# MTH-573 Numerical Linear Algebra Project Report

## -Ishan Bhargava (02017165)

## Title

### Using Euclidean Distance to Recommend Movies

## Objective

The goal of this project is to use the concept of norms to find a user with similar movie preferences and recommend movies based on the same

## Motivation

In this era of free information internet provides us with thousands of choices for anything we search. In this scenario a recommendation system becomes very important to save user's time. Not the user's, it helps business's to better sell their products or services. The biggest and most famous example of a recommendation system being Netflix, it recommends movies not only based on a genre the user usually watches but also based on other users whose genre preferences are similar.

# Background

Where services like Netflix and Youtube uses complex machine learning algorithms to recommend movies or videos, I am using concept of *Norms*, specifically 2-Norm or the Euclidean Norm of a matrix.

From Linear Algebra point of view I'm using the concept of Euclidean Norm. The Euclidean Norm is used to measure the shortest distance from origin to the vector head. This concept can be extended to find the distance between two vectors/matrices/tensors by finding the 2-norm of the difference of the two vectors.

I'm using Python to code my project. In Python I'm using two libraries Pandas and Numpy, which provide great flexibilty when working with matrices and huge datasets

# My Work

In the project I have used movies.csv (which contain movieId, movie title and movie genres) and ratings.csv (which contain userId, movieId and the rating from 1 to 5 that the user has given to the respective movie). The movie.csv is loaded as a matrix of size 62423 x 3 and the ratings.csv is loaded as a matrix of size 25000095 x 3

I am creating a test user matrix by selecting the first 20 movies from the movies matrix and rating these 20 movies. As a result the trial_user matrix will have a size of 20 x 2

Next, I'm searching for users in the ratings matrix who have rated the same 20 movies as our trial_user. Once I have the list of users, I am creating one matrix for each user (of size 20 x 2) which contains the movieId and the ratings and calculating the 2-Norm of the difference matrix of the trial_user matrix and the user matrix. The norm values are stored in a 2-Dimensional array where the first element of each row is the user_id with whose matrix the norm was calculated and the second element of the row is the 2-norm.

The user which has the minimum Euclidean norm with the trial_user has the closest movie taste as the trial_user and that user's movies with rating 5 is recommended to the trial_user

### Reference

Dataset: The MovieLens 25M Dataset from https://grouplens.org/datasets/movielens/

# Appendix

In [2]:
```
import pandas as pd
import numpy as np
```

## Importing movies.csv and ratings.csv using pandas

In [89]:
```
movies = pd.read_csv('./NLA_Project/ml-25m/movies.csv')
print("Movies dataset")
display(movies)

ratings = pd.read_csv('./NLA_Project/ml-25m/ratings.csv')
del ratings['timestamp']
print('--------------------------------------------------------------------------
print("Ratings dataset")
display(ratings)
```

```
Movies dataset
```

| | movieId | title | genres |
|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy |
| **...** | ... | ... | ... |
| **62418** | 209157 | We (2018) | Drama |
| **62419** | 209159 | Window of the Soul (2001) | Documentary |
| **62420** | 209163 | Bad Poems (2018) | Comedy\|Drama |
| **62421** | 209169 | A Girl Thing (2001) | (no genres listed) |
| **62422** | 209171 | Women of Devil's Island (1962) | Action\|Adventure\|Drama |

62423 rows × 3 columns

```
----------------------------------------------------------------------------
------------
Ratings dataset
```

| | userId | movieId | rating |
|---|---|---|---|
| **0** | 1 | 306 | 3.5 |
| **1** | 1 | 296 | 5.0 |
| **2** | 1 | 307 | 5.0 |
| **3** | 1 | 665 | 5.0 |
| **4** | 1 | 899 | 3.5 |
| **...** | ... | ... | ... |
| **25000090** | 162541 | 50872 | 4.5 |
| **25000091** | 162541 | 55768 | 2.5 |
| **25000092** | 162541 | 56176 | 2.0 |
| **25000093** | 162541 | 58559 | 4.0 |
| **25000094** | 162541 | 63876 | 5.0 |

25000095 rows × 3 columns

## Selecting movies for the trial user

In [91]:
```
sample_movie_set = movies.loc[0:19]
display(sample_movie_set)

''' Creating a list of the ids of the selected movies '''
movie_id_list = list(sample_movie_set[sample_movie_set.columns[0]])
```

| | movieId | title | genres |
|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy |
| **5** | 6 | Heat (1995) | Action\|Crime\|Thriller |
| **6** | 7 | Sabrina (1995) | Comedy\|Romance |
| **7** | 8 | Tom and Huck (1995) | Adventure\|Children |
| **8** | 9 | Sudden Death (1995) | Action |
| **9** | 10 | GoldenEye (1995) | Action\|Adventure\|Thriller |
| **10** | 11 | American President, The (1995) | Comedy\|Drama\|Romance |
| **11** | 12 | Dracula: Dead and Loving It (1995) | Comedy\|Horror |
| **12** | 13 | Balto (1995) | Adventure\|Animation\|Children |
| **13** | 14 | Nixon (1995) | Drama |
| **14** | 15 | Cutthroat Island (1995) | Action\|Adventure\|Romance |
| **15** | 16 | Casino (1995) | Crime\|Drama |
| **16** | 17 | Sense and Sensibility (1995) | Drama\|Romance |
| **17** | 18 | Four Rooms (1995) | Comedy |
| **18** | 19 | Ace Ventura: When Nature Calls (1995) | Comedy |
| **19** | 20 | Money Train (1995) | Action\|Comedy\|Crime\|Drama\|Thriller |

## Creating a matrix for our trial user. Column 1 represents the movieId and column 2 represents the ratings the trial user gave to the respective movies

In [119…

```
# Giving ratings to the movies
trial_user_ratings = [5.0, 5.0, 3.5, 4.0, 2.0, 5.0, 2.0, 4.5, 3.0, 1.0, 2.5,

trial_user = pd.DataFrame(list(zip(movie_id_list, trial_user_ratings)), colu
display(trial_user)
```

| | movieId | rating |
|---|---|---|
| **0** | 1 | 5.0 |
| **1** | 2 | 5.0 |
| **2** | 3 | 3.5 |
| **3** | 4 | 4.0 |
| **4** | 5 | 2.0 |
| **5** | 6 | 5.0 |
| **6** | 7 | 2.0 |
| **7** | 8 | 4.5 |
| **8** | 9 | 3.0 |
| **9** | 10 | 1.0 |
| **10** | 11 | 2.5 |
| **11** | 12 | 5.0 |
| **12** | 13 | 1.0 |
| **13** | 14 | 2.0 |
| **14** | 15 | 3.5 |
| **15** | 16 | 1.0 |
| **16** | 17 | 4.0 |
| **17** | 18 | 2.0 |
| **18** | 19 | 5.0 |
| **19** | 20 | 5.0 |

## Getting a list of the users from the ratings dataset who rated all the same movies as our trial user

In [94]:

```
users_rated = set(ratings[ratings['movieId'] == movie_id_list[0]]['userId'])

for i in range(1,len(movie_id_list)-1):
    movie_id = movie_id_list[i]
    next_movie_id = movie_id_list[i+1]
    users_rated = users_rated & set(ratings[ratings['movieId'] == movie_id][

users_rated = list(users_rated)
print("User ids of all the users who rated movies as the trial user:\n")
display(users_rated)
```

```
User ids of all the users who rated movies as the trial user:

[17794, 57548, 83094, 6039, 122011]
```

## Calculating Euclidean Distance (2-Norm) between the trial user matrix and each of the user's matrix who rated the same movies as the trial user

Storing the calculated norm in the norms array

In [109…

```
norms = []
for i in users_rated:
    r1 = ratings[ratings['userId'] == i]
    r1 = r1[r1['movieId'].isin(movie_id_list)].reset_index(drop=True)
    del r1['userId']
    norms.append(np.linalg.norm(trial_user.to_numpy() - r1.to_numpy()))
```

Sorting the norms array so that the user with the minimum 2-norm is at the $0^{th}$ index

In [110…

```
norms = list(zip(users_rated, norms))
norms.sort(key=lambda x: x[1])

sorted_norm = pd.DataFrame(norms, columns=['userId', 'Euclidean Dist from Tr
display(sorted_norm)
```

| | userId | Euclidean Dist from Trial User |
|---|---|---|
| **0** | 6039 | 7.937254 |
| **1** | 17794 | 8.944272 |
| **2** | 122011 | 9.380832 |
| **3** | 83094 | 9.591663 |
| **4** | 57548 | 12.893797 |

### The movie preferences of our trial user is closest to the user with id: 6039 as the Euclidean distance between trial user's rating matrix and the user is minimum

## Recommending highest rated movies from user 6039's list to trial user

In [118…
```
recommendations = []
closest_user_movies = ratings[ratings['userId'] == norms[0][0]]
cu = closest_user_movies.to_numpy()
recommendations = []

for r in cu:
    if r[2] == 5:
        df = movies[movies['movieId'] == r[1]]
        recommendations.append(df.to_numpy()[0][1])
print(len(recommendations), 'movies recommended\n\n')

for i in enumerate(recommendations,1):
    print(str(i[0])+'.',i[1])
```

```
102 movies recommended


1. Toy Story (1995)
2. Waiting to Exhale (1995)
3. Heat (1995)
4. Sudden Death (1995)
5. GoldenEye (1995)
6. Dracula: Dead and Loving It (1995)
7. Nixon (1995)
8. Ace Ventura: When Nature Calls (1995)
9. Othello (1995)
10. Dangerous Minds (1995)
11. Clueless (1995)
12. Richard III (1995)
13. Dead Presidents (1995)
14. Seven (a.k.a. Se7en) (1995)
15. Pocahontas (1995)
16. Usual Suspects, The (1995)
17. Lawnmower Man 2: Beyond Cyberspace (1996)
18. Misérables, Les (1995)
19. Bed of Roses (1996)
20. Screamers (1995)
21. Juror, The (1996)
22. Braveheart (1995)
23. Anne Frank Remembered (1995)
24. Race the Sun (1996)
25. Up Close and Personal (1996)
26. Batman Forever (1995)
27. Canadian Bacon (1995)
28. Clockers (1995)
29. Congo (1995)
30. Desperado (1995)
31. Die Hard: With a Vengeance (1995)
32. First Knight (1995)
33. Reckless (1995)
34. Something to Talk About (1995)
35. Species (1995)
36. To Wong Foo, Thanks for Everything! Julie Newmar (1995)
```

```
37. Cure, The (1995)
38. Don Juan DeMarco (1995)
39. Dumb & Dumber (Dumb and Dumber) (1994)
40. Gumby: The Movie (1995)
41. Little Women (1994)
42. Legends of the Fall (1994)
43. Mary Shelley's Frankenstein (Frankenstein) (1994)
44. Perez Family, The (1995)
45. Pulp Fiction (1994)
46. Swan Princess, The (1994)
47. Stargate (1994)
48. Shawshank Redemption, The (1994)
49. To Live (Huozhe) (1994)
50. Walking Dead, The (1995)
51. Virtuosity (1995)
52. While You Were Sleeping (1995)
53. Ace Ventura: Pet Detective (1994)
54. Client, The (1994)
55. Crow, The (1994)
56. Forrest Gump (1994)
57. Higher Learning (1995)
58. Jungle Book, The (1994)
59. Lion King, The (1994)
60. Wes Craven's New Nightmare (Nightmare on Elm Street Part 7: Freddy's Fin
ale, A) (1994)
61. Mask, The (1994)
62. Naked Gun 33 1/3: The Final Insult (1994)
63. Speed (1994)
64. True Lies (1994)
65. Brother Minister: The Assassination of Malcolm X (1994)
66. Addams Family Values (1993)
67. Cliffhanger (1993)
68. With Honors (1994)
69. Hot Shots! Part Deux (1993)
70. In the Line of Fire (1993)
71. Kalifornia (1993)
72. Poetic Justice (1993)
73. Robin Hood: Men in Tights (1993)
74. Schindler's List (1993)
75. Terminal Velocity (1994)
76. Nightmare Before Christmas, The (1993)
77. Three Musketeers, The (1993)
78. Trial by Jury (1994)
79. Ghost (1990)
80. Terminator 2: Judgment Day (1991)
81. Beauty and the Beast (1991)
82. Candyman: Farewell to the Flesh (1995)
83. Last Supper, The (1995)
84. Diabolique (1996)
85. Mission: Impossible (1996)
86. Kids in the Hall: Brain Candy (1996)
87. Space Jam (1996)
88. Twister (1996)
```

```
89. Thinner (1996)
90. Phantom, The (1996)
91. Independence Day (a.k.a. ID4) (1996)
92. Hunchback of Notre Dame, The (1996)
93. Time to Kill, A (1996)
94. Convent, The (O Convento) (1995)
95. Island of Dr. Moreau, The (1996)
96. Fly Away Home (1996)
97. That Thing You Do! (1996)
98. William Shakespeare's Romeo + Juliet (1996)
99. Candidate, The (1972)
100. English Patient, The (1996)
101. Mirror Has Two Faces, The (1996)
102. Crucible, The (1996)
```