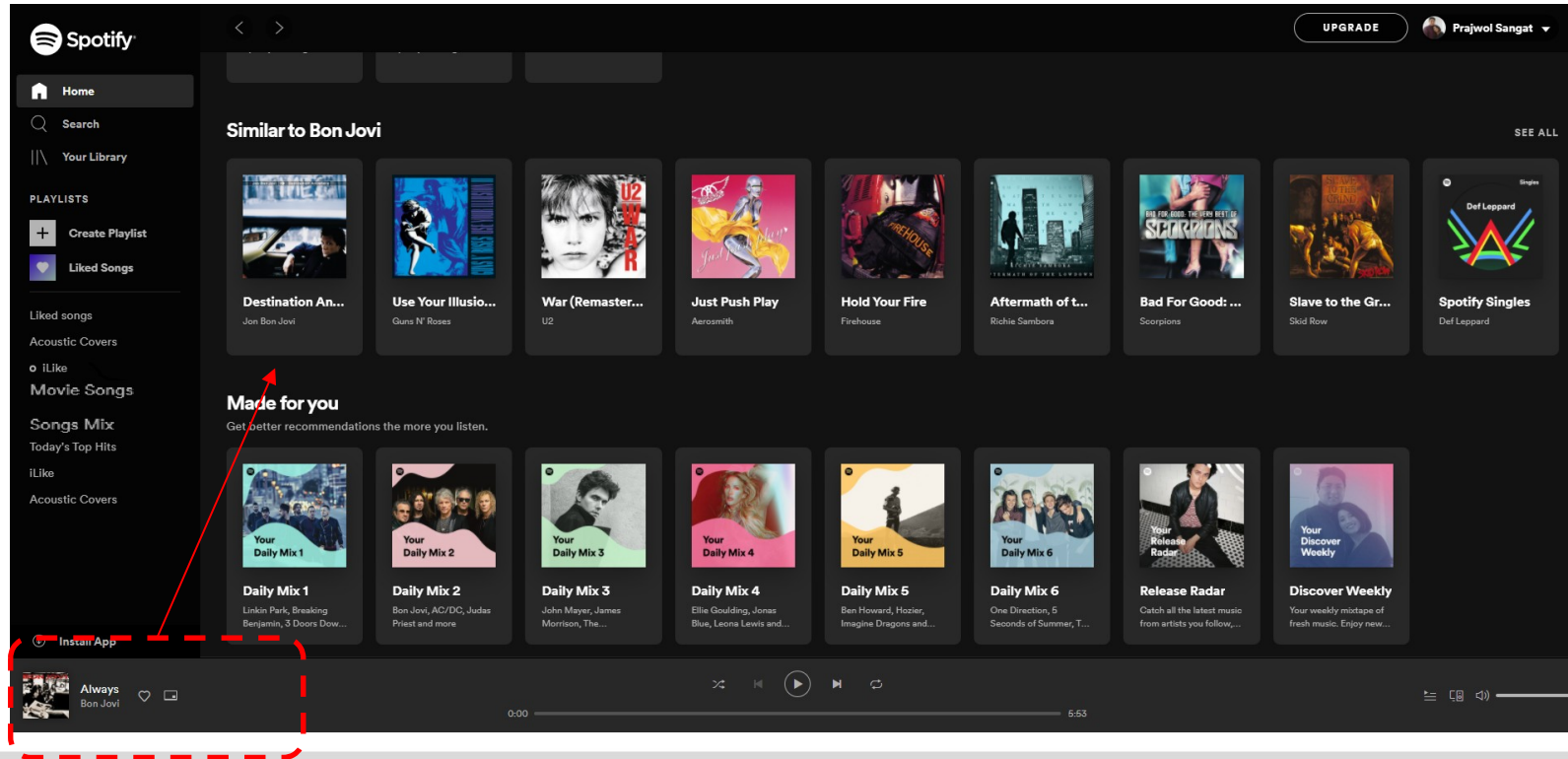
# Recommender System

A recommender system makes 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:

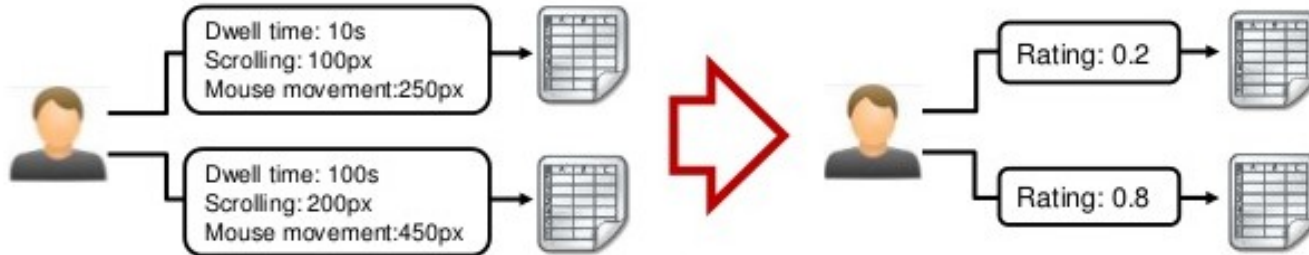
- Content based
- **Collaborative Filtering**

# Everyday Examples of Collaborative Filtering



# User feedback

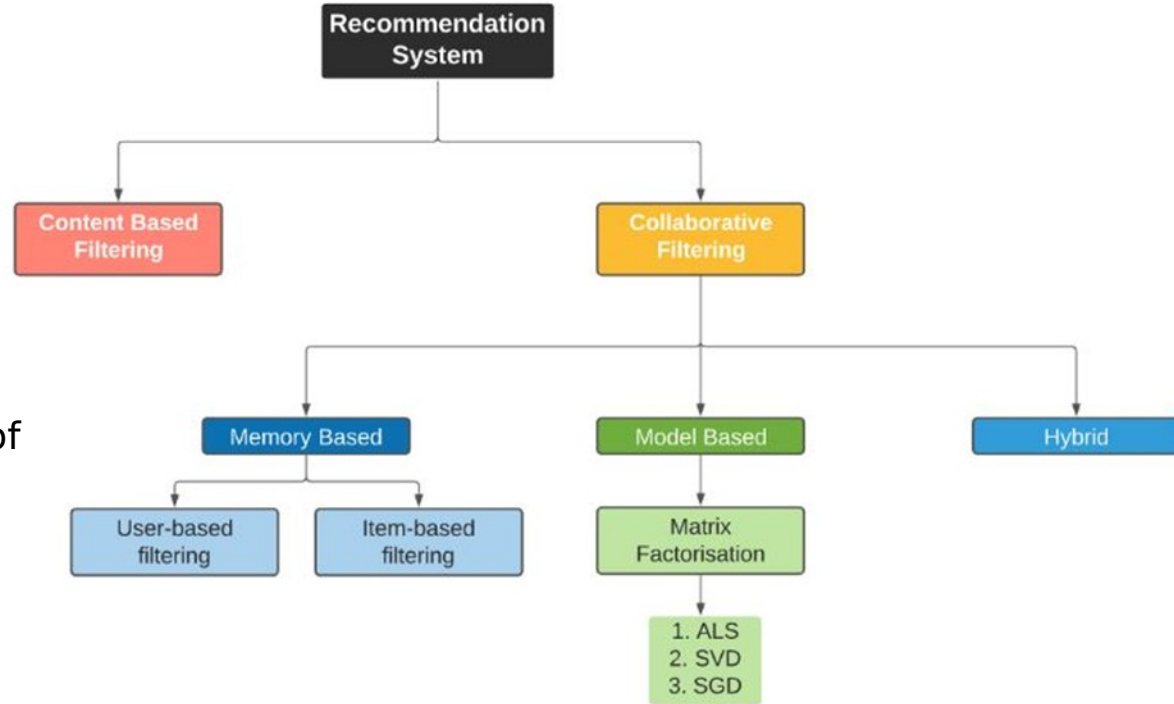
- **Explicit feedback:** Direct preferences given by the user to the item (e.g., user rating) 
- **Implicit feedback:** Indirect feedback, gathered from user behaviour (e.g. number of views, clicks, shares, likes, visited/brought objects etc.).



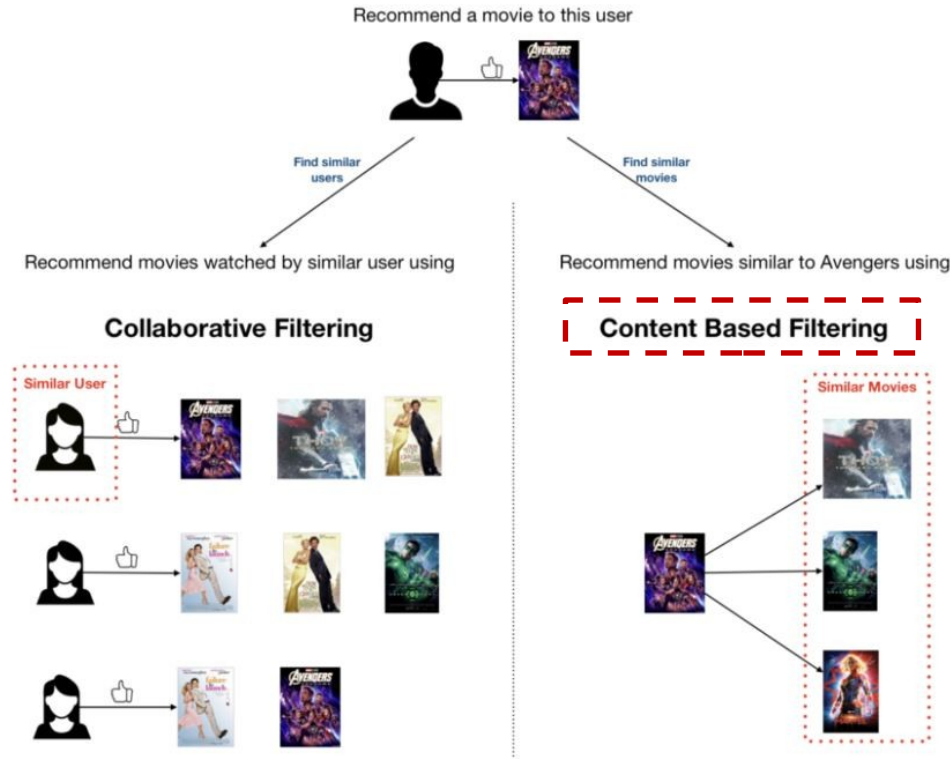
# Recommender System

- 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)



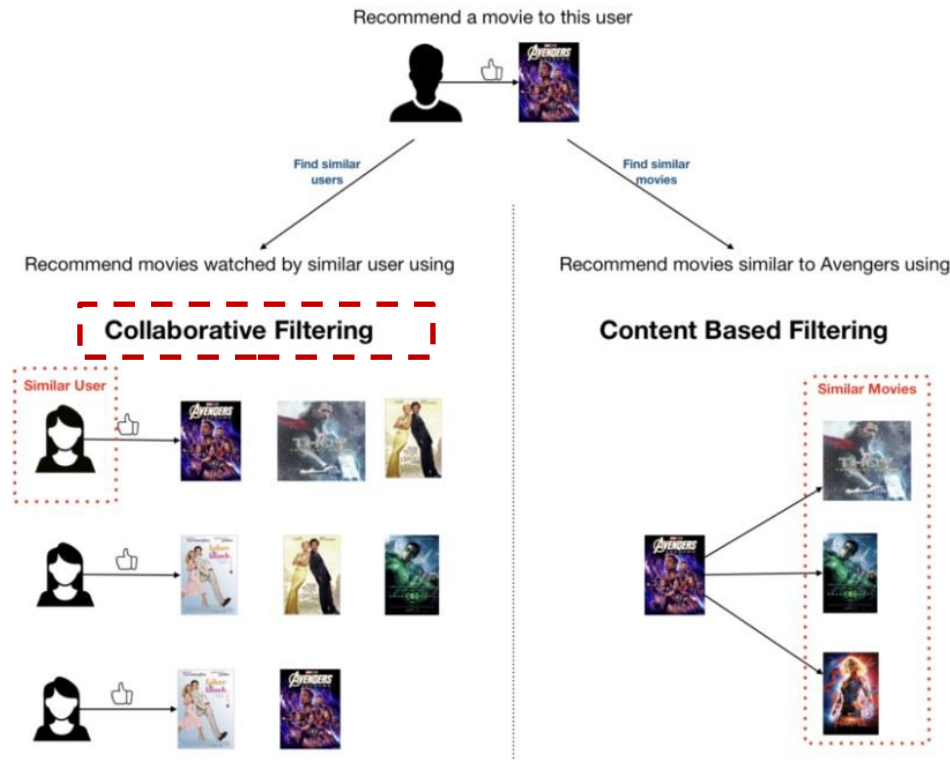
# Content based Filtering



- ❑ **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



# Collaborative Filtering (CF)



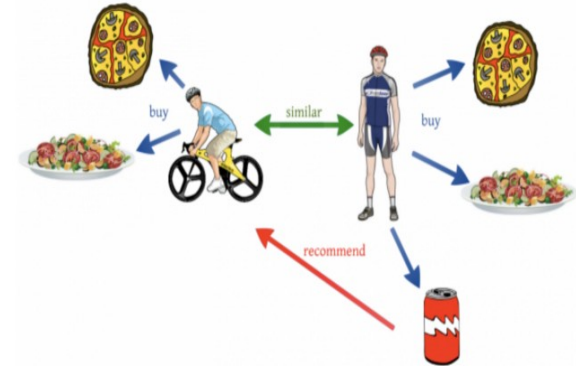
- ❑ **Main Idea:** Use input/behavior of all previous users to make future recommendation
- ❑ Recommend items to a user based on the items liked by **another set of users whose rating pattern (like & dislike) are similar to the user**
- ❑ Example:
  - **Movie recommendations:** Recommend movies watched 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 mathematical method/formula to find the predictions about how much a user can rate a particular item by comparing that user to 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 respect 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

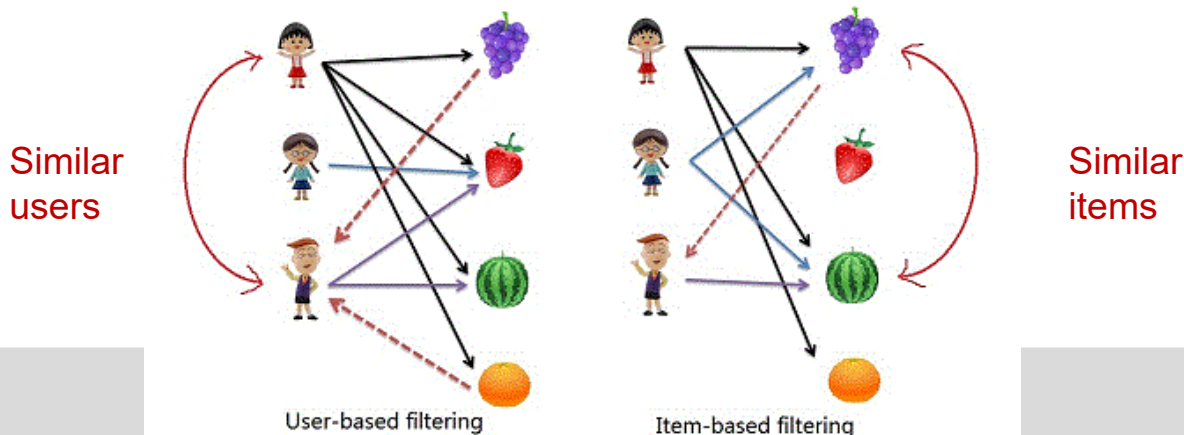
**Data Collection → Data Processing → Calculate Referrals → Derive Results**

- **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 (neighbourhood approach) 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

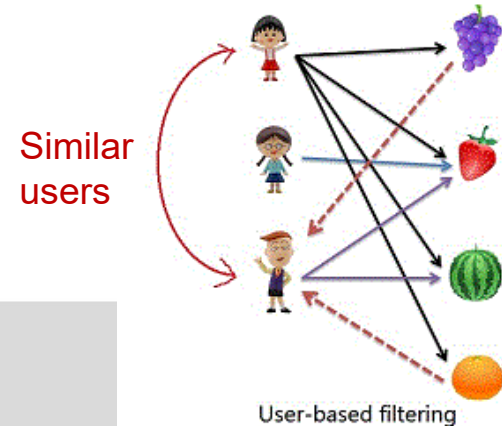


# Collaborative Filtering – User based

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 ' '
3. Select top-N rated items for *PersonA*.



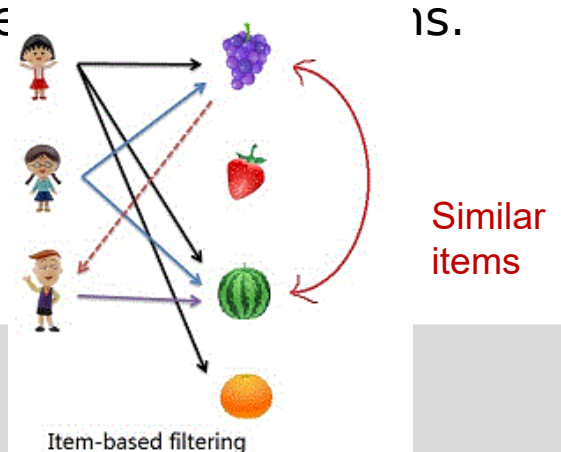
# Collaborative Filtering – Item based

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

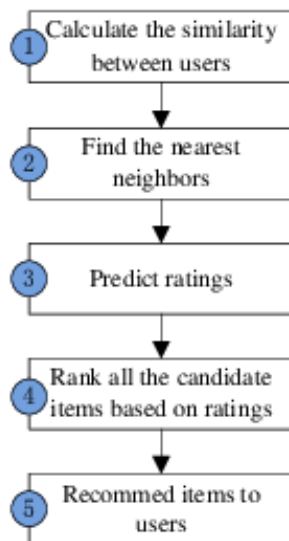
.

Steps:

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



# User-Based Vs Item-Based CF



User-based CF

	p1	p2	p3	p4	p5	p6
a	5	4	4	?	?	2
b	4	?	?	2	4	3
c	2	2	3	?	5	?
d	3	5	?	3	4	?
e	1	?	2	5	?	3

Active user

Calculate similarity of neighbors

Item-based CF

	p1	p2	p3	p4	p5	p6
a	5	4	4	?	?	2
b	4	?	?	2	4	3
c	2	2	3	?	5	?
d	3	5	?	3	4	?
e	1	?	2	1	?	3

Active item

Calculate similarity of neighbor items

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.



# How to get similarity?

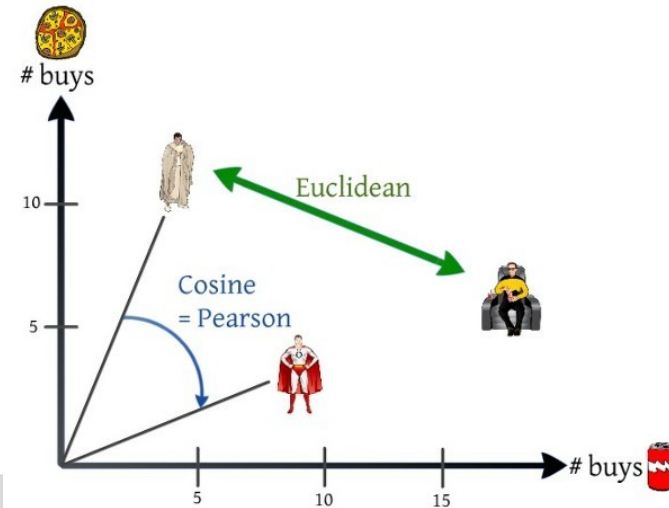
## ❑ Cosine similarity:

- Measures the cosine of the vector angle 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

## ❑ 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).

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



# Collaboration Filtering: Walkthrough Example

Name	Star Trek	Star wars	Superman	Batman	Hulk
Harry	4	2	?	5	4
John	5	3	4	?	3
Rob	3	?	4	4	3

Aim: Recommend top-2 movies to Harry

# Collaboration Filtering: Walkthrough Example (user-based)

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

Name	Star Trek	Star wars	Superman	Batman	Hulk
Harry	4	2	?	5	4
John	5	3	4	?	3
Rob	3	?	4	4	3

Cosine similarity

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

$r_{ui}$  - 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	Superman	Batman	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

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

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

# Collaboration Filtering: Walkthrough Example (user-based)

## Step 2: Predict the ratings of movies for Harry

Name	Star Trek	Star wars	Superman	Batman	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  $k = \frac{1}{(0.97+1)} = 0.51$

$$\begin{aligned} R(\text{Harry}, \text{Superman}) &= k * ((\text{sim}(\text{Harry}, \text{John}) * R(\text{John}, \text{Superman})) + (\text{sim}(\text{Harry}, \text{Rob}) * R(\text{Rob}, \text{Superman}))) \\ &= 0.51((0.97 * 4) + (1 * 4)) = 4.02 \end{aligned}$$

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

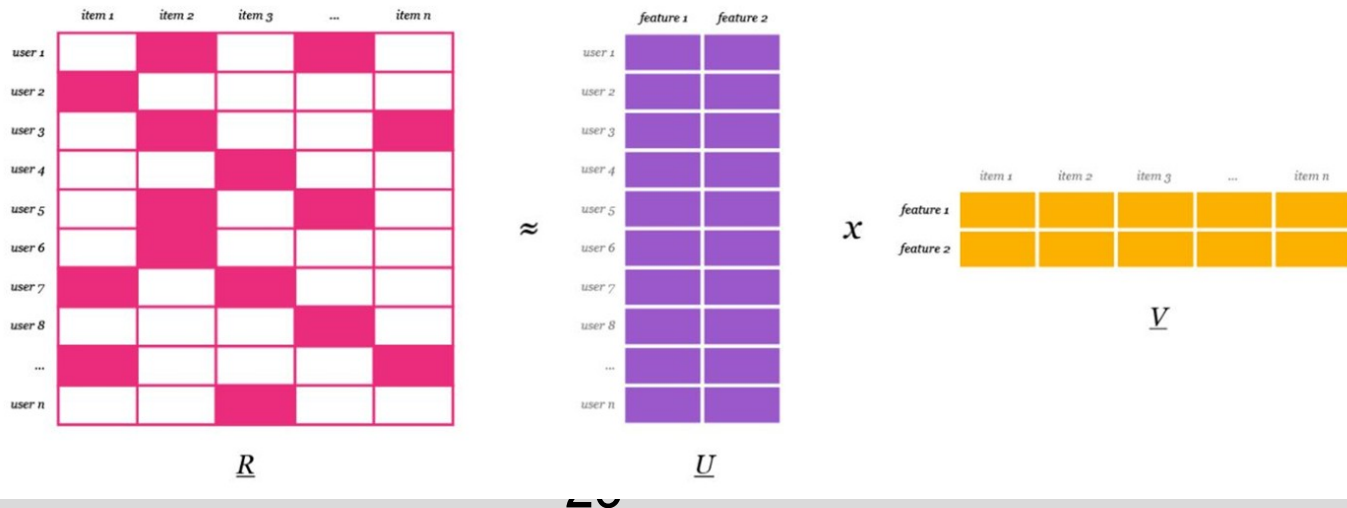
Step 3: Select top-2 rated movies for Harry

Name	Star Trek	Star wars	Superman	Batman	Hulk
Harry	4	2	<b>4.02</b>	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 learns the (latent) user and item profiles through **matrix factorization**
- ❑ **Matrix factorization:** Factor a large matrix into some smaller representation of the original matrix through alternating least squares. The 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 the user and item factor matrices (**U** & **V**) such that their product will approximate the original rating matrix **R**.
- This is achieved by minimizing root mean square error (RMSE) between the original ratings and the predicted values

	✓		✓	✓	
		✓			✓
	✓	✓	✓		
				✓	✓

Rating matrix, **R**

	1	.1
	-1	0
	.2	-1
	.1	1

User factor matrix, **U**



	.9	-1	1	1	-.9
	-.2	-.8	-1	.9	1

Item factor matrix, **V**



	.88	-1.08	0.9	1.09	-0.8
	-0.9	1.0	-1.0	-1.0	0.9
	0.38	0.6	1.2	-0.7	-1.18
	-0.11	-0.9	-0.9	1.0	0.91

Predicted rating matrix  **$\hat{R}$**

## 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



# 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 of a user, a movie, a rating and a timestamp.

Lets train an **ALS model** which assumes, 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 Filtering
- Walkthrough examples and implementation in Apache Spark with Python