

# Introduction to Recommendation Engine

Jeff Liu and Gary Chen  
2019/01/12



# Outline

- Intro. to Recommendation Engine
  - ◆ Intro. to Collaboratory
  - ◆ Lab 1: Memory-based Collaborative Filtering
- Intro. to Deep Neural Networks (DNN) & Tips for Optimization
  - ◆ Lab 2: DNN Practice
- Recommendation with Matrix Factorization
  - ◆ Lab 3: Basic Matrix Factorization
  - ◆ Lab 4: Matrix Factorization with User Histories
- Recommendation with Matrix Factorization and DNN
  - ◆ Lab 5: Matrix Factorization using DNN

# Notes and Course Materials

**[goo.gl/vmJGJ4](http://goo.gl/vmJGJ4)**

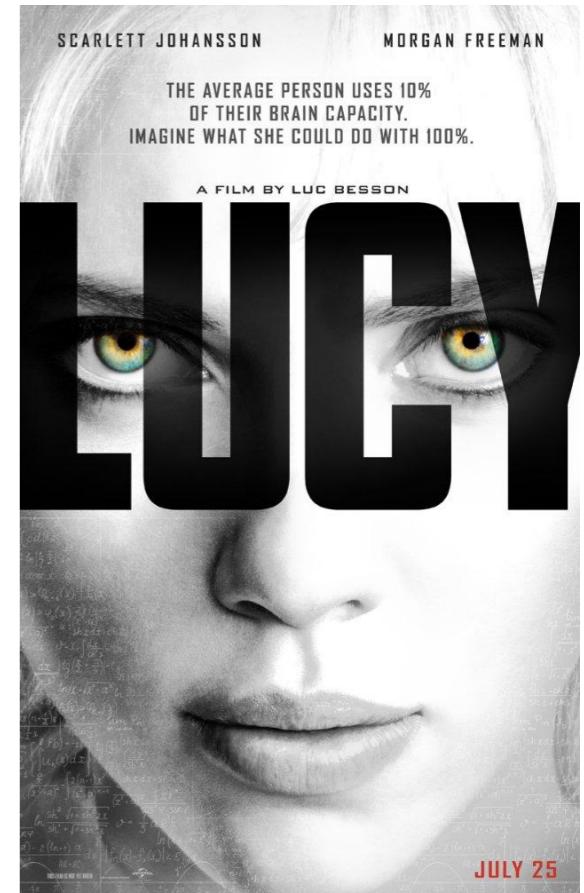
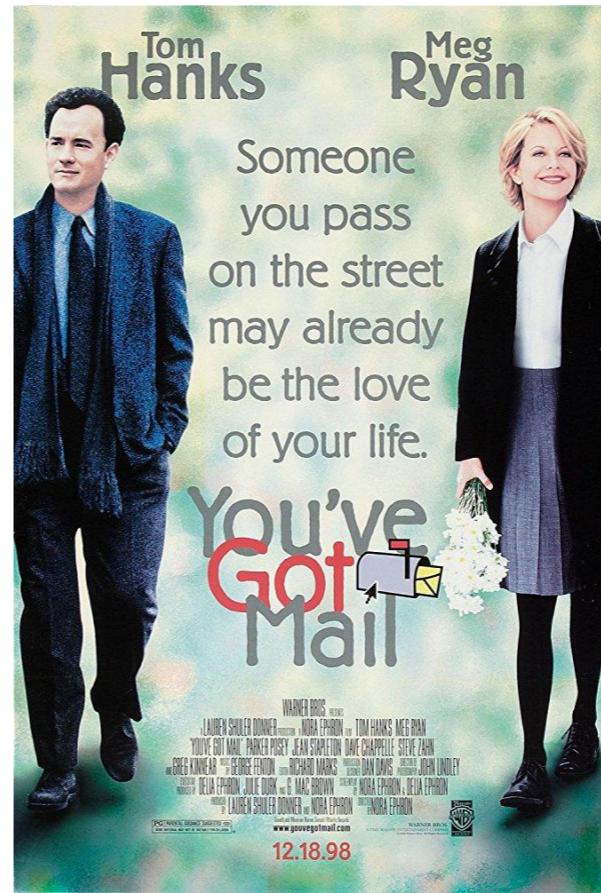
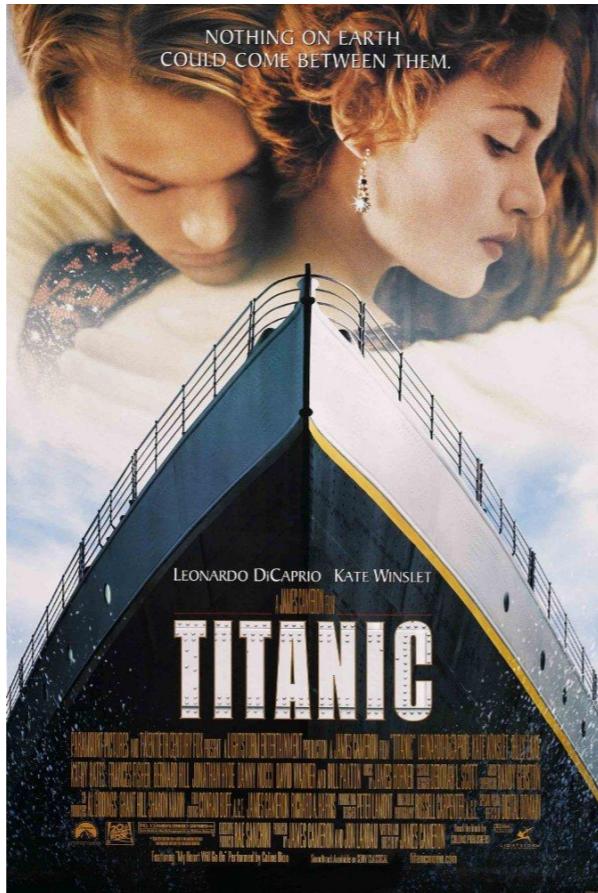
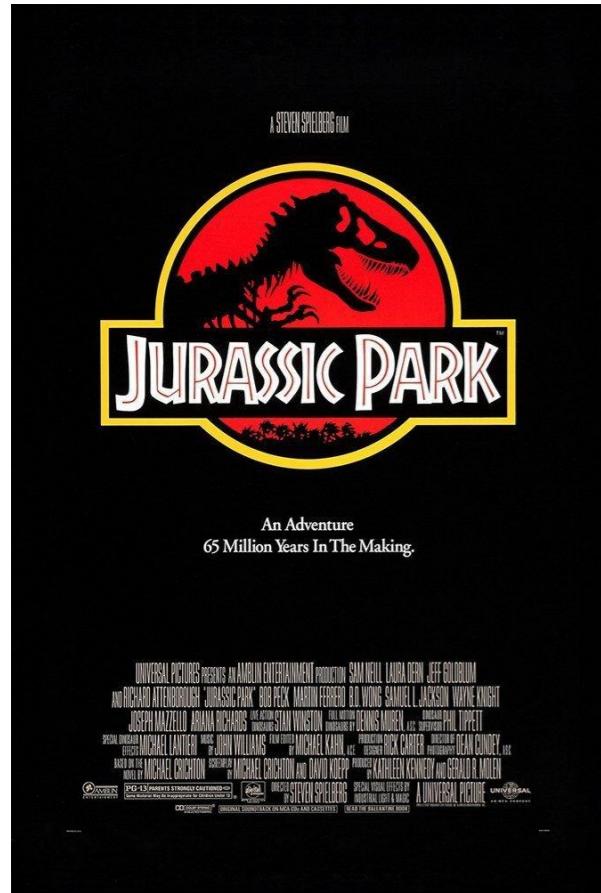
# Prologue

- People are increasing the reliance on conveniences such as e-commerce store or streaming entertainment.
- “Guess” what the customers may like in advance, so to promote more things to sell and in turn generate more revenue.
- A recommendation engine (RE) is any kind of model that can infer the relationship between users/items and make proper prediction for the users.



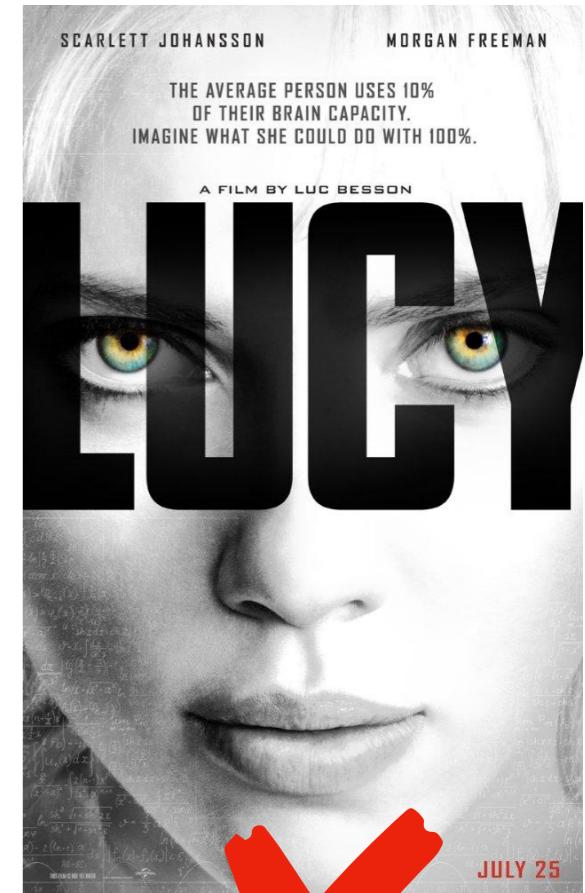
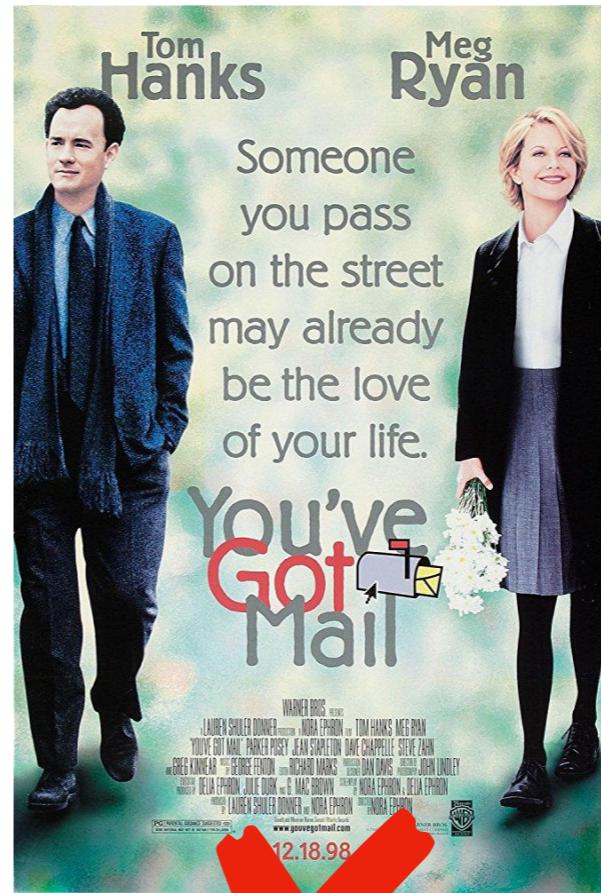
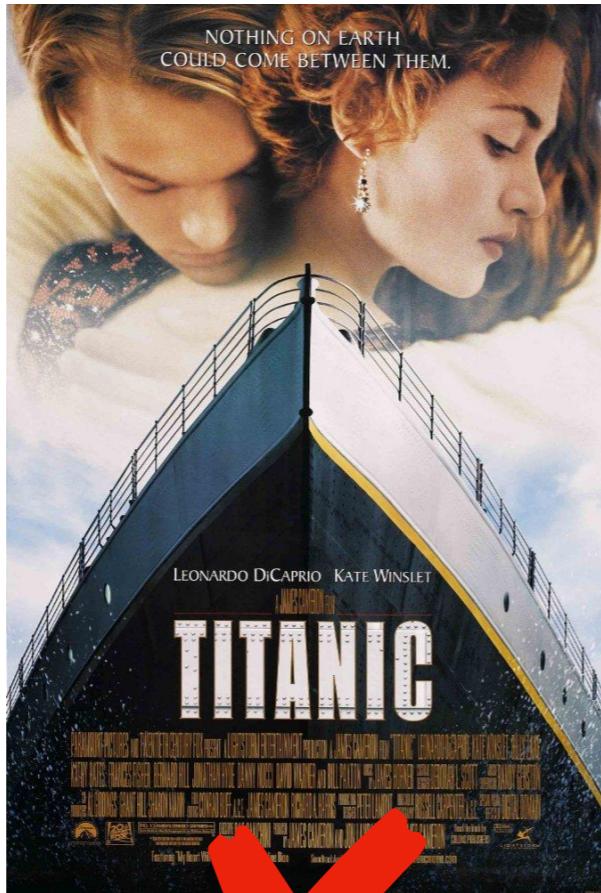
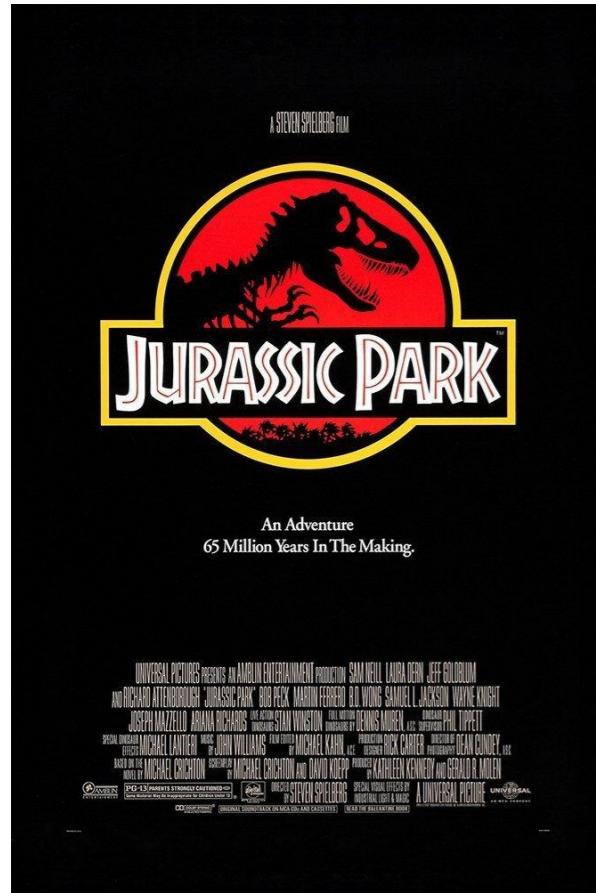


Gender	Male
Age	30
Prefer genre	Sci-fi, comedy, action
Prefer director	Steven Spielberg, Christopher Nolan, Michael Bay, James Cameron
Prefer actor/actress	Tom Hanks, Leonardo DiCaprio, Anne Hathaway, Scarlet Johansson





Gender	Male
Age	30
Prefer genre	Sci-fi, comedy, action
Prefer director	Steven Spielberg, Christopher Nolan, Michael Bay, James Cameron
Prefer actor/actress	Tom Hanks, Leonardo DiCaprio, Anne Hathaway, Scarlet Johansson

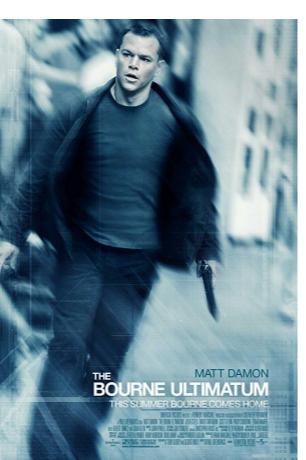
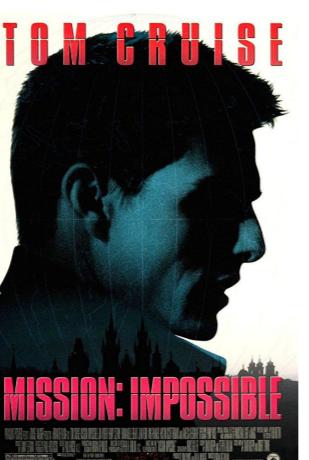
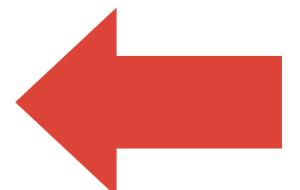
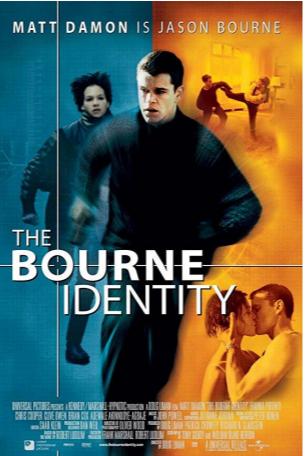
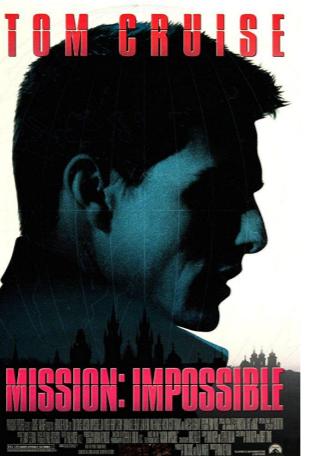
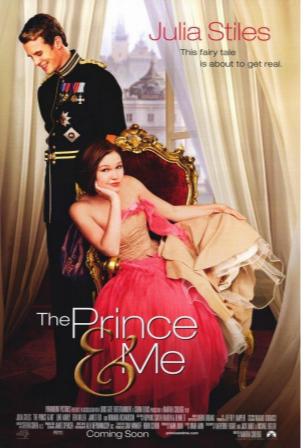
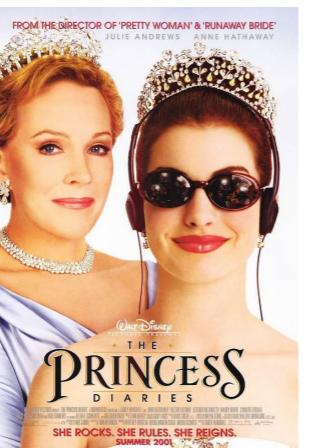


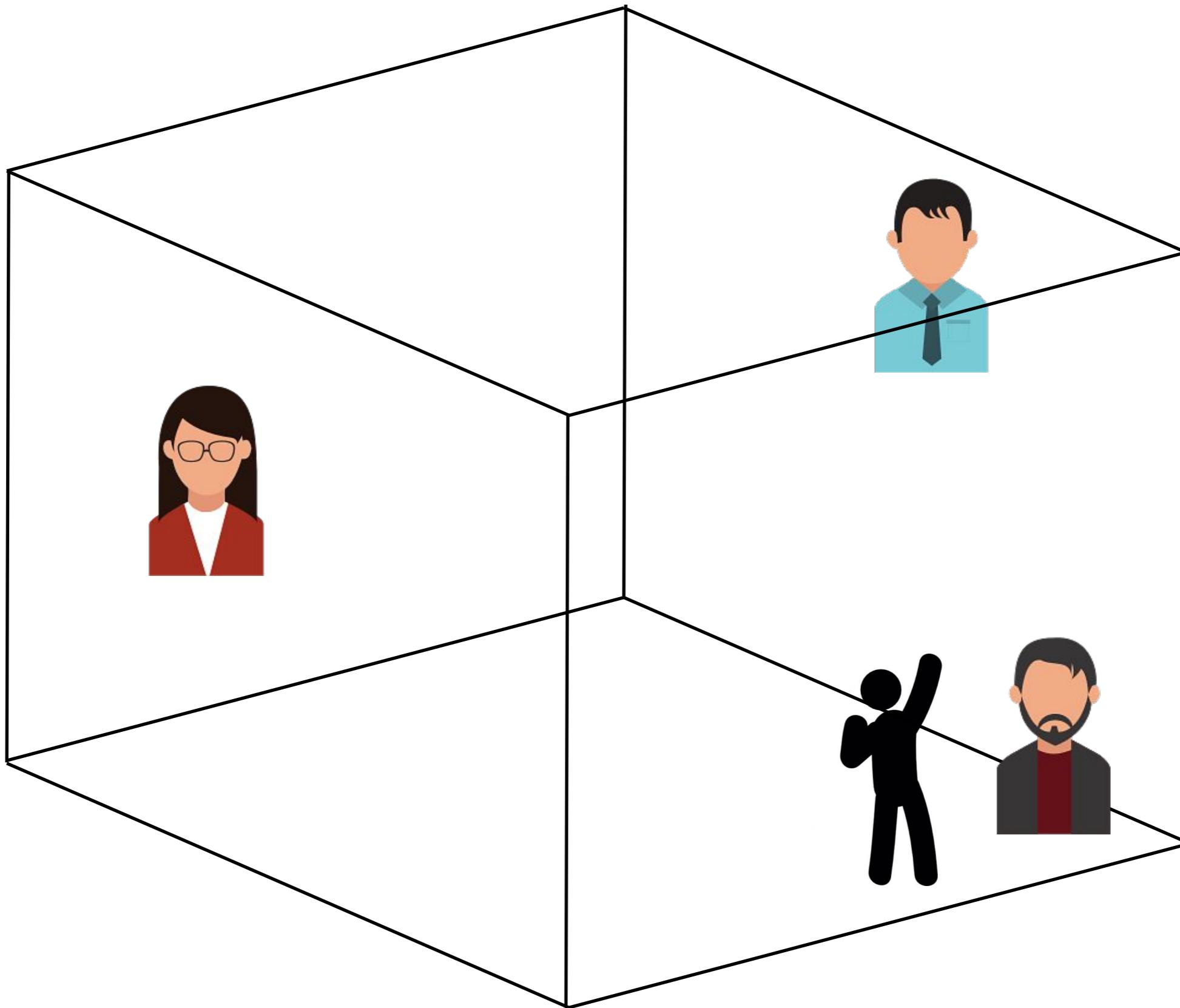
# Content Filtering

- What we have just conducted is essentially one way of doing recommendation: *content filtering*.
- By **building profiles** for both the users and movies, we can provide recommendation by matching the **content** between the two groups, hence the name.
- Take a lot of efforts to build such profiles and many conditional settings to fit a person's taste. 

# Strategies for Recommendation

- Beside *content filtering* we had just mentioned, there is another method called *collaborative filtering (CF)*.
- *CF* relies on past user behavior among a group of users (hence *collaborative*), so we aren't required to create profiles explicitly.
- The two primary areas of *CF* are the *neighborhood methods* (*memory-based*) and *latent factor models*.





# User-based and Item-based CF

- For neighborhood methods, we have two primary types of algorithms: **user-based** and **item-based**.
- In *user-based CF*, we group together users who gave similar ratings to the same set of items, whereby we could later use the ratings of a specific user to predict those of his/her peers.
- For *item-based CF*, we group the items instead.



8.7	10	10	9	?	?	?	9	8	8.5
-----	----	----	---	---	---	---	---	---	-----



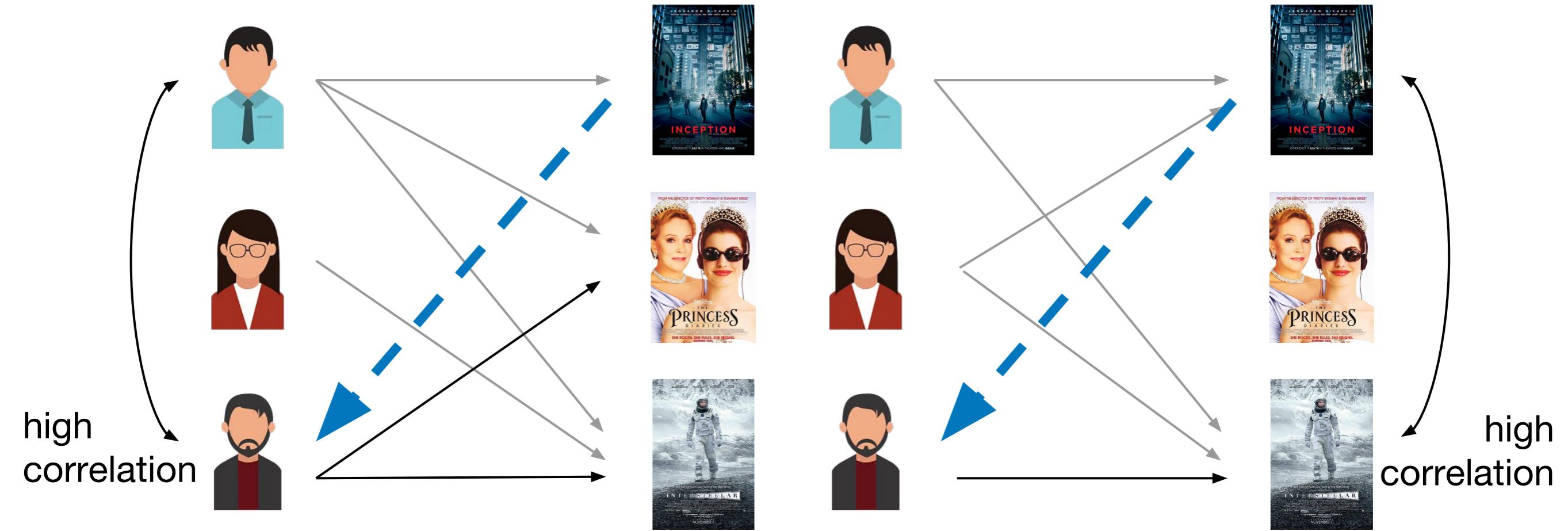
?	?	?	1	10	10	10	?	?	2
---	---	---	---	----	----	----	---	---	---



8	9	9	9	?	?	?	10	10	9.5
---	---	---	---	---	---	---	----	----	-----

--- User-based CF

--- Item-based CF



## User-based filtering

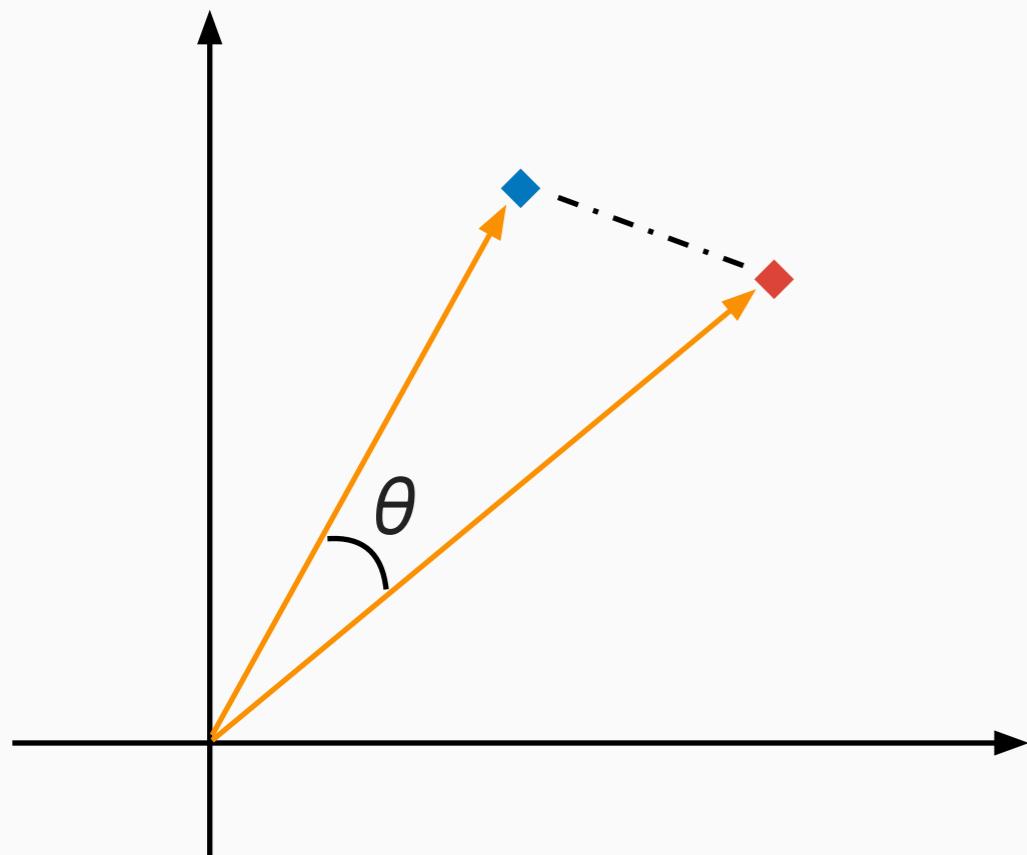
You may like it because your “friends” liked it.

## Item-based filtering

You tend to like that item since you have liked those items.

# Measuring Similarity

- To measure the similarity between users or items, we can use metrics like **Euclidean distance** or **cosine similarity**.



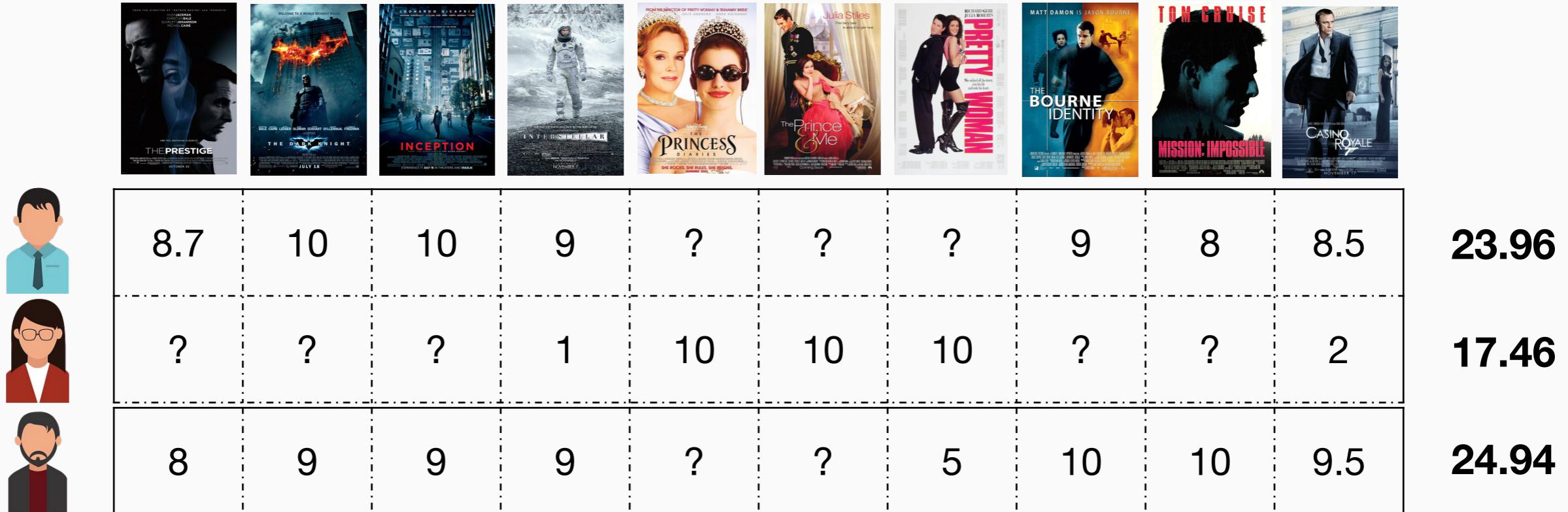
**Euclidean distance: distance between points**

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

**Cosine similarity: angle between vectors**

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

# Example: User-Based CF



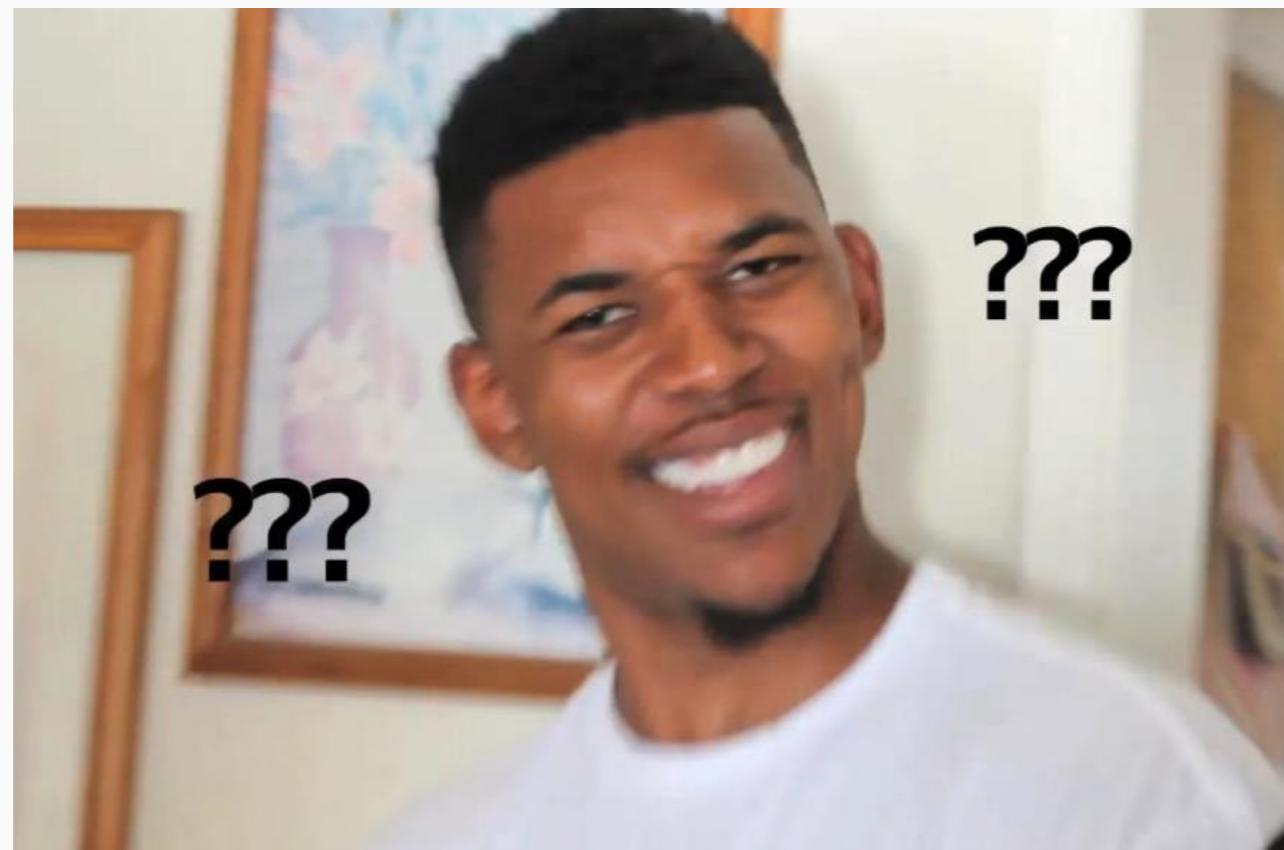
→ cosine similarity:

$$\begin{array}{ccc} \img[alt="User 1 icon"] & \img[alt="User 2 icon"] & : \frac{9 * 1 + 8.5 * 2}{23.96 * 17.46} = 0.062 \end{array}$$

$$\begin{array}{ccc} \img[alt="User 1 icon"] & \img[alt="User 3 icon"] & : \frac{8.7 * 8 + 10 * 9 + 10 * 9 + 9 * 9 + 9 * 10 + 8 * 10 + 8.5 * 9.5}{23.96 * 24.94} = 0.973 \end{array}$$

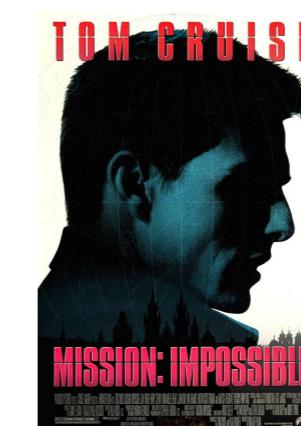
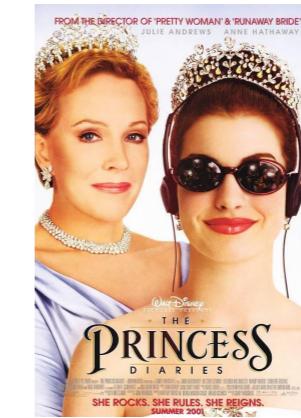
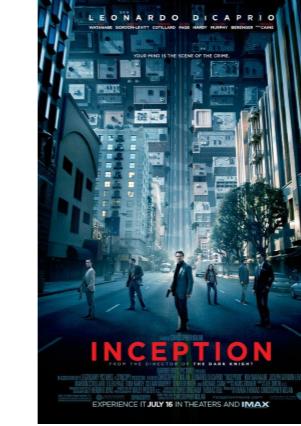
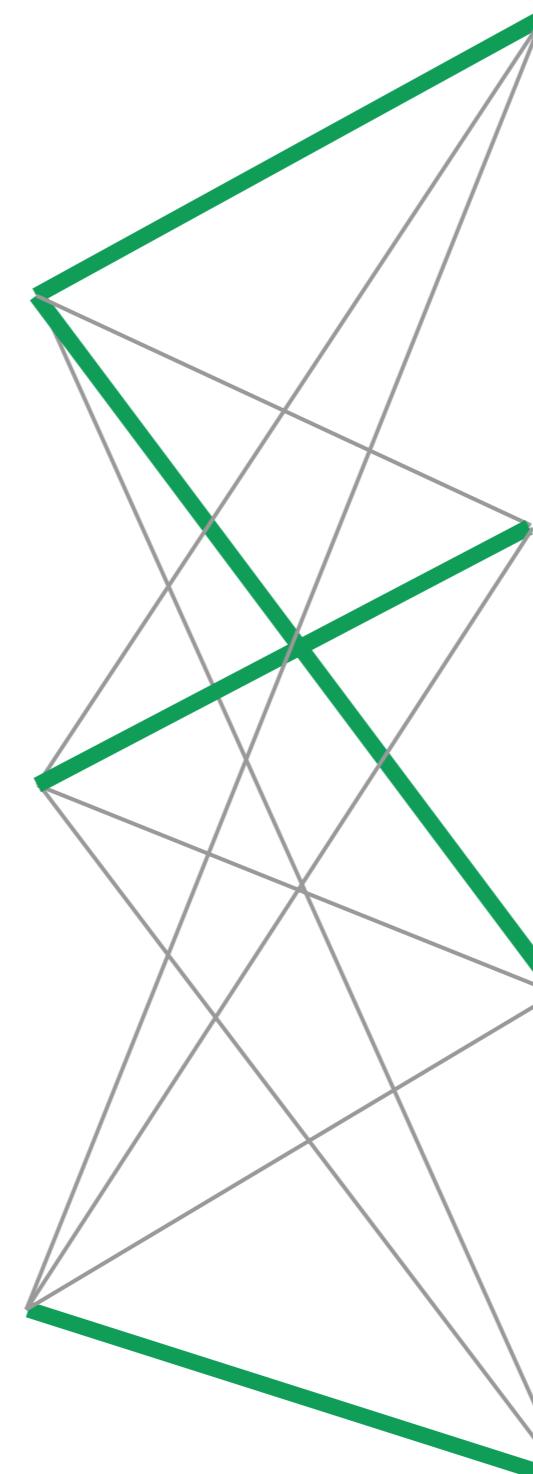
# Latent Factor Models

- Explain the rating by characterizing both users and items on a number of *latent factors* inferred from rating patterns.



# The factors are latent.

No one  
cares ...



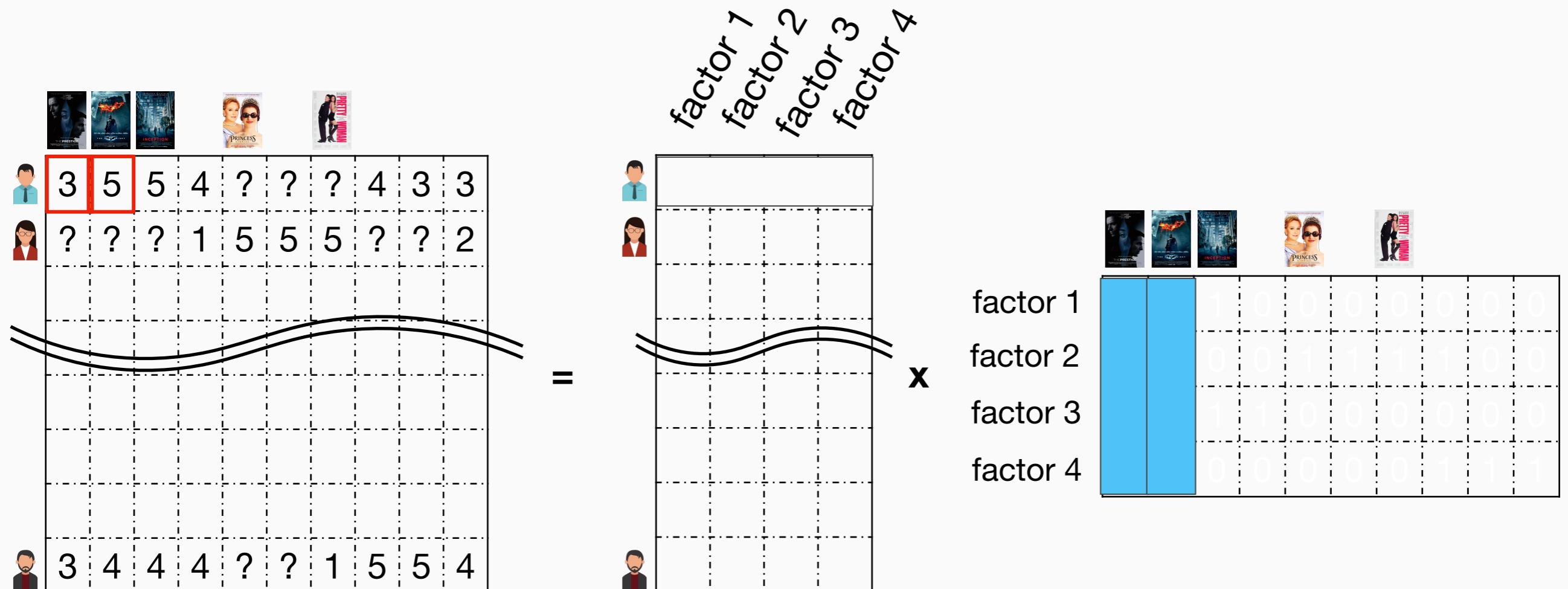
Not directly  
observable ...



3	5	5	4	?	?	?	4	3	3
?	?	?	1	5	5	5	?	?	2
●									
●									
●									
●									
3	4	4	4	?	?	1	5	5	4



# What is latent factor anyway?



# Singular Value Decomposition (SVD)

$$A = U \Sigma V^T$$

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix  $A$ . It shows a 4x4 matrix  $A$  on the left, followed by an equals sign. To the right of the equals sign is a 4x4 matrix  $U$ , which is orthogonal. Next is a 4x4 diagonal matrix  $\Sigma$ , representing the singular value matrix. Finally, there is a 4x4 matrix  $V^T$ , which is also orthogonal. The matrices  $U$  and  $V^T$  are drawn with blue outlines, while  $\Sigma$  is black.

# That is (kind of) what SVD is

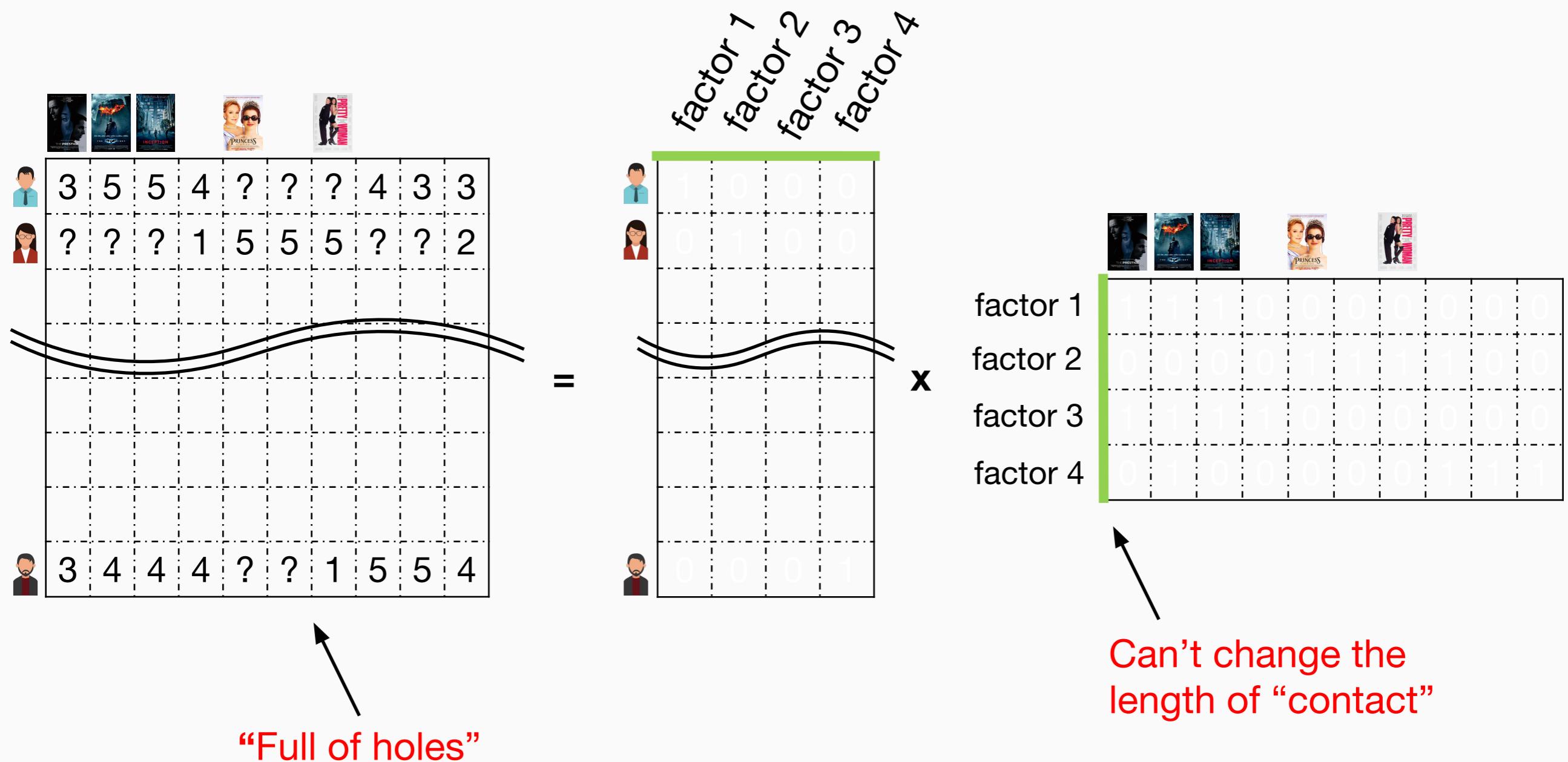
3	5	5	4	?	?	?	4	3	3
?	?	?	1	5	5	5	?	?	2
3	4	4	4	?	?	1	5	5	4

=


x

1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	0	1	0	1	0	0
0	1	0	0	0	0	1	0	1	0
0	0	1	0	0	1	0	0	0	1

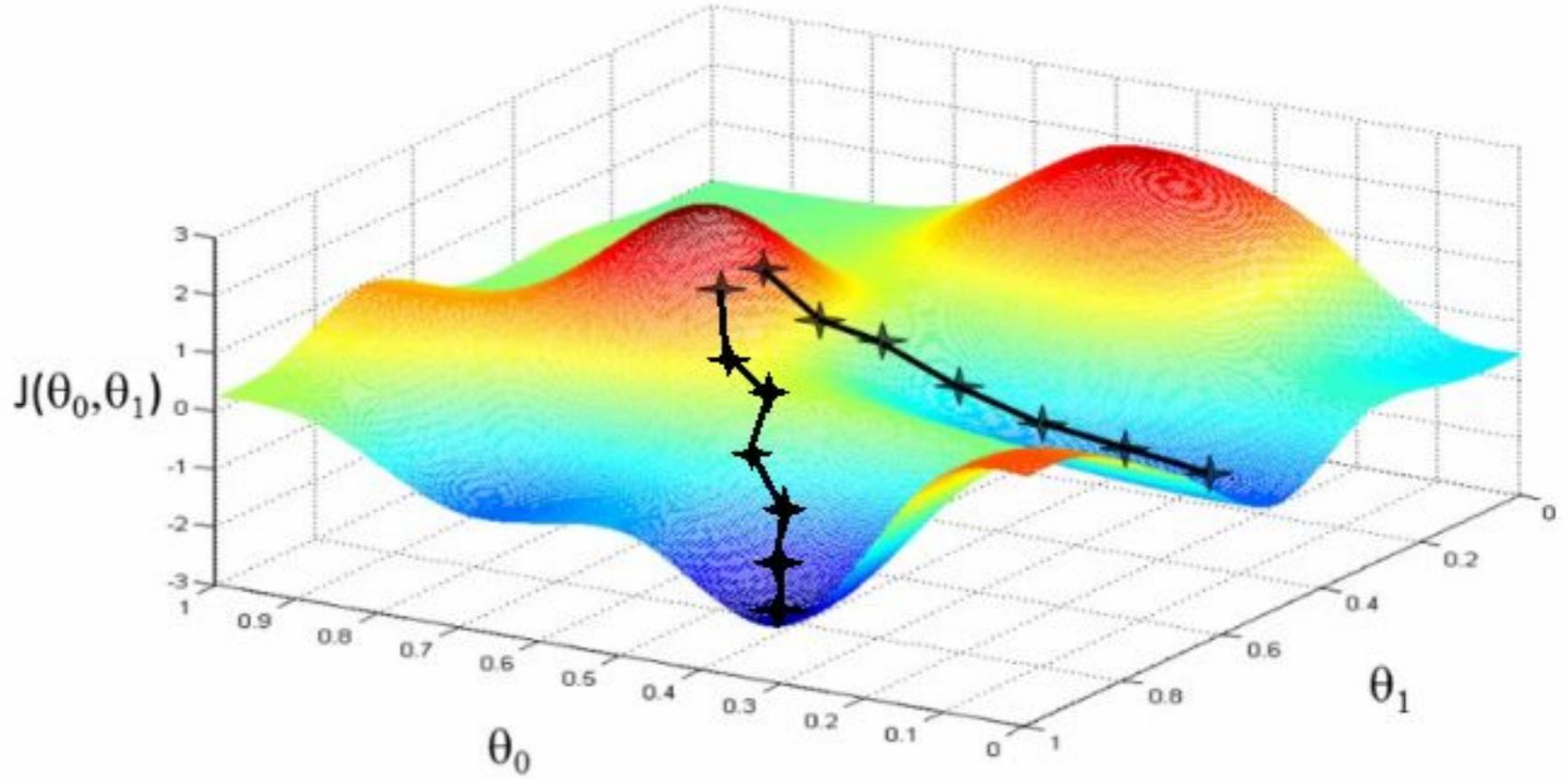
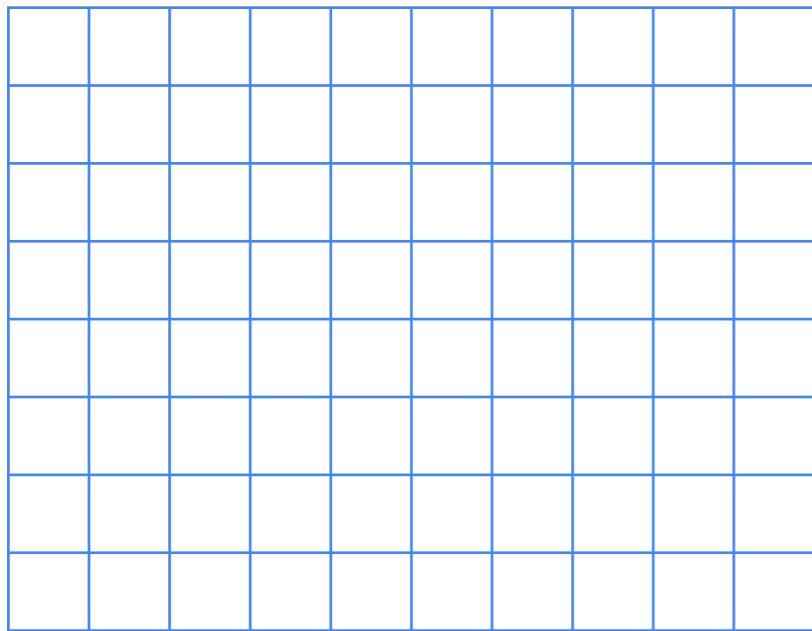
# But in practice...



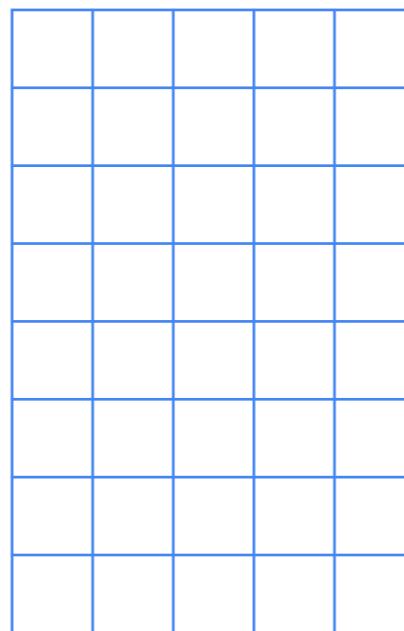
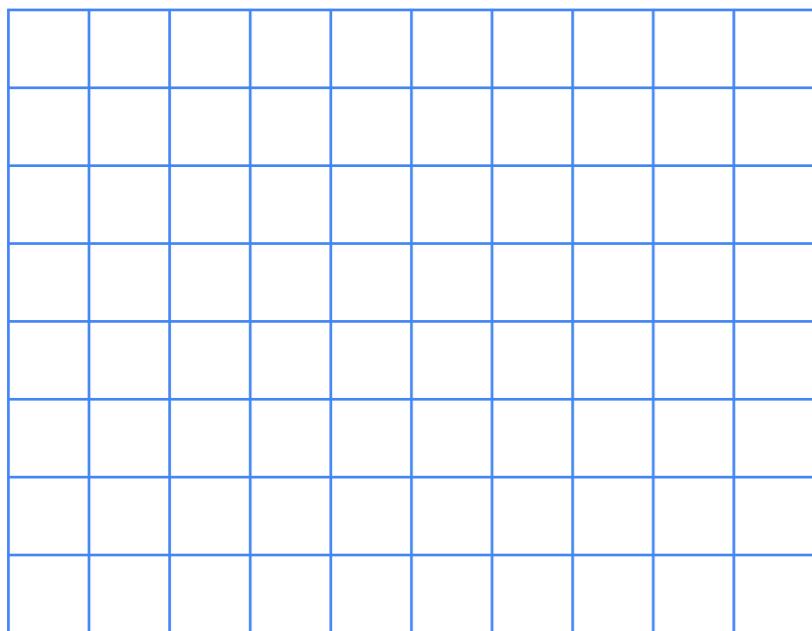
# How to factor the matrix

- To perform matrix factorization for large matrices, we learn the entries through optimization methods such as *stochastic gradient descent* (SGD).
- Methods like *alternating least square* (ALS) are also used when computation can be parallelized.
- We are going to briefly introduce SGD for its popularity.

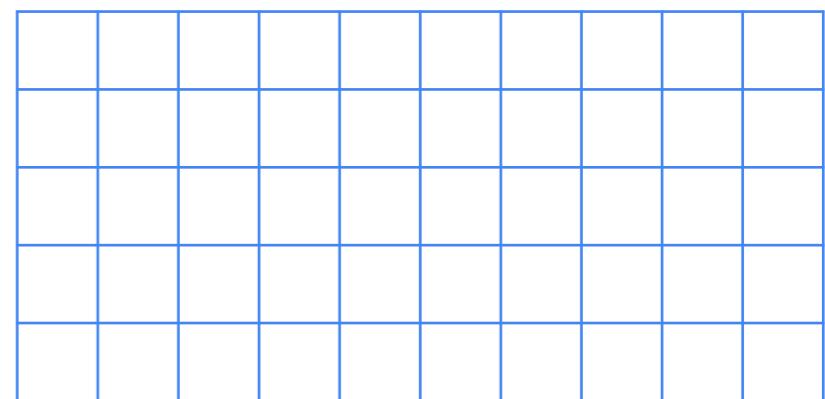
**Goal:**



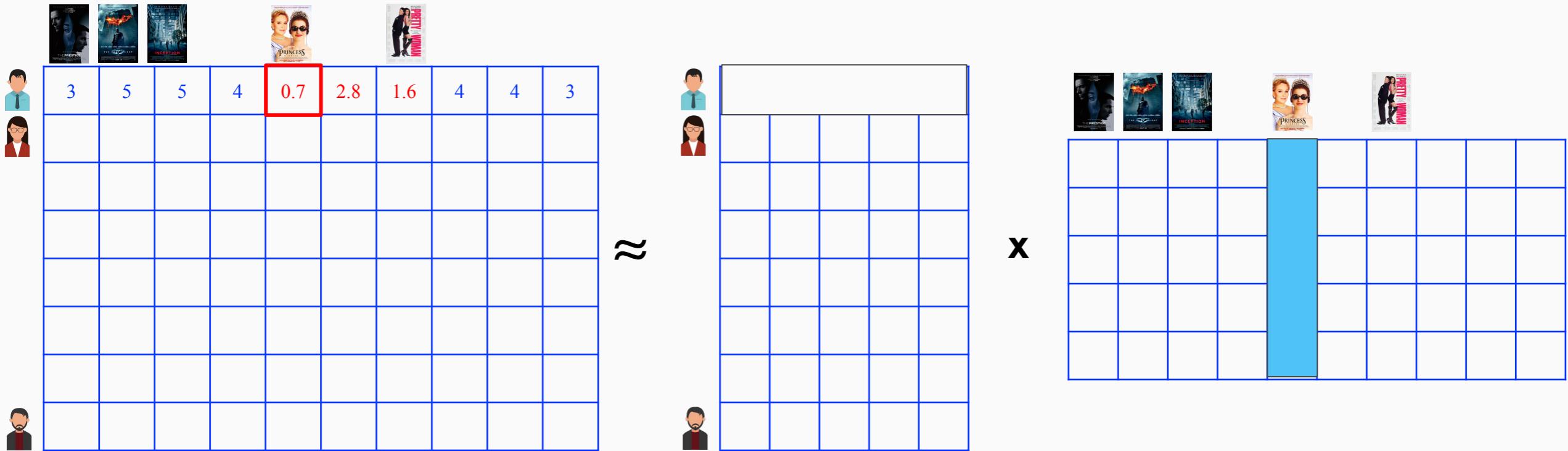
**Initialize these and tune the numbers!**



**x**



# Ok, I've got the matrices. Then what?



## → Key points:

1. The approximate matrix will **not** be identical to the original.
2. The factor matrices will keep changing as long as there are users changing the rating (even if you stay inactive for a while).

# Evaluating a model

- To evaluate how a machine learning model did, we use metrics such as *precision* and *recall*.

		Actual value	
		True	False
		True	True positive (TP)
Predicted value	True	False positive (FP)	
	False	False negative (FN)	True negative (TN)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

## Scenario 1:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	45	5
	Safe	190	1000

$$\text{Precision} = \frac{45}{45 + 5} = 90\%$$

$$\text{Recall} = \frac{45}{45 + 190} = 19.15\%$$

## Scenario 2:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	200	800
	Safe	35	205

$$\text{Precision} = \frac{200}{200 + 800} = 20\%$$

$$\text{Recall} = \frac{200}{200 + 35} = 85.11\%$$

# That's why we have F1-score

- F1-score (or F-score) is the harmonic mean between precision and recall:

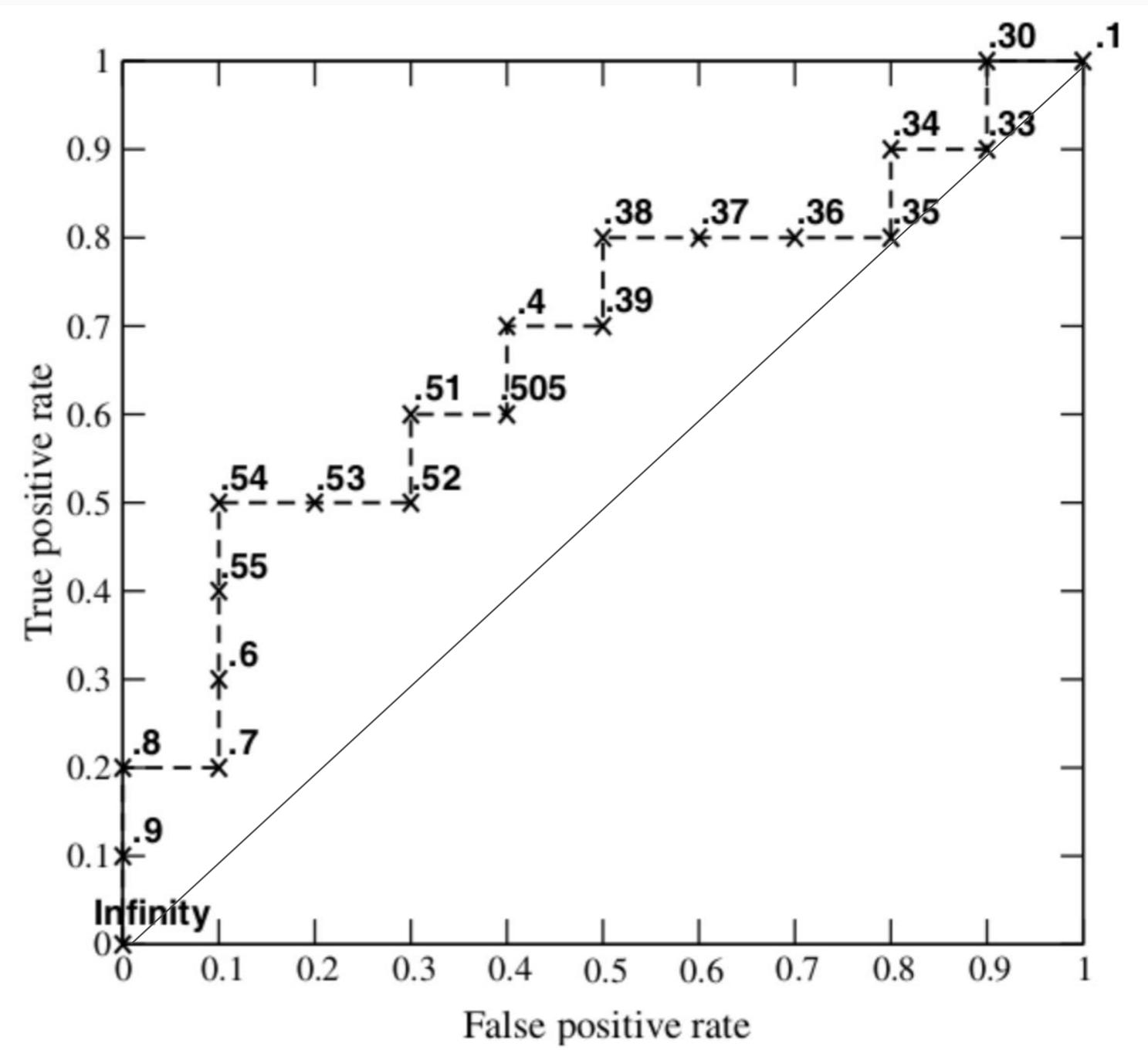
$$F1 = 2 * \frac{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}{2} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

# Receiver Operating Characteristic (ROC) curve

		Actual value	
		True	False
		True	False positive (FP)
Predicted value	True	True positive (TP)	
False	False negative (FN)	True negative (TN)	

True positive rate:  $TPR = TP / (TP + FN)$

False positive rate:  $FPR = FP / (FP + TN)$



## Let's evaluate the ranking as well

- A RE often delivers numerous outputs, while only a portion of them is most relevant to what the user really wants.
- We **rank our results** for the users, so the entries that the user would most likely select would be near the top.
- How do we evaluate the *ranking* of results? 

# 鍾欣怡

Search



(鍾欣桐)



1



3



(鍾欣凌)

4

## Relevance (0-3 scale):



1

**Cumulative Gain:**  $1 + 3 + 3 + 1 = 8$   
(@ rank 4)



3

**Discounted Cumulative Gain (DCG):**

(@ rank p)

$$\sum_{i=1}^p \frac{rel_i}{\log_2(i+1)}$$

$$\frac{1}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{3}{\log_2(3+1)} + \frac{1}{\log_2(4+1)} = 4.824$$



3

$$\frac{3}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{1}{\log_2(3+1)} + \frac{1}{\log_2(4+1)} = 5.824$$



IDCG : 最理想化的分數



1

**Normalized DCG:**  $\frac{DCG_p}{IDCG_p}$

(@ rank p)