Math 551 Fall 2018
Lab 9

Goal: Use a norm and an inner product to detect similarity between users' choices and to create a simple recommender system.

Getting started: Open a new Matlab script and save it as "m551lab09.m".

- Open a new Matlab script and save it as "m551lab09.m".
- Download the file "users movies.mat" which contains the arrays `movies`, `users_movies`, `users_movies_sort`, `index_small`, `trial_user` and place it in your working directory.

What you have to submit: The file `m551lab09.m` which you will create during the lab session.

Reminders About the Grading Software

Remember, the labs are graded with the help of software tools. If you do not follow the instructions, you will lose points. If the instructions ask you to assign a value to a variable, you must **use the variable name given in the instructions**. If the instructions ask you to make a variable named `x1` and instead you make a variable named `x` or `X` or `X1` or any name other than `x1`, the grading software will not find your variable and you *will* lose points. Required variables are listed in the lab instructions in a gray box like this:

Variables: `x1`, `A`, `q`

At times you will also be asked to answer questions in a comment in your M-file. You must **format your text answers correctly**. Questions that you must answer are shown in gray boxes in the lab. For example, if you see the instruction

Q7: What does the Matlab command `lu` do?

your file must contain a line similar to the following somewhere

```
% Q7: It computes the LU decomposition of a matrix.
```

The important points about the formatting are

- The answer is on a single line.
- The first characters on the line is the comment character `%`.
- The answer is labeled in the form `Q`n`:`, where n stands for the question number from the gray box.

If you do not format your answers correctly, the grading software will not find them and you *will* lose points.

Introduction

Being able to predict users' choices is a big business. Many internet services are studying your choices and preferences to be able to offer you the products you might like (and correspondingly the products you are more likely to buy). Netflix, Pandora radio, Spotify, Facebook targeted ads, and even online dating websites are all examples of recommender systems. These services are sometimes willing to go to great lengths in order to improve their algorithms. In fact, in 2006, Netflix offered 1 million dollars to the team which would be able to improve their Cinematch algorithm predictions by just 10%. Music Genome Project which is a trademark of Pandora radio is another example of a recommender system. Every song is assigned a rating from 0 to 5 in 450 different categories ("genes"). Thus, any song is represented by a vector with 450 components. Mathematical comparison of these vectors allows then to suggest the next song to a user.

Observe that data about movies, consumer products, and etc, can be often arranged in a form of a vector. These vector representations can be then used in different algorithms to compare items and predict similarity. In this project, we will use linear algebra to establish similarities in tastes between different users. The main idea is that if we know the current ratings from a certain user, then comparing it with the ratings of other users in database, we can find users with similar tastes. Finally, we can look at the items highly rated by those users and suggest these items to the current user. In this lab, we will look at the MovieLens database which contains around 1 million ratings of 3952 movies by 6040 users. We will create a very simple recommender system by finding similarities between users' tastes. Our techniques will be based on Euclidean distance (norm), scalar product, and finding correlation coefficients (or angles between the data).

Tasks

1. Load the arrays `movies`, `users_movies`, `users_movies_sort`, `index_small`, `trial_user` from the file `users_movies.mat` using the `load` command. The `matrix_users` movies should be a 6040×3952 matrix containing integer values between 0 and 5 with 1 meaning "strongly dislike" and 5 meaning "strongly like". The 0 in the matrix means that the user did not rate the movie. The array movies contains all the titles of the movies. The matrix `users_movies_sort` contains an extract from the matrix `users_movies.mat` with only rating for the 20 popular movies selected. The indexes of these popular movies are recorded in the array `index_small`. Finally, ratings of these popular movies by yet another user (not the one contained in the database) are given by the vector `trial_user`. It is suggested to view all the variables and their dimensions by using the "Workspace" window of 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);
```

Variables: `movies`, `users_movies`, `users_movies_sort`, `index_small`, `trial_user`, `m, n`

2. Print the titles of the popular movies at which ratings we will be looking by using the following code:

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

Observe that the movie titles are called from the cell array `movies` (notice the curly parentheses) by using an intermediate array `index_small`.

3. Now let us select the users we will compare the `trial_user` to. Here, we want to select the people who rated all of the 20 movies under consideration. This means that there should not be zeros in corresponding rows of the matrix `users_movies_sort`. This can be accomplished by the following code:

```
%% 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;
```

Observe the use of the command `prod` to compute the product of the elements of the row of the array `users_movies_sort`. Now, the array ratings contains all the users which have rated all the popular movies from the array `index_small`. In general, it may happen that this array is empty or too small to create any meaningful comparison - we won't consider these cases in this project.

Variables: `ratings, m1, n1`

Q1: What does the command `ratings=[]` do?

4. Next, we can look at the similarity metric based on the Euclidean distance. The idea here is that we treat the array `trial_user` and the rows of the array ratings as vectors in 20-dimensional real space $\mathbb{R}^{20}$. Assume that all the vectors have the origin as the beginning point. We can find the distance between the end points of the vector `trial_user` and each of the vectors `ratings(j,:)`. In other words, we are looking for the user with the closest ratings to the `trial_user`. This can be accomplished by the following code:

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

   The vector eucl contains all the Euclidean distances between `trial_user` and the rows of the matrix `ratings`.

   Variables: `eucl`

5. Now let us select the user with the smallest Euclidean distance. Instead of using the usual function `min` we will use a slightly more complicated approach. Let us sort the elements of the vector `eucl` by using the function `sort`. The advantage of this is that it allows us to find the second closest user, the third closest user, etc. There may be only a small difference between several closest users and we might want to use their data as well.

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

   Variables: `MinDist`, `DistIndex`, `closest_user_Dist`

6. The similarity metric above is one of the simplest ones which can be used to compare two objects. However, when it comes to user ratings it has certain disadvantages. For instance, what if the users have similar tastes, but one of them consistently judges movies harsher than the other one? The metric above would rate those two users as dissimilar since the Euclidean distance between vectors of their opinions might be pretty large. To rectify this problem, we can look at a different similarity metric known in statistics as the Pearson correlation coefficient which can be defined as

$$r(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{N}(y_i-\overline{y})^2}},$$

   where $\mathbf{x} = (x_1, x_2, \ldots, x_N), \mathbf{y} = (y_1, y_2, \ldots, y_N)$ are two vectors with N components, and $\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$, $\overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ are the average (mean) values of the vectors $\mathbf{x}$ and $\mathbf{y}$. If we look at the formula for $r(\mathbf{x},\mathbf{y})$, we can see that it accounts for the average user opinion $\overline{x}, \overline{y}$, and also for how harshly/exuberantly they tend to judge their movies (magnitude of the vectors $\mathbf{x}-\overline{\mathbf{x}}$ and $\mathbf{y}-\overline{\mathbf{y}}$). Geometrically, the $r(\mathbf{x},\mathbf{y})$ corresponds to the cosine angle between the vectors $\mathbf{x}-\overline{\mathbf{x}}$ and $\mathbf{y}-\overline{\mathbf{y}}$ (the closer this angle is to zero, the more similar are the opinions of two people). To compute the Pearson correlation coefficient, let us centralize the columns of the matrix ratings and the vector trial user first:

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

   Variables: `ratings_cent`, `trial_user`

7. Next, 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`.

   Variables: `pearson`

8. Finally, observe that the value $r(\mathbf{x}, \mathbf{y})$ belongs to the interval $(-1, 1)$. The closer the coefficient is to 1, the more similar the tastes of the users are. Finally, let us sort the vector `pearson` as before using the `sort` function. Save the results of this function as [`MaxPearson`, `PearsonIndex`] and find the maximal correlation coefficient which will be the first element in the matrix `MaxPearson`. Save this element as `closest_user_Pearson`.

   Variables: `MaxPearson`, `PearsonIndex`, `closest_user_Pearson`

9. Compare the elements of the vectors `DistIndex`, `PearsonIndex`.

   Q2: Are the variables `closest_user_Pearson` and `closest_user_Dist` the same?