**Big Data Analysis**

**Collaborating Filtering & Alternating Least Squares**

**(Dataset: Audioscrobbler data set and recommending music)**

Ji-Hyeong Han

## ( [jhhan@seoultech.ac.kr](mailto:jhhan@seoultech.ac.kr) )

Dept. of Computer Science and Engineering

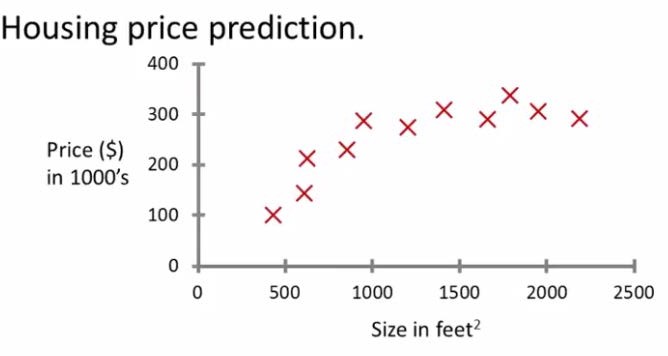


**Supervised Learning**

* **Regression**
* **Classification**

2

**Regression**



## Supervised learning

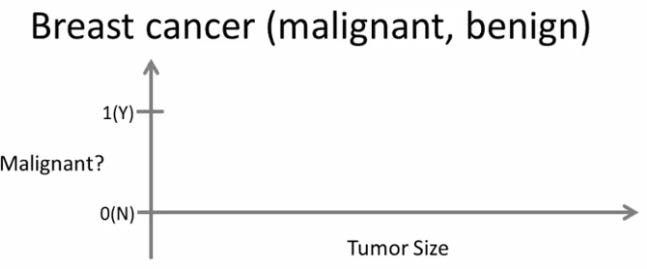
* + “Right Answer” given

## Regression

* + Predict continuous valued output (ex: price)

3

**Classification**



**X**

**X**

**X X X**

**X**

**X**

**X**

**X X**

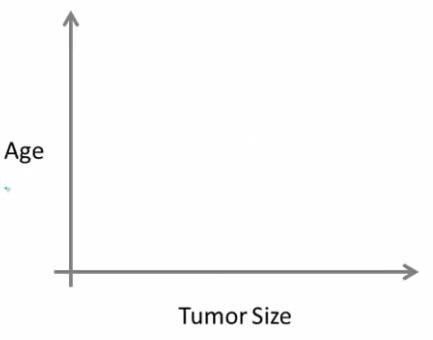
## Classification

* + Discrete valued output (ex: multiple labels)

4

**Classification**

## More than one feature



**O**

**O**

**O**

**O O**

**O**

**X X**

**X**

**X**

**X**

**O**

**X O**

**O**

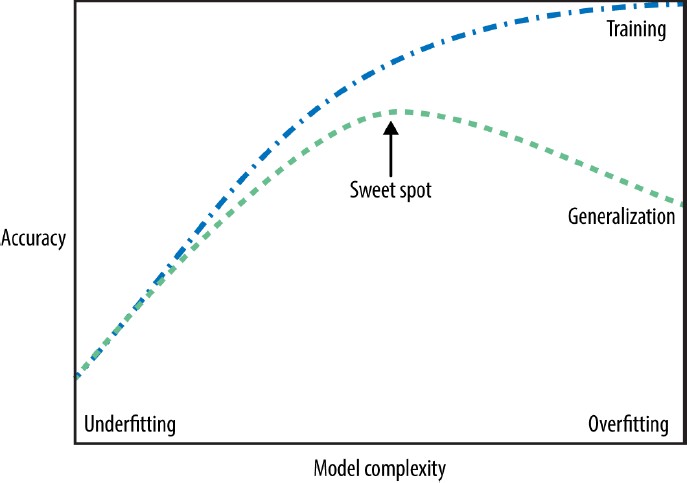
**O X X**

**X**

5

**Generalization, Overfitting, and Underfitting**

## Trade-off of model complexity against training and test accuracy

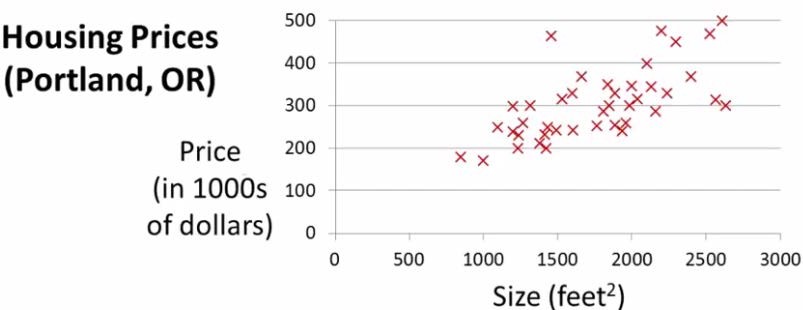


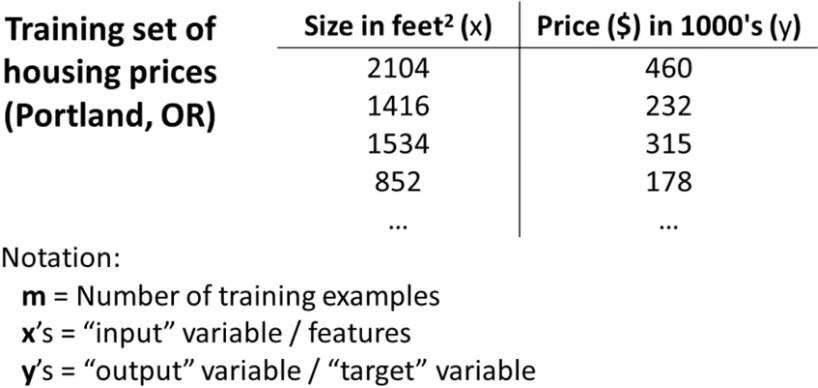
6

# Gradient Descent Algorithm

7

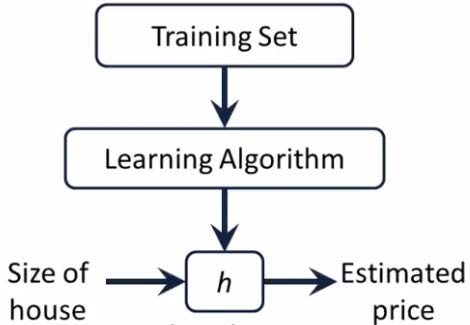
**Data Example**





8

## How do we represent *h*?

ℎ𝜃𝜃

y

x

x

x

x

x

= 𝜃𝜃0 + 𝜃𝜃1𝑥𝑥

x

𝑥𝑥

x Hypothesis

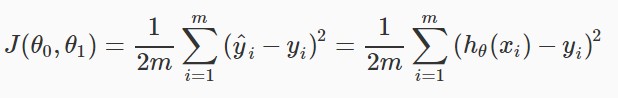
Estimated value of y

**Learning**

9

**Cost Function**

## Minimize the cost function



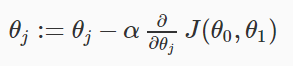
* + Choose 𝜃𝜃0, 𝜃𝜃1 so that ℎ𝜃𝜃(𝑥𝑥) is close to 𝑦𝑦 for our training examples (x,y)

10

**Gradient Descent Algorithm**

## https://d3c33hcgiwev3.cloudfront.net/imageAssetProxy.v1/bn9SyaDIEeav5QpTGIv-Pg_0d06dca3d225f3de8b5a4a7e92254153_Screenshot-2016-11-01-23.48.26.png?expiry=1506470400000&hmac=nHhdZ9bJAMTEbn_-rPNEhsgBLcHpiImy8qLVMdoEue0Graph of cost function according to 𝜃𝜃0, 𝜃𝜃1

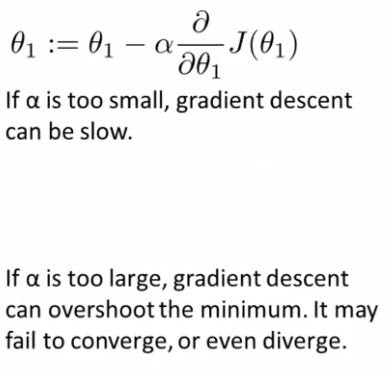
* Repeat until convergence

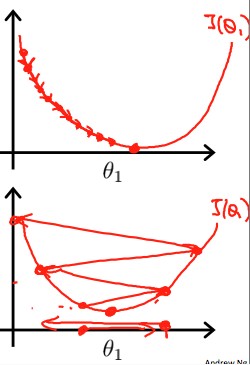


11

**Gradient Descent Algorithm**

## Learning rate





12

# Algorithm (Collaborating Filtering,

**Alternating Least Squares)**

13

**Characteristics of Dataset**

## A data set published by Audioscrobbler

* + Audioscrobbler was the first music recommendation system for [last.fm](https://www.last.fm/) (one of the first internet streaming radio sites, founded in 2002)
  + Audioscrobbler provided an open API for “scrobbling”1) or recording listeners’ song plays.
  + last.fm used this information to build a powerful music recommender engine

## At that time, research on recommender engines was mostly confined to learning from rating-like data.

* + ex: “Bob rates Prince 3.5 stars”

1. scrobble: (of an online music service) to record a listener’s musical preferences and recommend

similar music that he or she might enjoy 14

**Characteristics of Dataset**

## But, the Audioscrobbler data set is interesting

* + It merely records plays
    - ex: “Bob played a Prince track”

## A play carries less information than a rating

* + - Because Bob played the track doesn’t mean he actually liked it

## However, listeners rate music far less frequently than they play music

* + So, dataset like Audioscrobbler is much larger, covers more users and artists, and contains more total information than a rating data set, even if each individual data point carries less information

15

**Characteristics of Dataset**

## Implicit feedback

* + - This type of data is called “implicit feedback”
    - Because, the user-artist connections are implied as a side effect of other actions, and not given as explicit ratings or thumbs-up

16

**Choose Algorithm**

## We need to choose a recommender algorithm that is suitable for this implicit feedback data

* + The data set consists entirely of interactions between users and artist’s songs
    - No information about the users, or about the artists

## So, we need an algorithm that learns without access to user or artist attributes

17

**Choose Algorithm**

## Collaborative filtering

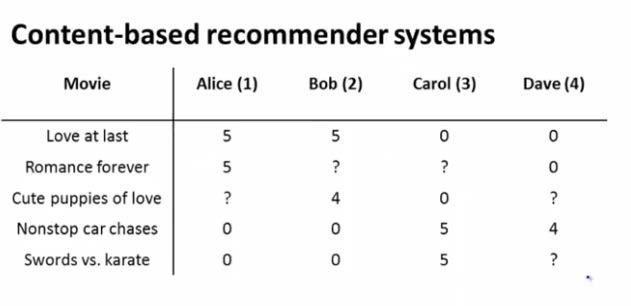
* + - Deciding that two users might share similar tastes because they are the same age is not an example of collaborative filtering
    - Deciding that two users might both like the same song because they play many other same songs is the correct example of collaborative filtering

## Content based recommendation

* + - Recommendation is an important application of machine learning
    - Amazon wants to recommend the new book
    - Netflix wants to recommend the new movie
    - By looking at what books you may have purchased in the past, or what movie you have rated in the past

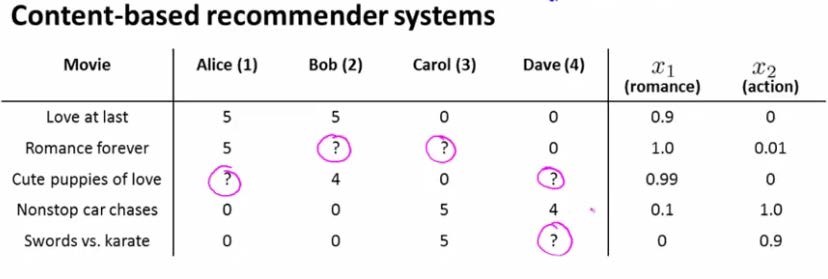
18

**Content Based Recommendation**



𝑛𝑛𝑢𝑢 = 4 𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑢𝑢𝑡𝑡𝑛𝑛𝑢𝑢

𝑛𝑛𝑚𝑚 = 5 (𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑜𝑜𝑚𝑚𝑚𝑚𝑡𝑡𝑢𝑢) 19



movie features

𝑥𝑥0 = 1

𝑥𝑥1

𝑥𝑥2

𝑥𝑥3

𝑥𝑥4

𝑥𝑥5

## 𝑥𝑥1 =

1

## 0.9

0

𝑥𝑥2 =

1

## 1.0

0.01

𝑥𝑥3 =

1

## 0.99

0

𝑥𝑥4 =

1

## 0.1

1.0

𝑥𝑥5 =

1

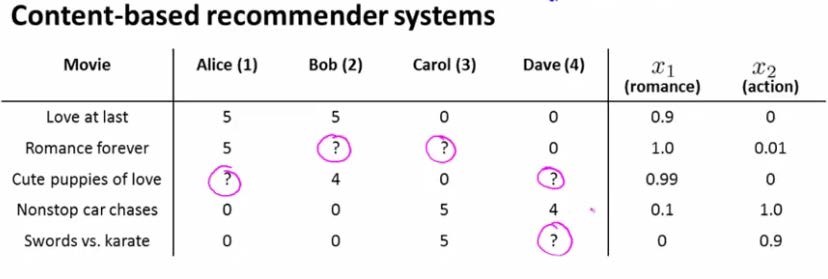
## 0

0.9

**Content Based Recommendation**

𝑛𝑛𝑢𝑢 = 4 𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑢𝑢𝑡𝑡𝑛𝑛𝑢𝑢

𝑛𝑛𝑚𝑚 = 5 (𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑜𝑜𝑚𝑚𝑚𝑚𝑡𝑡𝑢𝑢) 20



movie features

𝑥𝑥0 = 1

𝜃𝜃1

𝜃𝜃2

𝜃𝜃3

𝜃𝜃4

𝑥𝑥1

𝑥𝑥2

𝑥𝑥3

𝑥𝑥4

𝑥𝑥5

## 𝑥𝑥1 =

1

## 0.9

0

𝑥𝑥2 =

1

## 1.0

0.01

𝑥𝑥3 =

1

## 0.99

0

𝑥𝑥4 =

1

## 0.1

1.0

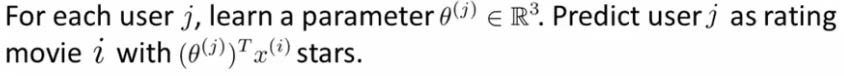
𝑥𝑥5 =

1

## 0

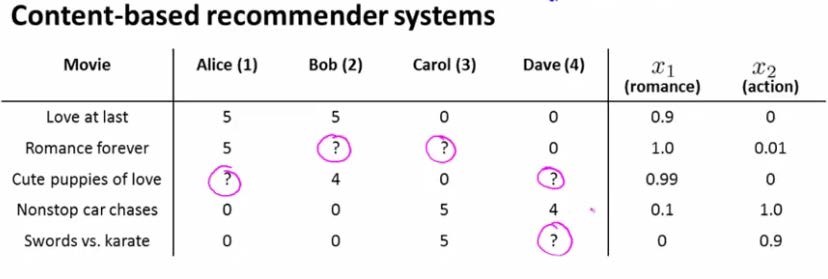
0.9

**Content Based Recommendation**



𝑛𝑛𝑢𝑢 = 4 𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑢𝑢𝑡𝑡𝑛𝑛𝑢𝑢

𝑛𝑛𝑚𝑚 = 5 (𝑡𝑡ℎ𝑡𝑡 𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑛𝑡𝑡𝑛𝑛 𝑜𝑜𝑜𝑜 𝑛𝑛𝑜𝑜𝑚𝑚𝑚𝑚𝑡𝑡𝑢𝑢) 21



movie features

𝑥𝑥0 = 1

𝜃𝜃1

𝜃𝜃2

𝜃𝜃3

𝜃𝜃4

𝑥𝑥1

𝑥𝑥2

𝑥𝑥3

𝑥𝑥4

𝑥𝑥5

## 𝑥𝑥1 =

1

## 0.9

0

𝑥𝑥2 =

1

## 1.0

0.01

𝑥𝑥3 =

1

## 0.99

0

𝑥𝑥4 =

1

## 0.1

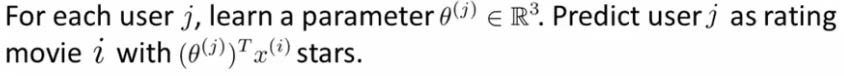
1.0

𝑥𝑥5 =

1

## 0

0.9



Suppose that some learning algorithm has learned 𝜃𝜃1 (Alice parameter vector)

0

𝜃𝜃1 = 5

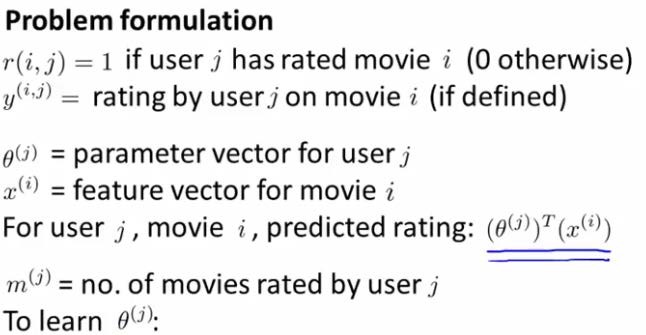
**Content Based Recommendation**

0

, then Alice’s rating for third movie would be 4.95

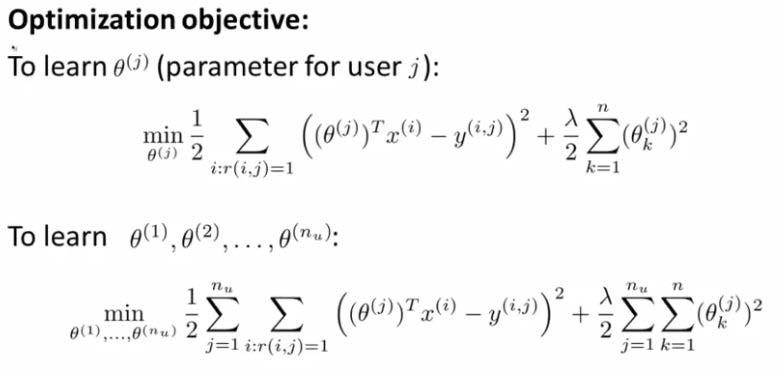
22

**Content Based Recommendation**



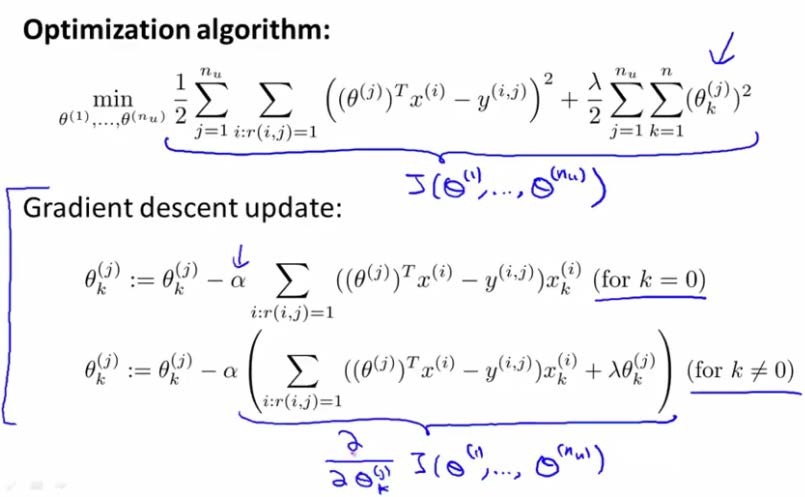
23

**Content Based Recommendation**



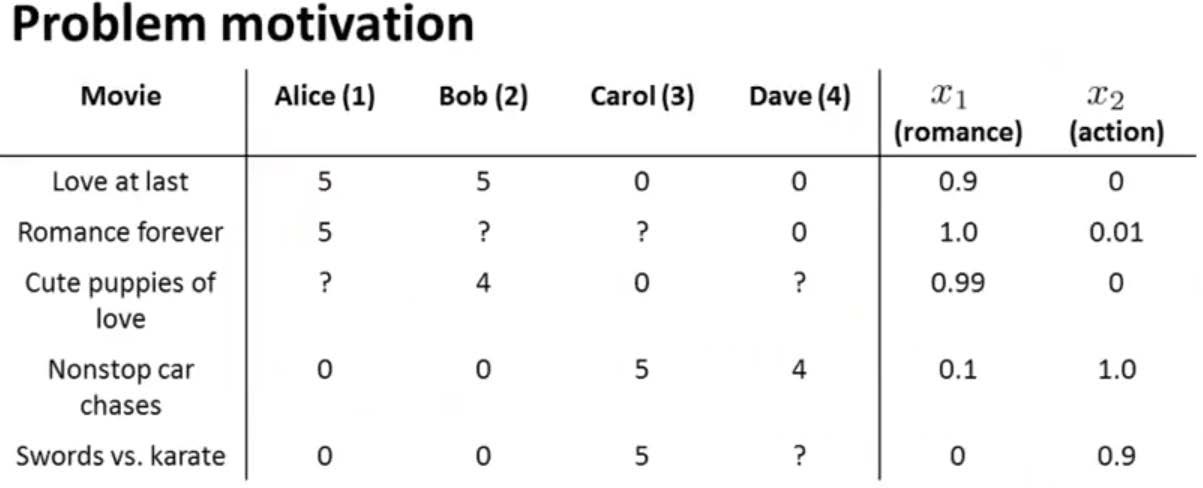
24

**Content Based Recommendation**



25

**Collaborative Filtering**

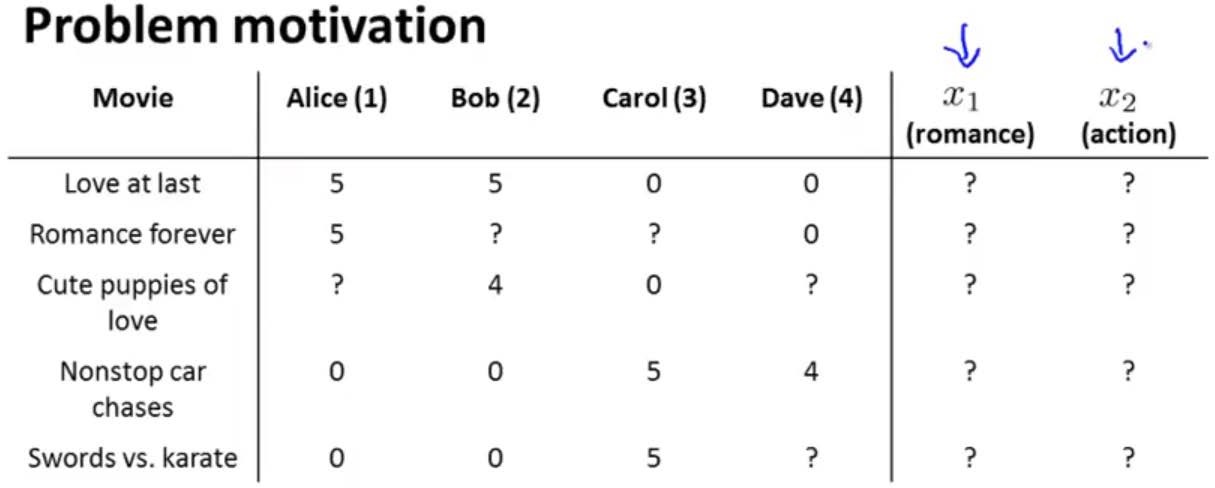


## It is difficult to say how much a movie is romance or action

* + So, most case we don’t have content based features

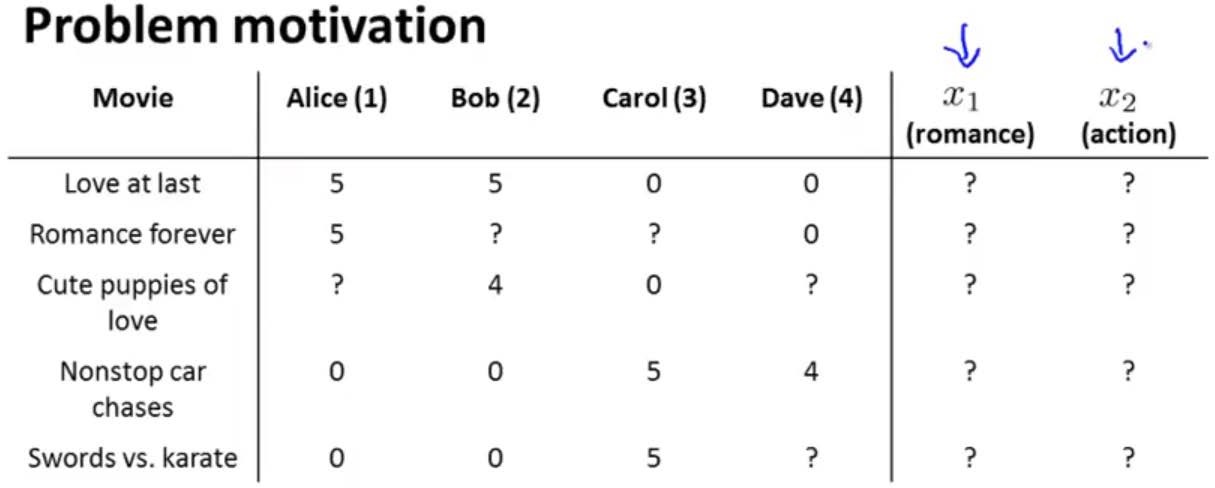
26

**Collaborative Filtering**



## Most case we don’t have content based features

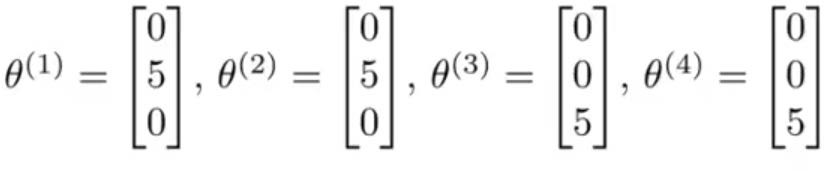
27



≈ 1.0

≈ 0.0

(𝜃𝜃1)𝑇𝑇𝑥𝑥1 ≈ 5



(𝜃𝜃2)𝑇𝑇𝑥𝑥1 ≈ 5

(𝜃𝜃3)𝑇𝑇𝑥𝑥1 ≈ 0

(𝜃𝜃4)𝑇𝑇𝑥𝑥1 ≈ 0

𝑥𝑥1 =

1

1.0

0.0

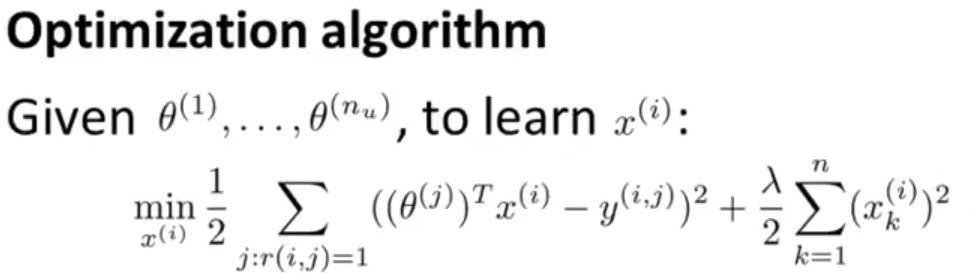
## Let’s assume that users tell us how much they like romance and actions movies

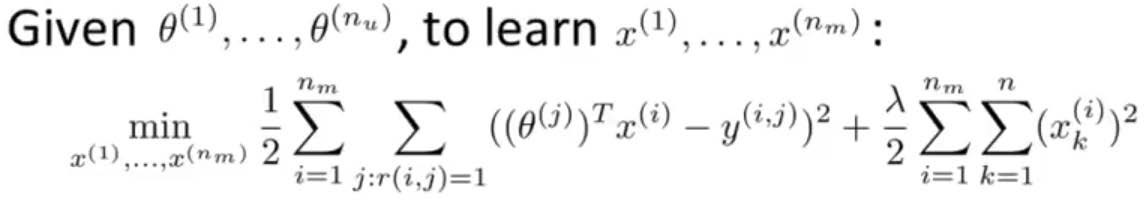
**Collaborative Filtering**

* + If we can get 𝜃𝜃 from users, then it turns out that it becomes possible

to try to infer what are the values of 𝑥𝑥1, 𝑥𝑥2 for each movie 28

**Collaborative Filtering**

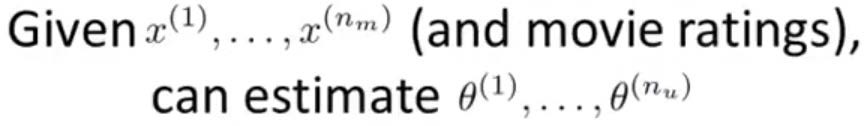




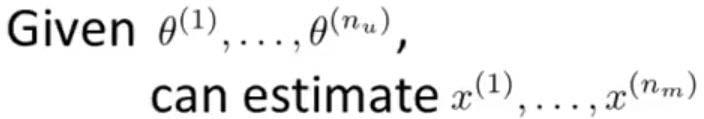
29

**Collaborative Filtering**

## Content based recommendation



* + User preference based feature estimation

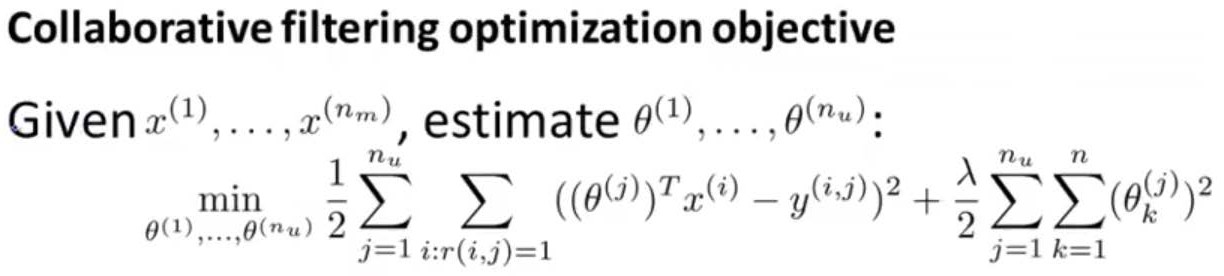


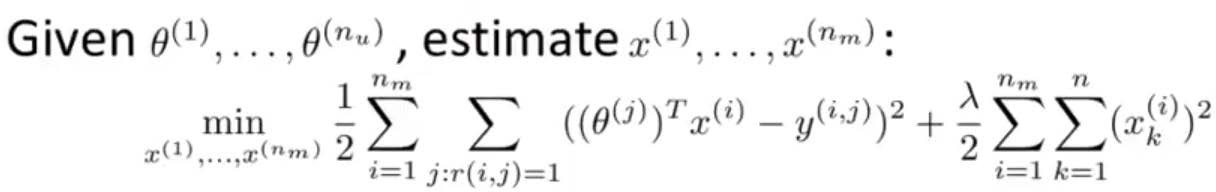
* + Collaborate filtering
    - Randomly choose 𝜃𝜃

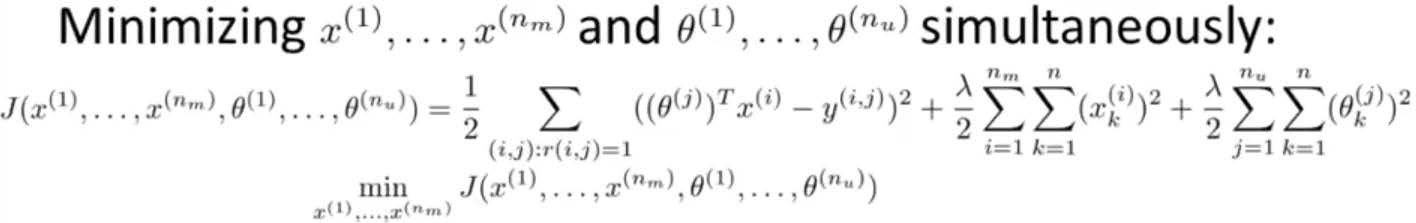
- 𝜃𝜃 → 𝑥𝑥 → 𝜃𝜃 → 𝑥𝑥 → 𝜃𝜃 → 𝑥𝑥 → ⋯

30

**Collaborative Filtering**

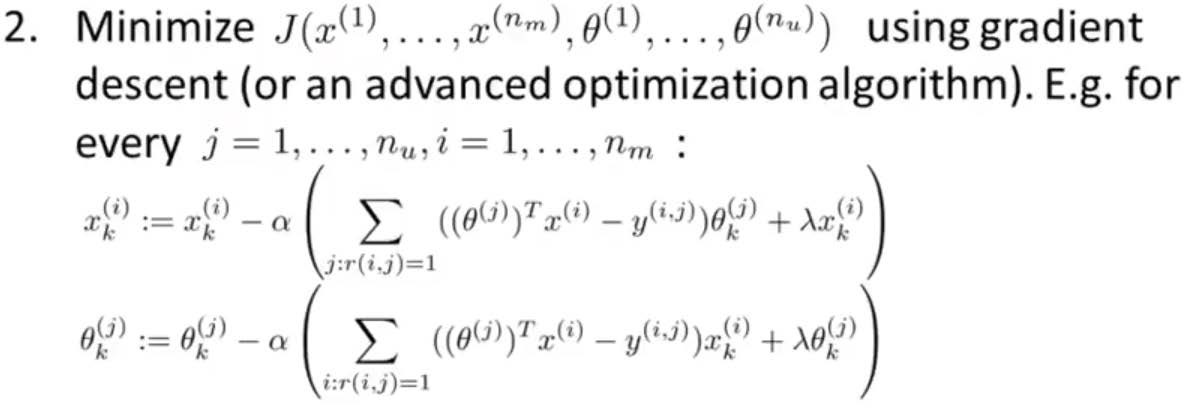
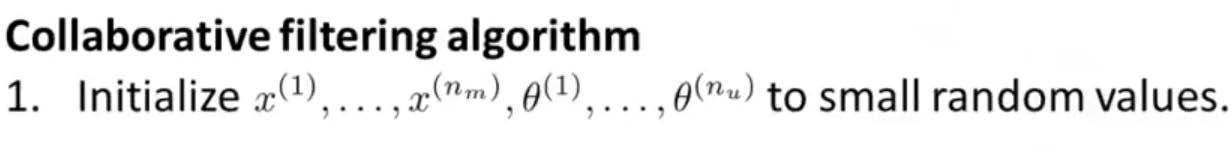


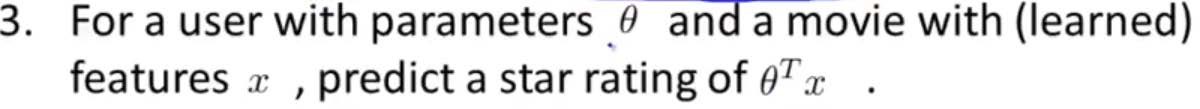




31

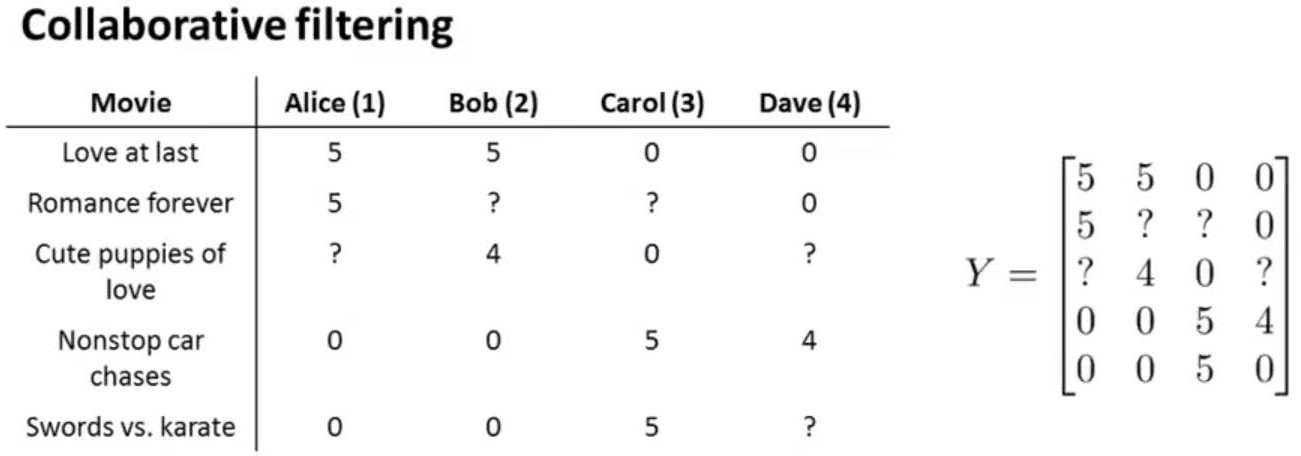
**Collaborative Filtering**





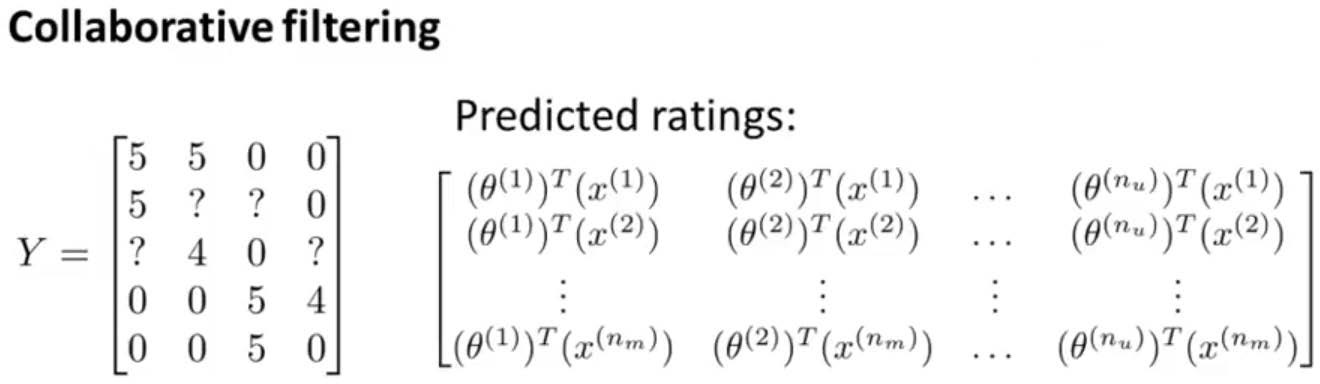
32

**Low Rank Matrix Factorization**



## Alternative way to write collaborative filtering

33



## X =

𝑥𝑥1 𝑇𝑇

𝑥𝑥2 𝑇𝑇

## ⋮

𝑥𝑥𝑛𝑛𝑚𝑚 𝑇𝑇

## , Ɵ =

𝜃𝜃1 𝑇𝑇

𝜃𝜃2 𝑇𝑇

## ⋮

𝜃𝜃𝑛𝑛𝑢𝑢 𝑇𝑇

**Low Rank Matrix Factorization**

## Predicted ratings: XƟ𝑇𝑇 → low rank matrix factorization

* + So, low rank matrix factorization is essentially exactly the same algorithm which is collaborative filtering

34

**Come back to the Audioscrobbler Data Set**

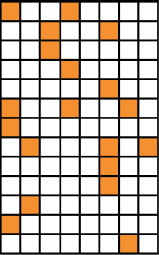
## This data set looks large

* + - Because, it contains tens of millions of play counts

## But, in a different sense, it is small and skimpy

* + - Because, it is sparse
    - On average, each user has played songs from about 171 artists – out of

1.6 million artists

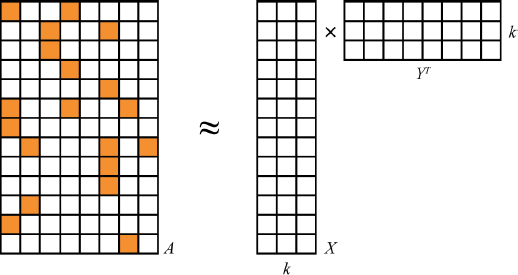
* + - Sparse means that most of the elements are zero

35

**Matrix Factorization**

## So, to deal with sparse matrix

* + Using matrix factorization
  + at row 𝑚𝑚 and column 𝑗𝑗 : user 𝑚𝑚 has played artist 𝑗𝑗
  + The factorization can only be approximate because 𝑘𝑘 is small
  + Sometimes called “matrix completion algorithm”
    - Because, original matrix A is sparse, but the product 𝑋𝑋𝑌𝑌𝑇𝑇 is dense

36

**Alternating Least Squares (ALS) Algorithm**

## We can not calculate the exact solution of 𝐴𝐴 = 𝑋𝑋𝑌𝑌𝑇𝑇

* + Because, X and Y aren’t large enough
  + Too low rank (small 𝑘𝑘)

## Still, 𝑋𝑋𝑌𝑌𝑇𝑇 should be as close to 𝐴𝐴 as possible

* We can use “Alternating Least Squares (ALS)” algorithm to compute

𝑋𝑋 and 𝑌𝑌

* + 𝑌𝑌 isn’t know, but it can be initialized as random value
  + Then, simple linear algebra gives the best solution for 𝑋𝑋 given 𝐴𝐴 and 𝑌𝑌

[1)](#_bookmark1)

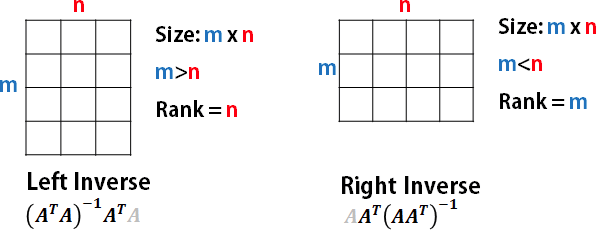
* + But, it can not be the same
  + So, we need to minimize the sum of squared error



* + And, we repeat the process to compute each 𝑌𝑌𝑗𝑗 from 𝑋𝑋 , and again compute 𝑋𝑋 from 𝑌𝑌, and until converge to decent solutions

37

**Ref: Pseudo Inverse**



[[Back to main page]](#_bookmark0) 38

# Example and Practice using Spark

39

**Example**

40



## Spark ML ALS example

[https://spark.apache.org/docs/2.0.0-preview/ml-collaborative- filtering.html](https://spark.apache.org/docs/2.0.0-preview/ml-collaborative-filtering.html)

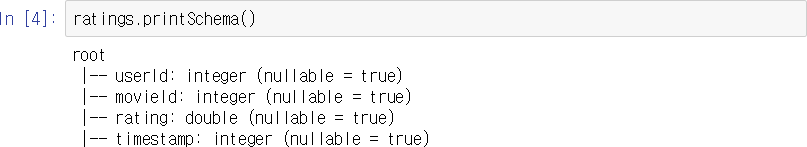
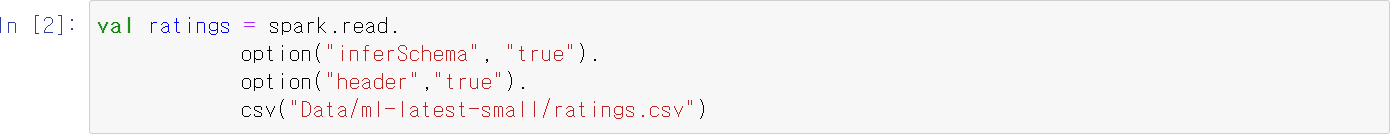
* Spark ML ALS API [https://spark.apache.org/docs/latest/api/scala/index.html#org.apache. spark.ml.recommendation.ALS](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.recommendation.ALS)
* Download data set
  + Movie lens

: recommended for education and development

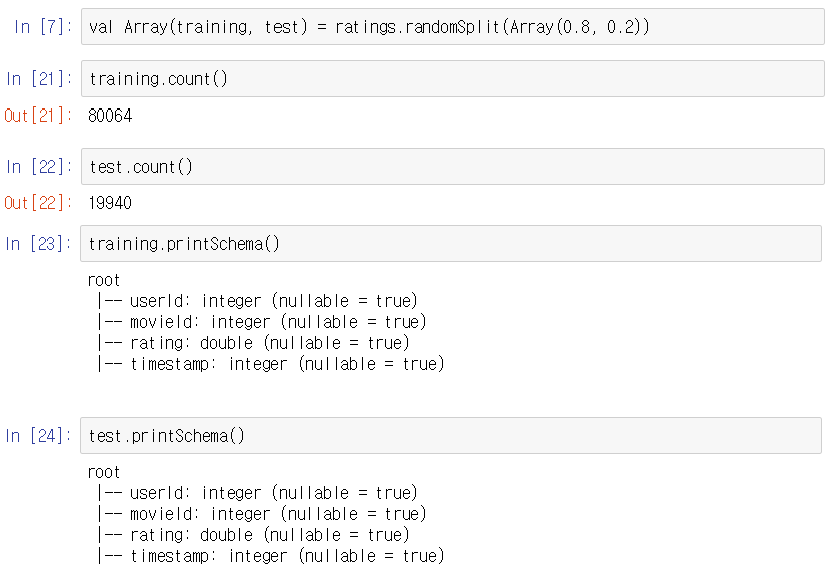
: small one!! <https://grouplens.org/datasets/movielens/>

**Example**





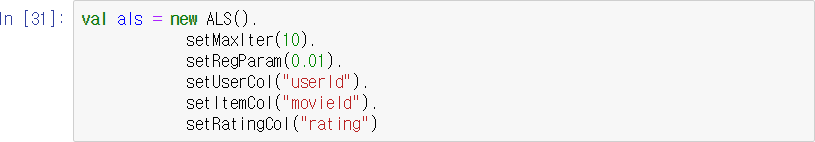
41



**Example**

42

**Example**

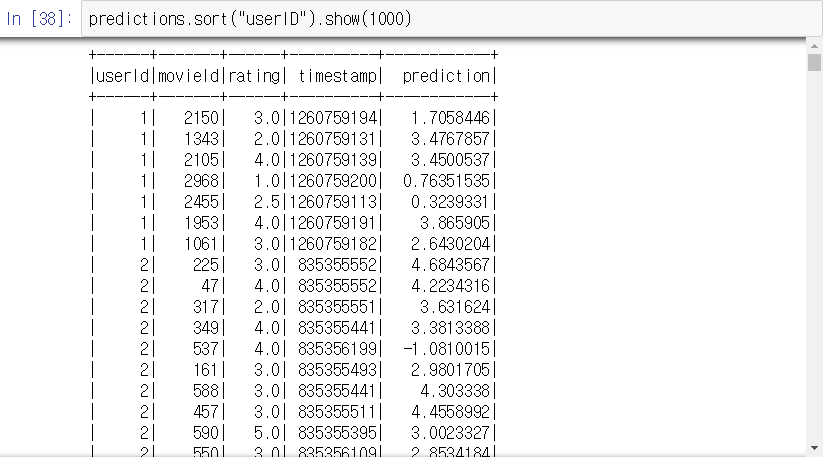






43

**Example**



44

**Practice (Audioscrobbler Data Set)**

## Download the data from e-class

* + The main data set is “user\_artist\_data.txt” file
  + It contains about 141,000 unique users, and 1.6 million unique artists
  + About 24.2 million users’ plays of artists are recorded, along with their counts
  + The names of each artist by ID are given in the “artist\_data.txt” file
  + There could be misspelled or nonstandard artist name
    - ex: “The Smiths”, “Smiths, The”, and “the smiths” may appear as distinct artist IDs in the data set even though they are plainly the same
  + So, the data set also includes “artist\_alias.txt” file, which maps artist IDs that are known misspellings or variants to the canonical ID of that artist

45

**Practice (Audioscrobbler Data Set)**

## Read “user\_artist\_data.txt” file

* You can add header to the DataFrame by using “toDF”



* This data set is too big, so we are going use a small part of data set



* Try to split the data set into training and test data set
* Train the ALS model by using training data
* Test the learned ALS model by using test data and show the result

46