EECS 16B HW10

Bryan Ngo

2020-04-12

1 Frobenius Norm

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} |\boldsymbol{A}_{ij}|^2} \tag{1}$$

1.a

Theorem 1. *Given some matrix* $\boldsymbol{A}$*,*

$$\|\boldsymbol{A}\|_F = \sqrt{\operatorname{tr}\{[\boldsymbol{A}]^\top \boldsymbol{A}\}} \tag{2}$$

where $\operatorname{tr}\{\boldsymbol{A}\}$ *is the trace operator*

$$\operatorname{tr}\{\boldsymbol{A}\} = \sum_{i=1}^{N} \boldsymbol{A}_{ii} \tag{3}$$

Proof. Examining $[\boldsymbol{A}]^\top \boldsymbol{A}$,

$$[\boldsymbol{A}]^\top \boldsymbol{A} = \begin{bmatrix} [\boldsymbol{a}_1]^\top \\ [\boldsymbol{a}_2]^\top \\ \vdots \\ [\boldsymbol{a}_N]^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N \end{bmatrix} = \begin{bmatrix} \langle \boldsymbol{a}_1, \boldsymbol{a}_1 \rangle & \langle \boldsymbol{a}_1, \boldsymbol{a}_2 \rangle & \cdots & \langle \boldsymbol{a}_1, \boldsymbol{a}_N \rangle \\ \langle \boldsymbol{a}_2, \boldsymbol{a}_1 \rangle & \langle \boldsymbol{a}_2, \boldsymbol{a}_2 \rangle & \cdots & \langle \boldsymbol{a}_2, \boldsymbol{a}_N \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \boldsymbol{a}_N, \boldsymbol{a}_1 \rangle & \langle \boldsymbol{a}_N, \boldsymbol{a}_2 \rangle & \cdots & \langle \boldsymbol{a}_N, \boldsymbol{a}_N \rangle \end{bmatrix} \tag{4}$$

where $\boldsymbol{a}_i$ is a column vector of $\boldsymbol{A}$. Thus, the trace is

$$\operatorname{tr}\{[\boldsymbol{A}]^\top \boldsymbol{A}\} = \sum_{i=1}^{N} \|\boldsymbol{a}_i\|^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{A}_{ij}^2 \tag{5}$$

$$\sqrt{\operatorname{tr}\{[\boldsymbol{A}]^\top \boldsymbol{A}\}} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{A}_{ij}^2} = \|\boldsymbol{A}\|_F \tag{6}$$

□

1.b

Theorem 2. *Given orthonormal matrices* $\boldsymbol{U}, \boldsymbol{V}$*,*

$$\|\boldsymbol{U}\boldsymbol{A}\|_F = \|\boldsymbol{A}\boldsymbol{V}\|_F = \|\boldsymbol{A}\|_F \tag{7}$$

Proof. Examining $\boldsymbol{UA}, \boldsymbol{AV}$,

$$\boldsymbol{UA} = \boldsymbol{U}\begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N\end{bmatrix} = \begin{bmatrix}\boldsymbol{Ua}_1 & \boldsymbol{Ua}_2 & \cdots & \boldsymbol{Ua}_N\end{bmatrix} \tag{8}$$

$$\boldsymbol{AV} = \begin{bmatrix}\boldsymbol{a}'_1\boldsymbol{V} \\ \boldsymbol{a}'_2\boldsymbol{V} \\ \vdots \\ \boldsymbol{a}'_N\boldsymbol{V}\end{bmatrix} \tag{9}$$

where $\boldsymbol{a}_i$ is a column vector and $\boldsymbol{a}'_i$ is a row vector. Using **Equation 6**,

$$\|\boldsymbol{UA}\|_F = \sqrt{\sum_{i=1}^{N} \|\boldsymbol{Ua}_i\|^2} \tag{10}$$

Since linear transformations by orthonormal matrices do not change the norm of a vector,

$$\sqrt{\sum_{i=1}^{N} \|\boldsymbol{Ua}_i\|^2} = \sqrt{\sum_{i=1}^{N} \|\boldsymbol{a}_i\|^2} = \|\boldsymbol{A}\|_F \tag{11}$$

For the $\boldsymbol{AV}$ case, we need only prove that $\mathrm{tr}\{[\boldsymbol{AV}]^\top\boldsymbol{AV}\} = \mathrm{tr}\{[\boldsymbol{A}]^\top\boldsymbol{A}\}$,

$$[\boldsymbol{AV}]^\top\boldsymbol{AV} = \begin{bmatrix}[\boldsymbol{a}'_1\boldsymbol{V}]^\top & [\boldsymbol{a}'_2\boldsymbol{V}]^\top & \cdots & [\boldsymbol{a}'_N\boldsymbol{V}]^\top\end{bmatrix}\begin{bmatrix}\boldsymbol{a}'_1\boldsymbol{V} \\ \boldsymbol{a}'_2\boldsymbol{V} \\ \vdots \\ \boldsymbol{a}'_N\boldsymbol{V}\end{bmatrix} \tag{12}$$

$$= \sum_{i=1}^{N} [\boldsymbol{a}'_i\boldsymbol{V}]^\top\boldsymbol{a}'_i\boldsymbol{V} = \sum_{i=1}^{N} [\boldsymbol{V}]^\top[\boldsymbol{a}']^\top\boldsymbol{a}'_i\boldsymbol{V} = \sum_{i=1}^{N} \|\boldsymbol{a}_i\|^2 \cancelto{\boldsymbol{I}}{[\boldsymbol{V}]^\top\boldsymbol{V}} = \sum_{i=1}^{N} \|\boldsymbol{a}_i\|^2 \tag{13}$$

Thus, the trace is the same. By **Equation 6**, so are their Frobenius norms. □

1.c

Theorem 3. *Given a matrix* $\boldsymbol{A} \in \mathbb{R}^{m\times n}$,

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{N} \sigma_i^2} \tag{14}$$

where σ_i *are the singular values of* $\boldsymbol{A}$.

Proof. Using the compact SVD, we can represent $\boldsymbol{A} = \boldsymbol{US}[\boldsymbol{V}]^\top$, where $\boldsymbol{U}, \boldsymbol{V}$ are orthonormal matrices and $\boldsymbol{S}$ is a diagonal matrix composed of the non-zero singular values. If we leverage the fact that left *and* right multiplication by orthonormal matrices does not change the Frobenius norm, we can simply reduce it to

$$\|\boldsymbol{A}\|_F = \|\boldsymbol{US}[\boldsymbol{V}]^\top\|_F = \|\boldsymbol{S}\|_F \tag{15}$$

Since $\boldsymbol{S}$ is a diagonal matrix, we use **Equation 6**,

$$\|\boldsymbol{S}\|_F = \sqrt{\sum_{i=1}^{N} \|\boldsymbol{s}_i\|^2} = \sqrt{\sum_{i=1}^{N} \sigma_i^2} \tag{16}$$

where the last equality holds because $\|\boldsymbol{s}_i\| = \sigma_i$ due to the diagonality of $\boldsymbol{S}$. □

2 Minimum Energy Control

$$\boldsymbol{x}[t+1] = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}\boldsymbol{x}[t] + \begin{bmatrix}0\\1\end{bmatrix}u[t] \tag{17}$$

2.a

We construct the controllability matrix for $n = 5$,

$$\boldsymbol{C} = \begin{bmatrix}\boldsymbol{b} & \boldsymbol{A}\boldsymbol{b} & \cdots & \boldsymbol{A}^{n-1}\boldsymbol{b}\end{bmatrix} \tag{18}$$

$$= \begin{bmatrix}0 & 1 & 2 & 3 & 4\\1 & 1 & 1 & 1 & 1\end{bmatrix} \tag{19}$$

Thus, to reach the target state, we have

$$\begin{bmatrix}0 & 1 & 2 & 3 & 4\\1 & 1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}u[0]\\u[1]\\u[2]\\u[3]\\u[4]\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \tag{20}$$

We can use the minimum norm equation $\hat{\boldsymbol{u}} = [\boldsymbol{C}]^\top(\boldsymbol{C}\,[\boldsymbol{C}]^\top)^{-1}\boldsymbol{x}_{tgt}$ to obtain

$$\boldsymbol{C}\,[\boldsymbol{C}]^\top = \begin{bmatrix}0 & 1 & 2 & 3 & 4\\1 & 1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}0 & 1\\1 & 1\\2 & 1\\3 & 1\\4 & 1\end{bmatrix} = \begin{bmatrix}30 & 10\\10 & 5\end{bmatrix} \overset{-1}{\Longrightarrow} \frac{1}{10}\begin{bmatrix}1 & -2\\-2 & 6\end{bmatrix} \tag{21}$$

$$[\boldsymbol{C}]^\top(\boldsymbol{C}\,[\boldsymbol{C}]^\top)^{-1} = \frac{1}{10}\begin{bmatrix}0 & 1\\1 & 1\\2 & 1\\3 & 1\\4 & 1\end{bmatrix}\begin{bmatrix}1 & -2\\-2 & 6\end{bmatrix} = \frac{1}{10}\begin{bmatrix}-2 & 6\\-1 & 4\\0 & 2\\1 & 0\\2 & -2\end{bmatrix} \tag{22}$$

$$\hat{\boldsymbol{u}} = [\boldsymbol{C}]^\top(\boldsymbol{C}\,[\boldsymbol{C}]^\top)^{-1}\boldsymbol{x}_{tgt} = \frac{1}{10}\begin{bmatrix}-2 & 6\\-1 & 4\\0 & 2\\1 & 0\\2 & -2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{10}\begin{bmatrix}-2\\-1\\0\\1\\2\end{bmatrix} \tag{23}$$

$$E = u[0]^2 + u[1]^2 + u[2]^2 + u[3]^2 + u[4]^2 = \frac{1}{10} \tag{24}$$

2.b

Theorem 4. *Given the system in* ***Equation 17****, initial condition* $\boldsymbol{x}[1] = \begin{bmatrix}1\\0\end{bmatrix}$*, and outputs* $u[t \geqslant 2] = 0$*,* $\boldsymbol{x}[t \geqslant 2] = \boldsymbol{x}[1]$.

Proof. By induction,

$$\boldsymbol{x}[2] = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}\boldsymbol{x}[1] = \begin{bmatrix}1\\0\end{bmatrix} \tag{25}$$

$$\boldsymbol{x}[t+1] = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}\boldsymbol{x}[t] = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \tag{26}$$

□

2.c

By inspection, we can pick $u[0] = \frac{1}{2}, u[1] = -\frac{1}{2}$. Given the rest of the inputs $u[t \in [2,5]] = 0$, $E = \frac{1}{2} > \frac{1}{10}$.

3 PCA

$$\boldsymbol{A} = \begin{bmatrix} 64 & 39 \\ 66 & 45 \\ 70 & 41 \\ 64 & 39 \end{bmatrix} \tag{27}$$

3.a

Our means are $\overline{\boldsymbol{A}_1} = 66\,°\text{F}$ and $\overline{\boldsymbol{A}_2} = 41\%$. Demeaning and calculating $\boldsymbol{C}$,

$$\widetilde{\boldsymbol{A}} = \begin{bmatrix} -2 & -2 \\ 0 & 4 \\ 4 & 0 \\ -2 & -2 \end{bmatrix} \tag{28}$$

$$\boldsymbol{C} = \frac{1}{m-1}\left[\widetilde{\boldsymbol{A}}\right]^\top \widetilde{\boldsymbol{A}} = \frac{1}{3}\begin{bmatrix} -2 & 0 & 4 & -2 \\ -2 & 4 & 0 & -2 \end{bmatrix}\begin{bmatrix} -2 & -2 \\ 0 & 4 \\ 4 & 0 \\ -2 & -2 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 24 & 8 \\ 8 & 24 \end{bmatrix} = \begin{bmatrix} 8 & \frac{8}{3} \\ \frac{8}{3} & 8 \end{bmatrix} \tag{29}$$

3.b

$$\det(\boldsymbol{C} - \lambda \boldsymbol{I}) = (8-\lambda)^2 - \frac{64}{9} = 0 \tag{30}$$

$$\Rightarrow \lambda_1 = \frac{32}{3} > \lambda_2 = \frac{16}{3} \tag{31}$$

3.c

$$\boldsymbol{C} - \lambda_1 \boldsymbol{I} = \begin{bmatrix} -\frac{8}{3} & \frac{8}{3} \\ \frac{8}{3} & -\frac{8}{3} \end{bmatrix} \Rightarrow \boldsymbol{p}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{32}$$

$$\boldsymbol{C} - \lambda_2 \boldsymbol{I} = \begin{bmatrix} \frac{8}{3} & \frac{8}{3} \\ \frac{8}{3} & \frac{8}{3} \end{bmatrix} \Rightarrow \boldsymbol{p}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{33}$$

3.d

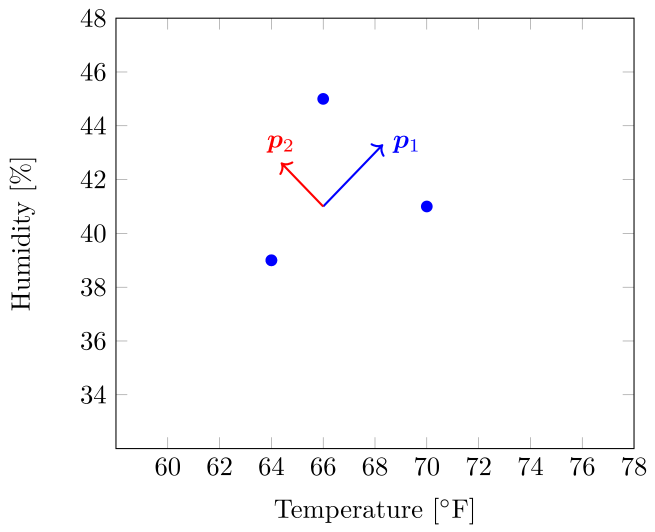
The principal components are nothing more that $\boldsymbol{p}_1, \boldsymbol{p}_2$, and

$$\boldsymbol{P} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{34}$$

3.e

The weights are nothing more than the square roots of the eigenvalues λ_1, λ_2, so $\sigma_1 = \sqrt{\frac{32}{3}}, \sigma_2 = \sqrt{\frac{16}{3}}$.

3.f



4 Netflix Recommendation Using SVD

4.a

See `recommender_system.ipynb`.

Recommender System

Question A

Loading the students' ratings from the csv file TA_data.csv using pandas.read_csv.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
student_data = pd.read_csv("data_TAs.csv")
student_data
```

Out[1]:

	Unnamed: 0	Harry Potter	Groundhog Day	Life of Brian	Indiana Jones	Lawrence of Arabia	Jurassic Park	Rushmore	Grease	The Godfather	...	Tita
0	Maruf	3	2	3	3	3	3	3	3	4	...	
1	Mauricio	2	3	4	3	2	3	4	3	3	...	
2	Ricky	2	5	4	2	3	2	4	4	1	...	
3	Simon	3	4	3	3	3	3	3	3	3	...	
4	Ting	4	1	1	5	4	4	2	2	4	...	
5	Mia	4	2	2	4	3	3	2	3	4	...	
6	Marie	2	5	5	1	2	2	4	4	2	...	
7	Gaoyue	3	4	3	3	3	3	3	3	2	...	
8	Nick	4	1	2	4	3	4	2	2	4	...	
9	Ramsey	3	4	3	3	3	3	3	3	2	...	

10 rows × 26 columns

Question A

Below, the ratings matrix is being stored in the variable `ratings` .

1. Calculate the mean of the rows u and subtract it from each row of the matrix.
2. Compute the SVD of the resulting matrix using `np.linalg.svd` and plot the singular values.

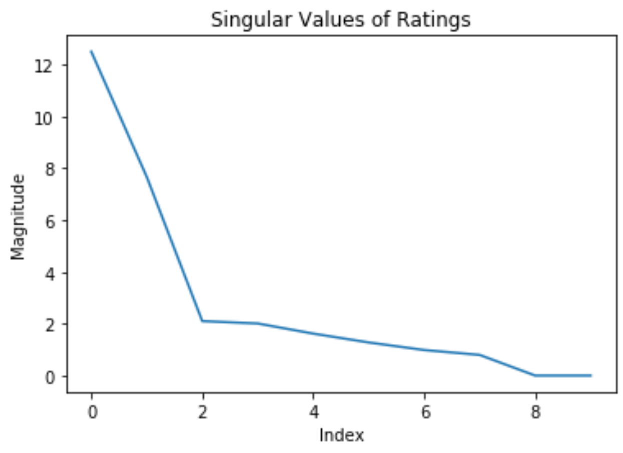
Please look at the documentation of np.linalg.svd to understand its return values.

```
In [2]: ratings = np.array(student_data.values)
        print(ratings)
```

```
[['Maruf' 3 2 3 3 3 3 3 3 4 3 4 4 3 4 3 2 2 3 4 2 2 3 3 4 4]
 ['Mauricio' 2 3 4 3 2 3 4 3 3 4 5 3 4 3 3 1 1 3 3 2 1 3 3 4 3]
 ['Ricky' 2 5 4 2 3 2 4 4 1 4 3 1 4 2 2 3 3 2 2 4 3 2 1 3 2]
 ['Simon' 3 4 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 4 4 3 3 3 3]
 ['Ting' 4 1 1 5 4 4 2 2 4 1 1 4 1 4 4 4 4 4 4 2 4 4 5 2 4]
 ['Mia' 4 2 2 4 3 3 2 3 4 2 2 4 2 4 4 3 3 4 4 2 3 4 4 3 4]
 ['Marie' 2 5 5 1 2 2 4 4 2 5 5 2 5 2 2 2 2 2 2 4 2 2 1 4 2]
 ['Gaoyue' 3 4 3 3 3 3 3 3 2 3 3 2 3 2 3 4 4 3 2 4 4 3 2 2 2]
 ['Nick' 4 1 2 4 3 4 2 2 4 2 3 5 3 4 4 3 3 4 4 2 2 4 4 4 4]
 ['Ramsey' 3 4 3 3 3 3 3 3 2 3 2 2 3 2 3 4 4 3 2 4 4 3 3 2 2]]
```

```
In [3]: # YOUR WORK FOR QUESTION A HERE
        r8s_no_name = ratings[:, 1:].T
        means = [np.mean(i) for i in r8s_no_name]
        r8s_no_name = np.array([r8s_no_name[i] - means[i] for i in range(r8s_no_name.sh
        ape[0])], dtype='float').T
        ratings_U, ratings_S, ratings_V = np.linalg.svd(r8s_no_name)

        plt.plot(ratings_S)
        plt.title('Singular Values of Ratings')
        plt.xlabel('Index')
        plt.ylabel('Magnitude')
        plt.show()
```



Question B

- It seems like 3 singular values contribute most of the information in the data set.
- Truncate the SVD to build a rank 3 model of the data. That is, find `(U_truncated, S_truncated, VT_truncated)` of dimensions `((10, 3), (3, 3), (3, 25))` respectively such that `U_truncated.dot(S_truncated.dot(VT_truncated))` is a good approximation to the data matrix. ***Hint.*** You might need `np.linalg.diag` on the singular values given the output of `np.linalg.svd`. **Look at the documentation.**
- What is the percentage error between the true ratings and the low rank model? the percentage error is defined as: `np.linalg.norm(ratings - ratings_approx)/np.linalg.norm(ratings) * 100`

```
In [4]: r = 3# rank of truncated SVD


        U_truncated = ratings_U[:, :r]
        S_truncated = np.diag(ratings_S[:r])
        VT_truncated = ratings_V[:r]
        ratings_approx = U_truncated.dot(S_truncated.dot(VT_truncated))


        print(np.linalg.norm(r8s_no_name - ratings_approx) / np.linalg.norm(r8s_no_nam
        e) * 100)
```

```
20.775200005168827
```

We will now load the professor's ratings.

```
In [5]: import json
        with open('data_arcak.json') as data_file:
            ratings_arcak = json.load(data_file)
        with open('data_sanders.json') as data_file:
            ratings_sanders = json.load(data_file)

        print("ratings_arcak =", ratings_arcak)
        print("ratings_sanders =", ratings_sanders)
```

```
ratings_arcak = {'The Godfather': 3.498, 'Jaws': 3.0, 'Grease': 2.253, 'Toy Sto
ry': 3.249, '2001: A Space Odyssey': 3.0}
ratings_sanders = {'Rushmore': 3.747, 'Spinal Tap': 3.996, 'Groundhog Day': 3.
0, 'Jaws': 2.751}
```

Since the professors have each only seen a small number of movies, we will need to estimate their preferences based on only those movies. Recall the we have abstracted our movie ratings in vector form. For example, Maruf's ratings in vectors form is,

$$\begin{bmatrix} \text{Harry Potter} \\ \text{Groundhog day} \\ \vdots \\ \text{Lord of the rings} \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ \vdots \\ 4 \end{bmatrix}$$

Question C

1. Construct the vector representations of the Professors' ratings by setting the unknown movie ratings to the corresponding entries from the mean of the rows u.
2. Subtract the mean u from the resulting vector.

Hint. Use `list(student_data.columns).index()` to determine the appropriate indexes.

In [6]:
```
# YOUR WORK FOR QUESTION C HERE
prof_arcak = np.array([
    ratings_arcak[i] - means[list(student_data.columns)[1:].index(i)]
    if i in ratings_arcak else 0
    for i in list(student_data.columns)[1:]
])
prof_sanders = np.array([
    ratings_sanders[i] - means[list(student_data.columns)[1:].index(i)]
    if i in ratings_sanders else 0
    for i in list(student_data.columns)[1:]
])
```

Some notes.

- V forms a basis for the rows of our ratings matrix and represents the "principal directions" of user reviews.
- It is a model assumption that, if you like a certain group of movies, those movies will share similar characteristics. Sci-fi and action films, for example, have more in common than rom-coms usually, and we try and exploit that.
- If we have two right singular vectors v_1 and v_2, v_1 could represent that principal "action" direction and v_2 could represent the principal "rom-com" direction. For example, dominantly action film will be something like $10v_1 + 02v_2$ and a dominantly rom-com film will have something like $0.1v_1 + 5v_2$. A movie that is a rom-com action film would be some weighted somewhat evenly, like $2v_1 + 2v_2$.
- Since we have a way of defining the characteristics of a film in terms of the basis vectors in V, we can try to guess the remaining ratings for the Professors by projecting their ratings onto V. For example, if the projection onto v_1 is higher than v_2, it suggests that the person would like action films more that rom-coms. We will use this idea to guess the ratings.

Question D

We have 3 principal components (v_1, v_2, v_3) with `VT_truncated` being,

$$\begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix}^T$$

Construct `proj_arcak` and `proj_sanders` by calculating the representations of `prof_arcak` and `prof_sanders` in terms of (v_1, v_2, v_3). That is to say, if x is `prof_arcak`,

$$\text{prof\_arcak} = \langle x, v_1\rangle v_1 + \langle x, v_2\rangle v_2 + \langle x, v_3\rangle v_3$$

Hint. Remember orthonormal projections.

In [7]:
```
# YOUR WORK FOR QUESTION D HERE
proj_arcak = VT_truncated.T.dot(VT_truncated).dot(prof_arcak)
proj_sanders = VT_truncated.T.dot(VT_truncated).dot(prof_sanders)
```

Question E

We're almost there! We now simply need to account for the mean u. Define `pred_arcak` and `pred_sanders` as,

$$\text{pred\_arcak} = \text{proj\_arcak} + u \text{ and } \text{pred\_sanders} = \text{proj\_sanders} + u$$

This effectively adds the bias back.

In [8]:
```
# YOUR WORK FOR PART E HERE
pred_arcak = proj_arcak + means
pred_sanders = proj_sanders + means
```

Question F

Find the movie corresponding the largest entry of `pred_arcak` and `pred_sanders` . This is your first guess at a movie recommendation!

In [9]:
```
# YOUR WORK FOR QUESTION F HERE
print(list(student_data.columns)[1:][np.argmax(pred_arcak)])
print(list(student_data.columns)[1:][np.argmax(pred_sanders)])
```

```
Spiderman
Groundhog Day
```

Question G

Professor Sanders has already seen his recommendation. What's the next best movie you can recommend?

In [10]:
```
# YOUR WORK FOR QUESTION G HERE
pred_sanders[np.argmax(pred_sanders)] = 0
print(list(student_data.columns)[1:][np.argmax(pred_sanders)])
```

```
Freddy Got Fingered
```