Chapter 4

Project 7: Norms, angles, and your movie choices

4.1 Introduction

Predicting user preferences is a lucrative business opportunity, driving innovation in recommendation systems across various internet services like Netflix, Pandora, and Amazon. These platforms invest significantly in enhancing algorithms to analyze consumer behavior and suggest personalized products or content. For example, Netflix famously offered a prize to improve its recommendation algorithm by 10%. Pandora's Music Genome Project rates songs across 450 categories, allowing it to suggest music based on mathematical comparisons of these vectors. Similarly, in this project, we'll leverage linear algebra to identify similarities in user tastes using the MovieLens database, comprising millions of ratings by thousands of users. By comparing users' ratings, we aim to recommend items favored by users with similar tastes, employing techniques such as Euclidean distance, scalar product, and correlation coefficients.

4.2 Theories

4.2.1 Norms

A norm is a function that assigns a strictly positive length or size to each vector in a vector space—except for the zero vector, which is assigned a length of zero. Norms are a way to measure the magnitude or "length" of vectors, and they play a crucial role in various mathematical and applied contexts, such as numerical analysis, optimization, and machine learning.

Euclidean Norm (L2 Norm):

The Euclidean norm, also known as the L2 norm, is the most common norm and is defined as the square root of the sum of the squares of the vector components. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the Euclidean norm is given by:

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

This norm corresponds to the usual notion of distance in Euclidean space.

Manhattan Norm (L1 Norm)

The Manhattan norm, or L1 norm, is defined as the sum of the absolute values of the vector components:

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

This norm is useful in various optimization problems and is also known as the taxicab norm.

Maximum Norm (L Norm)

The maximum norm, or L norm, is defined as the maximum absolute value among the vector components:

$$\|\mathbf{x}\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|)$$

This norm measures the largest magnitude among the components of the vector.

p-Norm

The p-norm generalizes the L1, L2, and L norms and is defined for any positive real number p:

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$$

For $p = 2$, it becomes the Euclidean norm, and for $p = 1$, it becomes the Manhattan norm.

Properties of Norms

For a norm $\|\cdot\|$ on a vector space, the following properties hold:

- **Non-negativity:** $\|\mathbf{x}\| \geq 0$ for all vectors $\mathbf{x}$, and $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$.
- **Scalability:** For any scalar α and vector $\mathbf{x}$, $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$.
- **Triangle Inequality:** For any vectors $\mathbf{x}$ and $\mathbf{y}$, $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.
- **Zero Vector:** $\|\mathbf{0}\| = 0$.

4.2.2 Angles

The concept of angles between vectors is closely related to the idea of similarity and orthogonality of vectors. The angle between two vectors can be quantified using the dot product (also known as the inner product) and norms (magnitudes) of the vectors.

Definition and Calculation

The angle θ between two non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ in an n-dimensional space can be found using the cosine of the angle, which is derived from the dot product of the vectors:

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

where:

- $\mathbf{u} \cdot \mathbf{v}$ is the dot product of $\mathbf{u}$ and $\mathbf{v}$.

- $\|\mathbf{u}\|$ is the norm (magnitude) of $\mathbf{u}$.
- $\|\mathbf{v}\|$ is the norm (magnitude) of $\mathbf{v}$.

Dot Product

The dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as:

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Norm (Magnitude)

The norm of a vector $\mathbf{u}$ is defined as:

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}$$

4.2.3 Pearson correlation coefficient

The Pearson correlation coefficient is a measure of the linear correlation between two sets of data. It provides a value between -1 and 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship. In linear algebra, this concept is particularly useful for comparing vectors that represent data points, such as user ratings in a recommender system.

Definition: For two vectors $\mathbf{x}$ and $\mathbf{y}$ representing two sets of data, the Pearson correlation coefficient r is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- x_i and y_i are the elements of vectors $\mathbf{x}$ and $\mathbf{y}$, respectively.
- $\bar{x}$ and $\bar{y}$ are the means of the vectors $\mathbf{x}$ and $\mathbf{y}$.

Calculation Steps:

1. Compute the mean of each vector:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

2. Compute the numerator:

$$\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

3. Compute the denominator:

$$\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

4. Divide the numerator by the denominator to get the Pearson correlation coefficient.

4.3 Procedure

4.3.1 Section 1

Load the data from the file ”users_movies.mat” using the ”load” command. The matrix ”users_movies” should have dimensions 6040×3952, with integer values ranging from 0 to 5. A rating of 1 represents ”strongly dislike”, while 5 represents ”strongly like”. A rating of 0 indicates that the user did not rate the movie. The array ”movies” contains the titles of all the movies. The matrix ”users_movies_sort” is a subset of ”users_movies”, containing ratings for the 20 most popular movies. The indexes of these popular movies are stored in the array ”index_small”. Finally, the vector ”trial_user” contains ratings of these popular movies by another user who is not part of the database. It is recommended to view all variables and their dimensions using the ”Workspace” window in the MATLAB environment.

```
%% Load the data
clear;
load('users_movies.mat','movies','users_movies','users_movies_sort','
    index_small','trial_user');
```

```
[m,n]=size(users_movies);
```

Listing 4.1: Load the data

- **Movies:** The array *movies* contains all the titles of the movies.
- **Users_movies_sort:** The matrix *users_movies_sort* contains an extract from the matrix *users_movies.mat* with ratings for the 20 most popular movies selected.
- **Index_small:** The indexes of these popular movies are recorded in the array *index_small*.
- **Trial_user:** Ratings of these popular movies by yet another user (not any of the users contained in the database) are given by the vector *trial_user*.

4.3.2 Section 2

Print the titles of the 20 most popular movie:

```
%% Load the data
clear;
fprintf('Rating is based on movies:\n')
for j=1:length(index_small)
    fprintf('%s \n',movies{index_small(j)})
end
fprintf('\n')
```

Listing 4.2: MATLAB's code

```
Rating is based on movies:
E.T. the Extra-Terrestrial (1982)
Star Wars Episode IV - A New Hope (1977)
Star Wars Episode V - The Empire Strikes Back (1980)
Star Wars Episode VI - Return of the Jedi (1983)
Jurassic Park (1993)
Saving Private Ryan (1998)
Terminator 2
Matrix, The (1999)
Back to the Future (1985)
```

```
Silence of the Lambs, The (1991)
Star Wars Episode I - The Phantom Menace (1999)
Raiders of the Lost Ark (1981)
Fargo (1996)
Sixth Sense, The (1999)
Braveheart (1995)
Shakespeare in Love (1998)
Princess Bride, The (1987)
Schindler's List (1993)
Shawshank Redemption, The (1994)
Groundhog Day (1993)
```

4.3.3 Section 3

```
%% Select the users to compare to
[m1,n1]=size(users_movies_sort);
ratings=[];
for j=1:m1
  if prod(users_movies_sort(j,:))~=0
      ratings=[ratings; users_movies_sort(j,:)];
  end
end
```

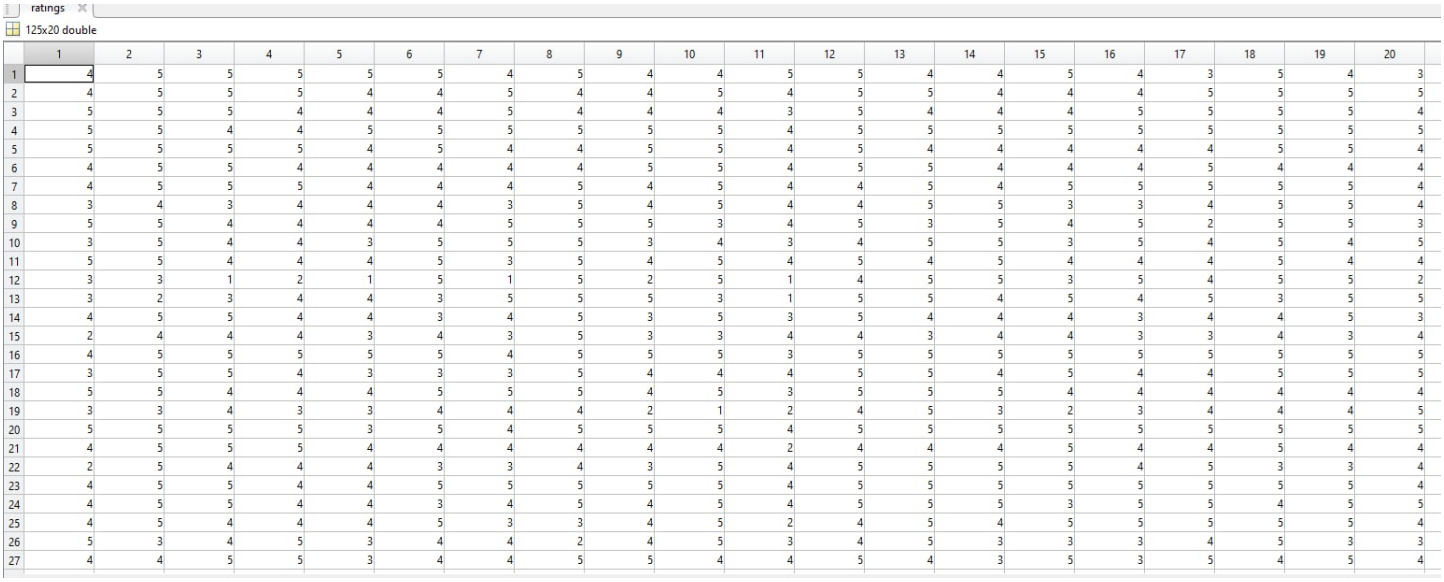
Listing 4.3: Select the people who rated all of the 20 movies under consideration

Explanation

prod: accumulates all elements in row j of array user movies sort.
Ratings: array is used to filter out people according to the test requirements.
Command: if the product of all elements in row j of array user movies sort is not 0, that row will be copied through ratings.

Question 1:What does the command ratings=[] do?

The command `ratings = [];` initializes an empty array named `ratings`. This means that `ratings` starts with no elements. This setup is commonly used to collect or build up data in subsequent operations, especially within loops.



ratings

125x20 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4	5	5	5	5	5	4	5	4	4	5	5	4	4	5	4	3	5	4	3
2	4	5	5	5	4	4	5	4	4	5	4	5	5	4	4	4	5	5	5	5
3	5	5	5	4	4	4	5	4	4	4	3	5	4	4	4	5	5	5	5	4
4	5	5	4	4	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5
5	5	5	5	5	4	5	4	4	5	5	4	5	4	4	4	4	4	5	5	4
6	4	5	5	4	4	4	4	4	5	5	4	5	5	4	4	4	5	4	4	4
7	4	5	5	5	4	4	4	5	4	5	4	4	5	4	5	5	5	5	5	4
8	3	4	3	4	4	4	3	5	4	5	4	4	5	5	3	3	4	5	5	4
9	5	5	4	4	4	4	5	5	5	3	4	5	3	5	4	5	2	5	5	3
10	3	5	4	4	3	5	5	5	3	4	3	4	5	5	3	5	4	5	4	5
11	5	5	4	4	4	5	3	5	4	5	4	5	4	5	4	4	4	5	4	4
12	3	3	1	2	1	5	1	5	2	5	1	4	5	5	3	5	4	5	5	2
13	3	2	3	4	4	3	5	5	5	3	1	5	5	4	5	4	5	3	5	5
14	4	5	5	4	4	3	4	5	3	5	3	5	4	4	4	3	4	4	5	3
15	2	4	4	4	3	4	3	5	3	3	4	4	3	4	4	3	3	4	3	4
16	4	5	5	5	5	5	4	5	5	5	3	5	5	5	5	5	5	5	5	5
17	3	5	5	4	3	3	3	5	4	4	4	5	5	4	5	4	4	5	5	5
18	5	5	4	4	4	5	5	5	4	5	3	5	5	5	4	4	4	4	4	4
19	3	3	4	3	3	4	4	4	2	1	2	4	5	3	2	3	4	4	4	5
20	5	5	5	5	3	5	4	5	5	5	4	5	5	5	5	5	5	5	5	5
21	4	5	5	5	4	4	4	4	4	4	2	4	4	4	5	4	4	5	4	4
22	2	5	4	4	4	3	3	4	3	5	4	5	5	5	5	4	5	3	3	4
23	4	5	5	4	4	5	5	5	5	5	4	5	5	5	5	5	5	5	5	4
24	4	5	5	4	4	3	4	5	4	5	4	5	5	5	3	5	5	4	5	5
25	4	5	4	4	4	5	3	3	4	5	2	4	5	4	5	5	5	5	5	4
26	5	3	4	5	3	4	4	2	4	5	3	4	5	3	3	3	4	5	3	3
27	4	4	5	5	3	4	4	5	5	4	4	5	4	3	5	3	5	4	5	4

Figure 4.1: Rating

`ratings = [];` sets up an empty array where user rating data will be accumulated. The code then iterates over `users_movies_sort` and adds rows to `ratings` for users who have rated all the movies (i.e., rows without any zero entries).

Here's a detailed breakdown:

1. **Initialization**: `ratings = [];` creates an empty array, so initially, `ratings` has no rows or columns.

2. **Loop through users**: The `for` loop goes through each user (row) in `users_movies_sort`.

3. **Check condition**: For each user, `if prod(users_movies_sort(j,:)) ≠ 0` checks if all ratings in the user's row are non-zero.

   - `users_movies_sort(j,:)` retrieves the ratings for the j-th user.
   - `prod(users_movies_sort(j,:))` calculates the product of these ratings.
   - If any rating is zero, the product will be zero; otherwise, it will be non-zero.

4. **Append row**: If the condition is met (all ratings are non-zero), the user's ratings are added to the `ratings` array using `ratings = [ratings; users_movies_sort(j`

By the end of the loop, `ratings` contains only the rows (users) from `users_movies_sort` where all the ratings are non-zero. This filtered array can then be used for further analysis or comparisons.

4.3.4 Section 4

Find the eucliden distance:

```
%% Find the Euclidean distances
[m2,n2]=size(ratings);
eucl = zeros(m2, 1);
for i=1:m2
    eucl(i)=norm(ratings(i,:)-trial_user);
end
```

Listing 4.4: Matlab's code

4.3.5 Section 5

Find the person closest to the eucl number using the sort command and save closest_user_dist:

```
[MinDist,DistIndex]=sort(eucl,'ascend');
closest_user_Dist=DistIndex(1);
disp('closest Euclidean distance:');
disp(closest_user_Dist);
```

Listing 4.5: Matlab's code

```
closest Euclidean distance:
    14
```

4.3.6 Section 6

To address the limitations of Euclidean distance in comparing user ratings, we turn to the Pearson correlation coefficient, a statistical metric. Unlike Euclidean distance, which might wrongly assess dissimilarity between users with similar

tastes but differing rating tendencies, the Pearson correlation coefficient considers both the average user opinion and the variability in their rating behavior. It calculates the correlation between two vectors of ratings, accounting for how users tend to judge movies either harshly or enthusiastically. Geometrically, the coefficient corresponds to the cosine of the angle between the vectors, where a smaller angle indicates greater similarity in opinions. To compute the Pearson correlation coefficient, we first centralize the columns of the ratings matrix and the trial user vector.3.5

```
%% Centering the ratings
ratings_cent=ratings-mean(ratings,2)*ones(1,n2);
trial_user_cent=trial_user-mean(trial_user);
```

Listing 4.6: Matlab's code

`mean(ratings,2)`: calculates the average value of the elements along the rows in the matrix.

`ones(1,n2)`: generates an identity matrix of size $n2$.

`ratings_cent`: a matrix containing each element as the initial value in `ratings` minus the average value of the equivalent row.

`mean(trial_user)`: computes the average value of array elements.

`trial_user_cent`: a matrix containing each element as the initial value in `trial_user` minus the average value.

4.3.7 Section 7

Use the for... end loop to compute the Pearson correlation coefficients between the rows of the matrix ratings and the vector trial user. Save the result as a vector pearson.

```
%% Compute the Pearson correlation coefficients (sec 7)
pearson = zeros(m2, 1);

% Ensure trial_user is a row vector
trial_user = trial_user(:)';
```

```
6
7  for i = 1:m2
8      % Extract the current row from ratings
9      current_user = ratings(i, :);
10
11     % Compute the means
12     mean_current_user = mean(current_user);
13     mean_trial_user = mean(trial_user);
14
15     % Subtract the means from the vectors
16     centered_current_user = current_user - mean_current_user;
17     centered_trial_user = trial_user - mean_trial_user;
18
19     % Compute the numerator and the denominators for the Pearson
       correlation coefficient
20     numerator = sum(centered_current_user .* centered_trial_user);
21     denominator = sqrt(sum(centered_current_user .^ 2) * sum(
       centered_trial_user .^ 2));
22
23     % Compute the Pearson correlation coefficient
24     if denominator ~= 0
25         correlation = numerator / denominator;
26     else
27         correlation = 0; % Handle the case where the denominator is 0
28     end
29
30     % Store the result in the pearson vector
31     pearson(i) = correlation;
32 end
```

Listing 4.7: Matlab's code

Pearson is a column vector of size $m2 \times 1$, containing Pearson correlation coefficients between **trial_user** and each other user.

Ensure **trial_user** is a row vector:

```
trial_user=trial_user(:)';
```

This line converts **trial_user** into a row vector if it is not already.

Loop through each user in **ratings**:

```
for i=1:m2
```

This loop iterates through all rows of the **ratings** matrix.

Extract the current row from **ratings**:

```
current_user=ratings(i, :);
```

current_user is the *i*-th row of the **ratings** matrix.

Compute the mean values of the vectors:

```
mean_current_user=mean(current_user);
mean_trial_user=mean(trial_user);
```

Subtract the mean values from the vectors:

```
centered_current_user=current_user-mean_current_user;
centered_trial_user=trial_user-mean_trial_user;
```

centered_current_user and **centered_trial_user** are the vectors with the mean values subtracted.

Compute the numerator and denominator for the Pearson correlation coefficient:

```
numerator = sum(centered_current_user .* centered_trial_user);
denominator = sqrt(sum(centered_current_user .^ 2) * sum(centered_trial
```

numerator is the sum of the element-wise products of the centered vectors. **denominator** is the product of the square roots of the sums of the squared centered vectors.

Compute the Pearson correlation coefficient:

```
if denominator ~= 0
    correlation = numerator / denominator;
else
    correlation = 0;
end
```

If **denominator** is not equal to 0, the Pearson correlation coefficient is computed by dividing **numerator** by **denominator**. If **denominator** is equal to 0, the Pearson correlation coefficient is set to 0 to avoid division by zero.

Save the result into the **pearson** vector:

```
    pearson(i) = correlation;
```

This code calculates the Pearson correlation coefficients between each user in the **ratings** matrix and the trial user **trial_user**. The process involves: extracting each user, computing the mean values and subtracting them from the vectors, computing the numerator and denominator of the Pearson correlation coefficient formula, and saving the result into the **pearson** vector. The Pearson correlation coefficient indicates the degree of linear correlation between the movie ratings of **trial_user** and each other user, ranging from -1 to 1.

4.3.8 Section 8

The Pearson correlation coefficient $r(x,y)$ lies in the range $(-1,1)$, where a coefficient closer to 1 signifies higher similarity in user tastes. Next, let's arrange the vector pearson using the `sort` function as previously instructed. Save the outcomes of this function as `[MaxPearson, PearsonIndex]`, where the maximal correlation coefficient will be the first element in the matrix `MaxPearson`. Save this element as `closest_user_Pearson`.

```
% Sort the pearson vector in descending order(sec 8)
[MaxPearson, PearsonIndex] = sort(pearson, 'descend');
% Find the user with the highest Pearson correlation (index of the first
% element in PearsonIndex)(sec 8)
closest_user_Pearson = PearsonIndex(1);
% Display the results
disp('Closest user Pearson index:');
disp(closest_user_Pearson);
```

Listing 4.8: Matlab's code

Here is the result:

```
Closest user Pearson index:
    88
```

4.3.9 Section 9

Compare the elements of the vectors DistIndex, PearsonIndex:

Question 2:Are the variables closest user Pearson and closest user Dist the same?

No, the variables `closest_user_Pearson` and `closest_user_Dist` are distinct and serve different purposes. They are identified using two different methods for measuring similarity between the trial user (`trial_user`) and other users in the dataset.

closest_user_Pearson:

This variable represents the user most similar to `trial_user` based on the Pearson correlation coefficient. The Pearson correlation coefficient measures the linear correlation between two datasets, ranging from -1 to 1. The user with the highest Pearson correlation coefficient with `trial_user` (i.e., closest to 1) is considered the user with the most similar preferences to `trial_user`.

closest_user_Dist:

This variable represents the user with the smallest Euclidean distance to `trial_user`. The Euclidean distance measures the straight-line distance between two points in multi-dimensional space. The user with the smallest Euclidean distance to `trial_user` is considered the user closest in terms of numerical ratings to `trial_user`.

Summary:

`closest_user_Pearson`: The user with the highest Pearson correlation coefficient with `trial_user`, indicating the user whose movie rating tendencies are most similar to `trial_user`. `closest_user_Dist`: The user with the smallest Euclidean distance to `trial_user`, indicating the user whose movie ratings are numerically closest to `trial_user`. These two variables are generally not the same because they measure similarity in different ways. The Pearson correlation coefficient focuses on linear correlation, while the Euclidean distance measures numerical closeness in multi-dimensional space.

4.3.10 Section 10

Recommendations:

```
%% Recommendations
recommend_dist = [];
for k = 1:n
    if users_movies(closest_user_Dist, k) == 5
        recommend_dist = [recommend_dist; k];
    end
end

recommend_Pearson = [];
for k = 1:n
    if users_movies(closest_user_Pearson, k) == 5
        recommend_Pearson = [recommend_Pearson; k];
    end
end

liked = [];
for k = 1:20
    if trial_user(k) == 5
        liked = [liked; index_small(k)];
    end
end
```

Listing 4.9: Matlab's code

Initialize an empty array to store recommendations based on Euclidean distance:

```
recommend_dist = [];
```

`recommend_dist` is an empty array used to store indices of movies that the nearest user according to Euclidean distance has rated as 5.

Search for recommendations based on Euclidean distance:

```
disp('Recommendations based on Euclidean distance:');
for i = 1:length(recommend_dist)
    fprintf('%s \n', movies{recommend_dist(i)});
end
```

Listing 4.10: Matlab's code

This loop iterates through all movies (k from 1 to n). It checks if the nearest user according to Euclidean distance (`closest_user_Dist`) has rated movie k as 5. If so, it adds index k to the `recommend_dist` array.

The Pearson function is similar to Euclidean:

```
disp('Recommendations based on Pearson correlation:');
for i = 1:length(recommend_Pearson)
    fprintf('%s \n', movies{recommend_Pearson(i)});
end
```

Listing 4.11: Matlab's code

Initialize an empty array to store movies that trial_user has rated highly:

```
liked = [];
```

`liked` is an empty array used to store indices of movies in the `index_small` array that trial_user has rated as 5.

Find movies that trial_user has rated highly (5 points):

```
disp('Movies liked by trial user:');
for i = 1:length(liked)
    fprintf('%s \n', movies{liked(i)});
end
```

Listing 4.12: Matlab's code

This loop iterates through the first 20 movies (k from 1 to 20). It checks if trial_user has rated movie k as 5. If so, it adds index `index_small(k)` to the `liked` array. `index_small` contains indices of movies in the smaller array used for comparison.

Summary: `recommend_dist`: Contains indices of movies that the nearest user according to Euclidean distance has rated as 5. `recommend_Pearson`: Contains indices of movies that the nearest user according to Pearson correlation coefficient has rated as 5. `liked`: Contains indices of movies in the `index_small` array that trial_user has rated as 5. These arrays are used to suggest new movies to trial_user based on ratings from similar users.

4.3.11 Section 11

```
%% Display Recommendations and Liked Movies
disp('Movies liked by trial user:');
for i = 1:length(liked)
  fprintf('%s \n', movies{liked(i)});
end

disp('Recommendations based on Euclidean distance:');
for i = 1:length(recommend_dist)
  fprintf('%s \n', movies{recommend_dist(i)});
end

disp('Recommendations based on Pearson correlation:');
for i = 1:length(recommend_Pearson)
  fprintf('%s \n', movies{recommend_Pearson(i)});
end
```

Listing 4.13: Matlab's code

Iterate through each of the functions created in the previous sections such as `likeds`, `recommend_dist`, `recommend_Pearson` to find a list of favorite movie genres based on the trial user, Euclidean distance, and Pearson correlation.

MOVIES LIKED BY TRIAL USER:

```
Movies liked by trial user:
Star Wars Episode IV - A New Hope (1977)
Star Wars Episode V - The Empire Strikes Back (1980)
Star Wars Episode VI - Return of the Jedi (1983)
Matrix, The (1999)
Silence of the Lambs, The (1991)
Raiders of the Lost Ark (1981)
Groundhog Day (1993)
```

RECOMMENDATIONS BASED ON EUCLIDEAN DISTANCE:

```
Recommendations based on Euclidean distance:
Pulp Fiction (1994)
Deer Hunter, The (1978)
Red Violin, The (Le Violon rouge) (1998)
Sixth Sense, The (1999)
```

```
Children of Paradise (Les enfants du paradis) (1945)
Being John Malkovich (1999)
```

RECOMMENDATIONS BASED ON PEARSON CORRELATION:

```
Recommendations based on Pearson correlation:
Taxi Driver (1976)
Schindler's List (1993)
Fargo (1996)
Godfather, The (1972)
North by Northwest (1959)
Casablanca (1942)
Citizen Kane (1941)
Mr. Smith Goes to Washington (1939)
Bonnie and Clyde (1967)
Bob Roberts (1992)
Paris Is Burning (1990)
12 Angry Men (1957)
To Kill a Mockingbird (1962)
Title not available
Grand Day Out, A (1992)
Raging Bull (1980)
Annie Hall (1977)
Stand by Me (1986)
Killing Fields, The (1984)
My Life as a Dog (Mitt liv som hund) (1985)
Tickle in the Heart, A (1996)
Boys, Les (1997)
There's Something About Mary (1998)
On the Waterfront (1954)
Ordinary People (1980)
Chariots of Fire (1981)
Rain Man (1988)
Saving Private Ryan (1998)
Life Is Beautiful (La Vita    bella) (1997)
Risky Business (1983)
Brief Encounter (1946)
Shower (Xizhao) (1999)
```

4.3.12 Section 12

```
%% Create myratings vector
% Define your own ratings for the 20 popular movies
myratings = [5, 4, 3, 2, 1, 5, 3, 4, 2, 1, 5, 4, 3, 2, 5, 1, 3, 4, 2,
   5];

% Assign random ratings for movies you haven't seen (e.g., pick some
   indices randomly)
unseen_indices = randi([1 20], 1, 5);  % Assuming 5 movies are unseen
myratings(unseen_indices) = randi([1, 5], 1, length(unseen_indices));

% Ensure myratings is a row vector
myratings = myratings(:)';

fprintf('My ratings for the 20 popular movies:\n');
for j=1:length(myratings)
  fprintf('Movie: %s, Rating: %d \n', movies{index_small(j)}, myratings(
   j));
end
fprintf('\n')
```

Listing 4.14: myratings

Create a personal rating vector (`myratings`):

```
myratings = [5, 4, 3, 2, 1, 5, 3, 4, 2, 1, 5, 4, 3, 2, 5, 1, 3, 4, 2, 5];
```

This line defines your personal ratings for 20 popular movies. Each number represents a rating from 1 to 5 for a movie.

Assign random ratings to unseen movies:

```
unseen_indices = randi([1 20], 1, 5);
myratings(unseen_indices) = randi([1, 5], 1, length(unseen_indices));
```

- `unseen_indices = randi([1 20], 1, 5)`: Generates an array of 5 random indices in the range from 1 to 20, representing the movies that have not been seen.
- `myratings(unseen_indices) = randi([1, 5], 1, length(unseen_indices))`: Assigns random ratings from 1 to 5 to the unseen movies.

```
My ratings for the 20 popular movies:
Movie: E.T. the Extra-Terrestrial (1982), Rating: 5
Movie: Star Wars Episode IV - A New Hope (1977), Rating: 4
Movie: Star Wars Episode V - The Empire Strikes Back (1980), Rating: 3
Movie: Star Wars Episode VI - Return of the Jedi (1983), Rating: 2
Movie: Jurassic Park (1993), Rating: 4
Movie: Saving Private Ryan (1998), Rating: 1
Movie: Terminator 2, Rating: 3
Movie: Matrix, The (1999), Rating: 4
Movie: Back to the Future (1985), Rating: 2
Movie: Silence of the Lambs, The (1991), Rating: 1
Movie: Star Wars Episode I - The Phantom Menace (1999), Rating: 5
Movie: Raiders of the Lost Ark (1981), Rating: 4
Movie: Fargo (1996), Rating: 3
Movie: Sixth Sense, The (1999), Rating: 2
Movie: Braveheart (1995), Rating: 5
Movie: Shakespeare in Love (1998), Rating: 1
Movie: Princess Bride, The (1987), Rating: 2
Movie: Schindler's List (1993), Rating: 4
Movie: Shawshank Redemption, The (1994), Rating: 3
Movie: Groundhog Day (1993), Rating: 5
```

4.3.13 Section 13

```
%% Personal Recommendations using myratings

% Ensure myratings is a row vector
myratings = myratings(:)';

%% Select the users to compare to
[m1, n1] = size(users_movies_sort);
ratings = [];
for j = 1:m1
  if prod(users_movies_sort(j, :)) ~= 0
      ratings = [ratings; users_movies_sort(j, :)];
  end
end

%% Find the Euclidean distances
[m2, n2] = size(ratings);
eucl = zeros(m2, 1);
for i = 1:m2
```

```
  eucl(i) = norm(ratings(i, :) - myratings);
end

[MinDist, DistIndex] = sort(eucl, 'ascend');
closest_user_Dist = DistIndex(1);

%% Centering the ratings
ratings_cent = ratings - mean(ratings, 2) * ones(1, n2);
myratings_cent = myratings - mean(myratings);

%% Compute the Pearson correlation coefficients
pearson = zeros(m2, 1);

for i = 1:m2
  % Extract the current row from ratings
  current_user = ratings(i, :);

  % Compute the means
  mean_current_user = mean(current_user);
  mean_myratings = mean(myratings);

  % Subtract the means from the vectors
  centered_current_user = current_user - mean_current_user;
  centered_myratings = myratings - mean_myratings;

  % Compute the numerator and the denominators for the Pearson
   correlation coefficient
  numerator = sum(centered_current_user .* centered_myratings);
  denominator = sqrt(sum(centered_current_user .^ 2) * sum(
   centered_myratings .^ 2));

  % Compute the Pearson correlation coefficient
  if denominator ~= 0
      correlation = numerator / denominator;
  else
      correlation = 0; % Handle the case where the denominator is 0
  end

  % Store the result in the pearson vector
  pearson(i) = correlation;
end

% Sort the pearson vector in descending order
```

```
[MaxPearson, PearsonIndex] = sort(pearson, 'descend');

% Find the user with the highest Pearson correlation (index of the first
    element in PearsonIndex)
closest_user_Pearson = PearsonIndex(1);


%% Recommendations based on myratings
recommend_dist = [];
for k = 1:n
  if users_movies(closest_user_Dist, k) == 5
      recommend_dist = [recommend_dist; k];
  end
end

recommend_Pearson = [];
for k = 1:n
  if users_movies(closest_user_Pearson, k) == 5
      recommend_Pearson = [recommend_Pearson; k];
  end
end

liked = [];
for k = 1:20
  if myratings(k) == 5
      liked = [liked; index_small(k)];
  end
end

%% Display Recommendations and Liked Movies
disp('Movies liked by user (myratings):');
for i = 1:length(liked)
  fprintf('%s \n', movies{liked(i)});
end

disp('Recommendations based on Euclidean distance (myratings):');
for i = 1:length(recommend_dist)
  fprintf('%s \n', movies{recommend_dist(i)});
end

disp('Recommendations based on Pearson correlation (myratings):');
for i = 1:length(recommend_Pearson)
  fprintf('%s \n', movies{recommend_Pearson(i)});
```

```
102 end
```

Listing 4.15: Matlab's code

```
Movies liked by user (myratings):
E.T. the Extra-Terrestrial (1982)
Star Wars Episode I - The Phantom Menace (1999)
Braveheart (1995)
Groundhog Day (1993)
Recommendations based on Euclidean distance (myratings):
Braveheart (1995)
Frequency (2000)
Gladiator (2000)
Me, Myself and Irene (2000)
Recommendations based on Pearson correlation (myratings):
Crimson Tide (1995)
Star Wars Episode IV - A New Hope (1977)
Shawshank Redemption, The (1994)
Fugitive, The (1993)
Schindler's List (1993)
Princess Bride, The (1987)
Cyrano de Bergerac (1990)
Parenthood (1989)
```

Bibliography

[1] https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-spatialfilters-convolution/

[2] L. GARCIA AND C. PENLAND, MATLAB Projects for Scientists and Engineers, Prentice Hall, Upper Saddle River, NJ, 1996.

[3] University of South Florida, Chapter 5, Edge Detection https://www.cse.usf.edu/ r1k/MachineVisionBook/MachineVision.files /MachineVisionChapter5.pdf

[4] https://blog.demofox.org/2022/02/26/image-sharpening-convolution-kernels/

[5] The University of Edinburgh, Spatial Filters – Mean Filters, https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm

[6] https://www.youtube.com/watch?v=uNP6ZwQ3r6Asi=C3QoKlUGx1mUsuBM (Edge Detection Using Laplacian)

[7] Linear Algebra and Its Applications (Gilbert Strang)