

E:\Internship\22_MovieRecommendationSystemUsingSVD\movies.dat - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

movies.dat

```
1 1::Toy Story (1995)::Animation|Children's|Comedy
2 2::Jumanji (1995)::Adventure|Children's|Fantasy
3 3::Grumpier Old Men (1995)::Comedy|Romance
4 4::Waiting to Exhale (1995)::Comedy|Drama
5 5::Father of the Bride Part II (1995)::Comedy
6 6::Heat (1995)::Action|Crime|Thriller
7 7::Sabrina (1995)::Comedy|Romance
8 8::Tom and Huck (1995)::Adventure|Children's
9 9::Sudden Death (1995)::Action
10 10::GoldenEye (1995)::Action|Adventure|Thriller
11 11::American President, The (1995)::Comedy|Drama|Romance
12 12::Dracula: Dead and Loving It (1995)::Comedy|Horror
13 13::Balto (1995)::Animation|Children's
14 14::Nixon (1995)::Drama
15 15::Cutthroat Island (1995)::Action|Adventure|Romance
16 16::Casino (1995)::Drama|Thriller
17 17::Sense and Sensibility (1995)::Drama|Romance
18 18::Four Rooms (1995)::Thriller
19 19::Ace Ventura: When Nature Calls (1995)::Comedy
20 20::Money Train (1995)::Action
21 21::Get Shorty (1995)::Action|Comedy|Drama
22 22::Conquest (1995)::Crime|Drama|Thriller
```

Normal text file

length: 1,71,308 lines: 3,884

Ln: 10 Col: 20 Pos: 396

Type here to search

Windows taskbar icons including File Explorer, Google Chrome, and other applications.

31°C Haze



Recommendation System

Content-based Filtering

It is based on a description of the item

Collaborative Filtering

The people who have liked an item in the past will also like the same in future.

This approach builds a model based on the past behaviour of users.

The user behaviour may include previously watched videos, purchased items, given ratings on items

Type here to search

31°C Haze

$$A = U D V^T$$

Left singular vectors Singular values Right singular vectors

Definition

It decomposes a matrix of any shape into a product of 3 matrices

Lucid Analogy

$57 = 1 \times 3 \times 19$ - Prime Number in Middle

Elements

Matrix U - $m \times r$

Singular matrix of (user*latent factors)

Matrix S - $r \times r$

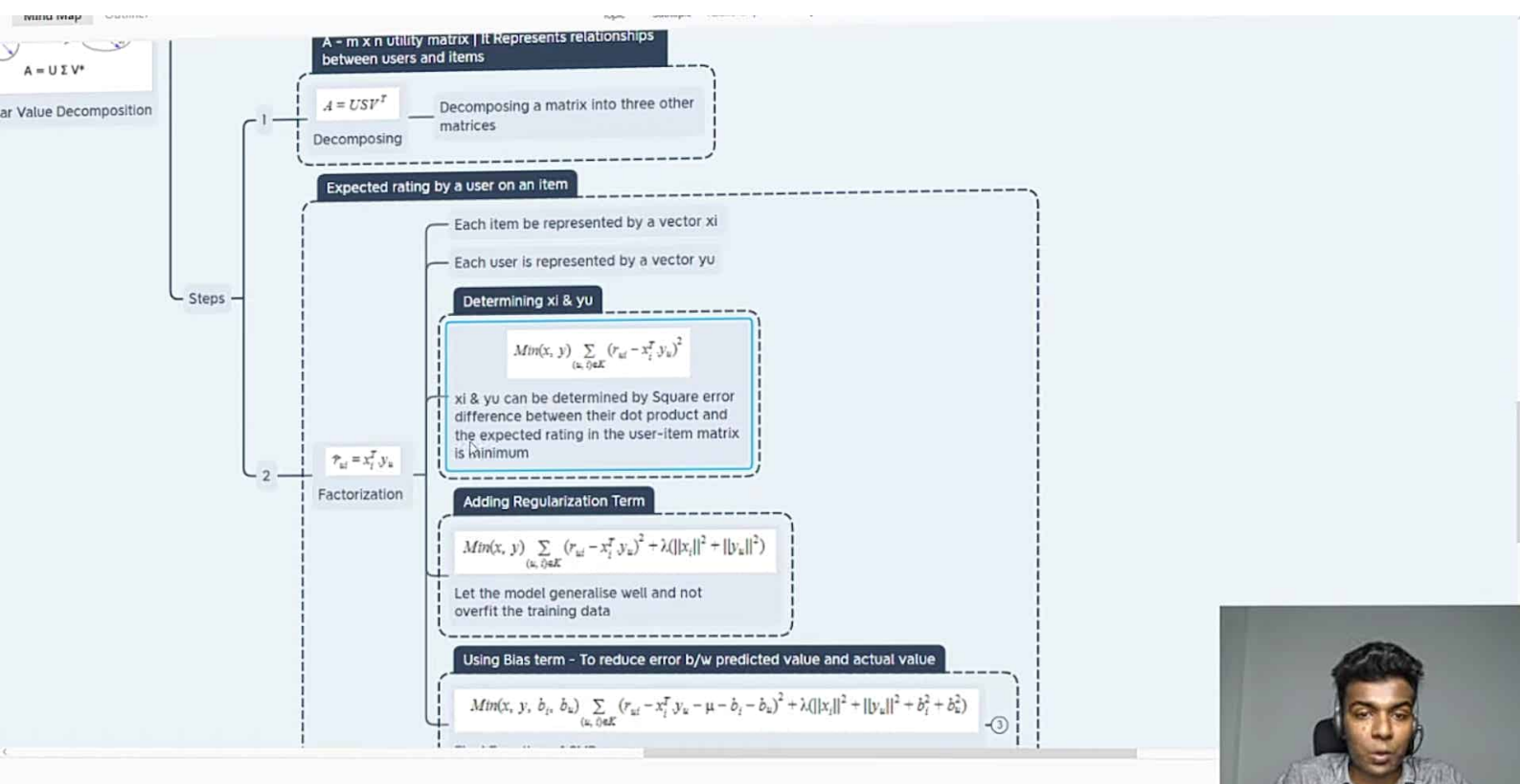
Diagonal matrix (shows the strength of each latent factor)

Matrix V - $r \times n$

Singular matrix of (item*latent factors)

Latent factors - characteristics of the item





Importing the basic libraries

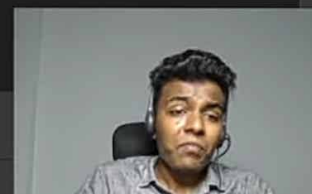
```
import numpy as np
import pandas as pd
```

Importing & Parsing the dataset as ratings and movies details

```
[2] ratingData = pd.io.parsers.read_csv('ratings.dat',
    names=['user_id', 'movie_id', 'rating', 'time'],
    engine='python', delimiter='::')
movieData = pd.io.parsers.read_csv('movies.dat',
    names=['movie_id', 'title', 'genre'],
    engine='python', delimiter='::')
print(ratingData)
```

	user_id	movie_id	rating	time
0	1	1193	5	978300760

0s completed at 19:29



2	1::661::3::978302109
3	1::914::3::978301968
4	1::3408::4::978300275
5	1::2355::5::978824291
6	1::1197::3::978302268
7	1::1287::5::978302039
8	1::2804::5::978300719
9	1::594::4::978302268
10	1::919::4::978301368
11	1::595::5::978824268
12	1::938::4::978301752
13	1::2398::4::978302281
14	1::2918::4::978302124
15	1::1035::5::978301753
16	1::2791::4::978302188
17	1::2687::3::978824268
18	1::2018::4::978301777
19	1::3105::5::978301713
20	1::2797::4::978302039
21	1::2321::3::978302205
22	1::720::3::978300760



[1000209 rows x 4 columns]

<> Create the ratings matrix of shape (m×u)

↑ ↓ ↻ ⌨ ⚙ 🗑 ⋮

✓ 0s ratingMatrix = np.ndarray(
shape=(np.max(ratingData.movie_id.values), np.max(ratingData.user_id.values)),
dtype=np.uint8)
ratingMatrix[ratingData.movie_id.values-1, ratingData.user_id.values-1] = ratingData.rating.values
print(ratingMatrix)

[[5 0 0 ... 0 0 3]
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
...
[42 0 0 ... 0 0 0]
[47 0 0 ... 0 0 0]
[51 0 0 ... 0 0 0]]

✓ 5s completed at 19:32




```
[3]: [42  0  0 ...  0  0  0]
      [47  0  0 ...  0  0  0]
      [51  0  0 ...  0  0  0]
```

Subtract Mean off - Normalization

```
normalizedMatrix = ratingMatrix - np.asarray([(np.mean(ratingMatrix, 1))]).T
print(normalizedMatrix)
```

```
[[ 3.57400662 -1.42599338 -1.42599338 ... -1.42599338 -1.42599338
  1.57400662]
 [-0.37152318 -0.37152318 -0.37152318 ... -0.37152318 -0.37152318
 -0.37152318]
 [-0.23874172 -0.23874172 -0.23874172 ... -0.23874172 -0.23874172
 -0.23874172]
 ...
 [36.53211921 -5.46788079 -5.46788079 ... -5.46788079 -5.46788079
 -5.46788079]
 [41.02980132 -5.97019868 -5.97019868 ... -5.97019868 -5.97019868
 -5.97019868]
```

0s completed at 19:35



colab.research.google.com/drive/14f6XGC50pM0Uuu_dyy5F-OAGTdmHnB64#scrollTo=wOuthXLIT0GI

22_MovieRecommendationSystemUsingSVD.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Computing SVD

```
A = normalizedMatrix.T / np.sqrt(ratingMatrix.shape[0] - 1)
U, S, V = np.linalg.svd(A)
```

Calculate cosine similarity, sort by most similar and return the top N


```
[6] def similar(ratingData, movie_id, top_n):
    index = movie_id - 1 # Movie id starts from 1
    movie_row = ratingData[index, :]
    magnitude = np.sqrt(np.einsum('ij, ij -> i', ratingData, ratingData)) #Einstein summation | traditional matrix
    similarity = np.dot(movie_row, ratingData.T) / (magnitude[index] * magnitude)
    sort_indexes = np.argsort(-similarity) #Perform an indirect sort along the given axis (Last
    return sort_indexes[:top_n]
```

Select k principal components to represent the movie, a movie_id to find recommend

Executing (54s) Cell > svd()

Type here to search

30°C Mostly cloudy



colab.research.google.com/drive/14f6XGC50pM0Uuu_dyy5F-OAGTdmHnB64#scrollTo=rTMd_brsUctX

22_MovieRecommendationSystemUsingSVD.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

Select k principal components to represent the movies, a movie_id to find recommendations and print the top_n results

```
k = 50
movie_id = 2
top_n = 5

sliced = V.T[:, :k] # representative data
indexes = similar(sliced, movie_id, top_n)

print('Recommendations for Movie {0}: \n'.format(
movieData[movieData.movie_id == movie_id].title.values[0]))
for id in indexes + 1:
    print(movieData[movieData.movie_id == id].title.values[0])
```

Recommendations for Movie Jumanji (1995):

1m 28s completed at 19:38

