# 2223-MA438 - Lab 3

Note: the MATLAB Live script that generated this PDF is **`MA438_Lab3_SVD_recommender.mlx`.**

## Table of Contents

```
clear;  % forget any old data
```

## 1. Outline

In this lab, you'll construct a realistic data set to which we can apply a matrix completion algorithm, based on the SVD. The matrix completion problem is a form of "recommender" problem. The data involved is:

- A set of users/people;
- A set of "things" that they can rate, like TV shows, movies, music artists, albums, restaurants, football teams, etc.
- A set of ratings that the users have given then "things".
- But, although every user rates at least one thing, none of them rate everything.
- The "missing" ratings are the indeterminates in the matrix completion problem.

The underlying idea is that a matrix of every possible rating does not have full rank: some columns/rows will be highly correlated. So, as explained in lab 2, we can approximate the missing values using an SVD algorithm.

## 2. An Example

### 2.1 Example Type

I'll first make up a small, simple example. In my case, I'll choose 9 TV shows to be ranked by 6 people. For your example, you should choose something else (ideally) with at least $m = 12$ people, and at least $n = 20$ "things" for them to rate. In your example,

- everyone should rank at least 2 things, most should rank 3 or 4, and some should rank at least 6.
- every item must be ranked by at least 2 people.

We'll discuss the specifications for this later.

### 2.2 Things to Rate

Next I'm going to make a list of $n = 9$ TV shows that users will rate. I'll list them in a cell array. They are numbered 1 to 9, since these these will correspond to a column in a matrix.

```
Shows = {"Octonauts", "Puffin Rock", "Paddington", "Match of The Day",  ...
  "The Sunday Game", "FIFA World Cup", "DWTS", "First Dates", "The Late Late Show"};
for i=1:length(Shows)
   fprintf("%3d : %s\n", i, Shows{i})
end
```

```
  1 : Octonauts
  2 : Puffin Rock
  3 : Paddington
  4 : Match of The Day
  5 : The Sunday Game
  6 : FIFA World Cup
  7 : DWTS
  8 : First Dates
  9 : The Late Late Show
```

In this example, I've deliberately

- Chosen Items 1-3 to be kids TV shows.
- Items 4-6 are about sports.
- Items 7-9 are light entertainment (I assume. I've never actually seen 7 or 8).

### 2.3 Users who give ratings

We will have $m$ =6 users: Ann & Barry (both kids), Cath & Donny (sports fans), and Eddie & Fiona (consumers of light entertainment). Of course the names are not important: we'll just identify them by number, 1 to 6. (Eventually, these numbers will correspond to a row in a matrix).
Again we'll use a cell array to record their names:

```
Users = {"Ann", "Barry", "Cath", "Donny", "Eddie", "Fiona"}
```

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | Ann | Barry | Cath | Donny | Eddie | Fiona |

For example, `Users{3}` will be "`Cath`".

## 3. The Ratings Matrix

The "*Ratings Matrix*", $M$, is a $r \times 3$ matrix, where $r$ is the total number of ratings made by users. If every user rates every show, then $\rho = mn$, but usually $\rho \ll mn$. The entries in column 1 of $M$ are the Users, the entries in Column 2 are the shows, and the entries in Column 3 are the ratings. In this example, we'll give ratings of 1 (a low rating) to 9 (high).

Then enries in row $k$ of the matrix means that User $M(k,1)$ gives Show $M(k,2)$ a rating of $M(k,3)$. For example, if Ann gives a rating of 9 to Octonauts, then there is a row in the matrix which is $[1,1,9]$. If she hates "First Dates", there will be a row with [1,8,1]. It does not matter which row is which: changing the order does not impact on the result.

In my example the ratings matrix is

```
M = [1 1 9;  1 2 9;   1 3 8;  1 8 1;
     2 1 8;  2 2 8;   2 7 1;
     3 4 8;  3 6 8;   3 1 1;
     4 4 8;  4 5 9;   4 2 1;  4 8 1;
     5 7 8;  5 8 8;   5 6 1;
     6 7 9;  6 8 9;   6 9 9;  5 1 1];
```

Take a little care when reading this. Each semicolon separates a row; I've put multiple rows on the same line to make the presentation compact. Also, I have one "row" of input for each user.

For computational reasons, it is better to have an average rating of 0. So we subtract 5 from all the ratings (i.e., column 3). This means that, now, -4 is the lowest rating, 0 is average, and 4 is the highest.

```
M(:,3)=M(:,3)-5
```

```
M = 21x3
     1     1     4
     1     2     4
     1     3     3
     1     8    -4
     2     1     3
     2     2     3
     2     7    -4
     3     4     3
     3     6     3
     3     1    -4
     .     .     .
     .     .     .
```

**4. Recommeder Matrix**

The recommender matrix, $R = (r_{ij})$ is an $m \times n$ matrix with $r_{ij} = q$ if, for some $k$, $M(k, 1) = i$, $M(k, 2) = j$, and $M(k, 3) = q$. We'll build it using a for loop that iterates over the rows of M. We'll use "0" (the average possible score) for the missing valiues.

```
m = length(Users) % number of users
```

```
m = 6
```

```
n = length(Shows) % number of things ranked
```

```
n = 9
```

```
rho = length(M)  % number of preferences expressed
```

```
rho = 21
```

```
R = zeros(m,n);
for k= 1:rho
    i = M(k,1); j = M(k,2); q = M(k,3);
    R(i,j)=q;
end
R
```

```
R = 6x9
     4     4     3     0     0     0     0    -4     0
     3     3     0     0     0     0    -4     0     0
    -4     0     0     3     0     3     0     0     0
     0    -4     0     3     4     0     0    -4     0
    -4     0     0     0     0    -4     3     3     0
     0     0     0     0     0     0     4     4     4
```

We'll take a moment to discuss this, and note it's Block-like structure. Moreover, we want to focus on the zeros. Those indicate where we have no information. They the "indeterminates". The goal of the recommender system is to try to infer likely values for these. The key to doing this is determining what the rank of the matrix should be, and then applying the SVD algorithm from Lab 2.

First, let's examine properties of $R$. For a start, let's check the rank of $R$:

```
rank(R)
```

```
ans = 6
```

You should see that $R$ has full rank.

We should also check the singular values:

```
svd(R)
```

```
ans = 6x1
   10.9031
    8.5582
    5.8004
    5.0051
    4.4914
    3.0019
```

If the prefernces had been “perfect”, we should have found that $R$ has rank 3, and so, as 3 non-zero singular values. If it was a good approximation of the true, $R$, then we would expect to see 3 large singular values, and 3 small, which would mean it would have a good rank-3 appromxation.

However, beause of the missing data, the singular values are not distributed like that. We’ll try to find a rank-3 completion to resolve that.

## 5. The SVD Algorithm

The SVD algorithm runs as follows:

1. Choose $r$ the rank of the completed matrix. (In reality, we have to experiment to find a good value).
2. Identify the interminate entries of $R$.
3. Compute the SVD of $R$. If $\sigma_4$ is "small enough", then $R$ has approximate rank of 3, and we are done. Otherwise...
4. Set $L$ to be the rank- $r$ low-rank approximation of $R$
5. Set $R_1$ to be the matrix with values: $(R_1)_{i,j} = \begin{cases} R_{i,j} & \text{if } R_{i,j} \text{ is not an intererminate} \\ L_{i,j} & \text{otherwise} \end{cases}$
6. Repeate from Step 3 with $R_1$for $R$.

Here is some code that does part of one step of this.

```
r = 3;
Indeterminates = find(R == 0); % Lazy: assumes all 0's correspond to missing values
[U,S,V] = svd(R);
L = U(:,1:r)*S(1:r,1:r)*V(:,1:r)';
R1 = R;
R1(Indeterminates) = L(Indeterminates);
```

The "new" recommender matrix is:

```
disp(R1)
```

```
    4.0000    4.0000    3.0000   -0.8589    0.4407   -0.3703   -0.9380   -4.0000    0.2929
    3.0000    3.0000    0.4624   -1.0298   -1.2011    0.7698   -4.0000   -0.9695   -0.8799
   -4.0000   -1.9047   -1.1894    3.0000    0.1899    3.0000   -0.9233    0.3008   -1.0464
   -0.2660   -4.0000    0.3305    3.0000    4.0000    0.5264    0.6266   -4.0000   -0.2865
   -4.0000   -1.1093   -0.6144   -0.3368   -0.3840   -4.0000    3.0000    3.0000    1.3952
   -0.7159   -0.0806    0.2197   -0.9996   -0.2865   -2.1800    4.0000    4.0000    4.0000
```

(We'll discuss in class how to interpret this).

We also want to see how the distribution of the singular values has changed:

```
svd(R1)
```

```
ans = 6x1
   12.2530
    9.7227
    6.6123
    2.9623
    2.5040
    1.3744
```

We see $R_1$ is somewhat closer to being Rank 3. (As it should be, since the algorithm forces that).

Of course, this is just one iteration of the algorithm. We should repeat it several times, until $\sigma_4$ is less than some value we choose.

## 6. Your Tasks

1. Make up a larger data set. You should have at least 12 users and 20 "things" to rate. When doing so, imagine that the users belong to 2 or 3 different groups that will have similar, but not identical preferences. Similarly the "things" should be grouped. (E.g., for Music, you could have folk, rock, classical, etc). You don't need to have the same number of things in each category, or people in each group, but there needs to be at least 2 in each.
2. Modify all the code and text above to show how the algorithm works for your example.
3. Modify the code that that the SVD algorithm is run several times - until $\sigma_4$ is less than some chosen value.
4. For my example, I set $r = 3$. What $r$ will you choosen and why? Can you justify it based on the singular values?
5. Submit your work on Blackboard (Lab 3) has both a Live Script (.mlx file) and a PDF.
6. Any questions? Ask Niall: Niall.Madden@UniversityOfGalway.ie

**Deadline: TBD!**