CSCI4022_F21_HW8

November 18, 2021

1 CSCI4022 Homework 8; Recommendations

1.1 Due Wednesday, November 17 at 11:59 pm to Canvas and Gradescope

Submit this file as a .ipynb with *all cells compiled and run* to the associated dropbox.

Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own**.

NOTES:

- Any relevant data sets should be available on Canvas. To make life easier on the graders if they need to run your code, do not change the relative path names here. Instead, move the files around on your computer.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Here is a reference guide linked on Canvas on writing math in Markdown. **All** of your written commentary, justifications and mathematical work should be in Markdown. I also recommend the wikibook for LaTex.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do **Kernel** → **Restart & Run All** as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- 45 points of this assignment are in problems. The remaining 5 are for neatness, style, and overall exposition of both code and text.
- This probably goes without saying, but… For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.
- There is *not a prescribed API* for these problems. You may answer coding questions with whatever syntax or object typing you deem fit. Your evaluation will primarily live in the clarity of how well you present your final results, so don't skip over any interpretations! Your code should still be commented and readable to ensure you followed the given course algorithm.

Shortcuts: Problem 1 | Problem 2 | Problem 3 | Extra Credit |

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
# import statsmodels.api as sm
```

Back to top # Problem 1 (7 pts; Theory: PCA) Prove that if M is any matrix, then M^TM and MM^T are symmetric.

Solution Markdown

Rule 1: $(AB)^T = B^TA^T$

$(M^TM)^T = M^T(M^T)^T = M^TM$

$(MM^T)^T = (M^T)^TM^T = MM^T$

In both cases, M^TM and MM^T are equal to their respective transposes. That is, $M^TM = (M^TM)^T$ and $MM^T = (MM^T)^T$ due to the transpose rule listed above.

Back to top # Problem 2 (13 pts; Theory: Recommendations and Scaling)

To date, we've used **centered cosine distance** as a mechanism to center data. This is not the only such option. Consider the following:

Three computers, A, B, and C, have the numerical features listed below:

Feature	A
Processor Speed	3.06
Disk Size	500
Main-Memory Size	6

We may imagine these values as defining a vector for each computer; for instance, A's vector is [3.06, 500, 6]. We can compute the cosine distance between any two of the vectors, but if we do not scale the components (via a scalar multiplication), then the disk size will dominate the dot product and make differences in the other components essentially invisible. Let us use 1 as the scale factor for processor speed, α for the disk size, and β for the main memory size.

(A) In terms of α and β, compute the cosines of the angles between the vectors for each pair of the three computers.

(B) What are the angles between the vectors if $\alpha = \beta = 1$?

(C) What are the angles between the vectors if $\alpha = 0.01$ and $\beta = 0.5$?

(D) One fair way of selecting scale factors is to make each inversely proportional to the average value in its component. What would be the values of α and β, and what would be the angles between the vectors?

[2]:
```
def cosine_similarity(A, B):
    return np.dot(A, B) / (np.linalg.norm(A) * np.linalg.norm(B))

raw_data = [[3.06, 500, 6],
            [2.68, 320, 4],
            [2.92, 640, 6]]

'''
A
'''

data = [[data[0], data[1], data[2]] for data in raw_data]

AB = cosine_similarity(data[0], data[1])
AC = cosine_similarity(data[0], data[2])
BC = cosine_similarity(data[1], data[2])

print(f"The angle between AB is: {np.round(AB, 5)} radians and {np.round(np.
 ↪degrees(AB), 5)} degrees")
print(f"The angle between AC is: {np.round(AC, 5)} radians and {np.round(np.
 ↪degrees(AC), 5)} degrees")
print(f"The angle between BC is: {np.round(BC, 5)} radians and {np.round(np.
 ↪degrees(BC), 5)} degrees")

'''
B
'''
alpha = 1
beta  = 1

data = [[data[0], alpha * data[1], beta * data[2]] for data in raw_data]
print(data)

AB = cosine_similarity(data[0], data[1])
AC = cosine_similarity(data[0], data[2])
BC = cosine_similarity(data[1], data[2])

print(f"The angle between AB is: {np.round(AB, 5)} radians and {np.round(np.
 ↪degrees(AB), 5)} degrees")
print(f"The angle between AC is: {np.round(AC, 5)} radians and {np.round(np.
 ↪degrees(AC), 5)} degrees")
print(f"The angle between BC is: {np.round(BC, 5)} radians and {np.round(np.
 ↪degrees(BC), 5)} degrees")

'''
C
'''
```

```
alpha = 0.01
beta  = 0.5

data = [[data[0], alpha * data[1], beta * data[2]] for data in raw_data]
print(data)

AB = cosine_similarity(data[0], data[1])
AC = cosine_similarity(data[0], data[2])
BC = cosine_similarity(data[1], data[2])

print(f"The angle between AB is: {np.round(AB, 5)} radians and {np.round(np.
 ↪degrees(AB), 5)} degrees")
print(f"The angle between AC is: {np.round(AC, 5)} radians and {np.round(np.
 ↪degrees(AC), 5)} degrees")
print(f"The angle between BC is: {np.round(BC, 5)} radians and {np.round(np.
 ↪degrees(BC), 5)} degrees")

'''
D
'''
```

```
The angle between AB is: 1.0 radians and 57.29563 degrees
The angle between AC is: 1.0 radians and 57.29551 degrees
The angle between BC is: 0.99999 radians and 57.29508 degrees
[[3.06, 500, 6], [2.68, 320, 4], [2.92, 640, 6]]
The angle between AB is: 1.0 radians and 57.29563 degrees
The angle between AC is: 1.0 radians and 57.29551 degrees
The angle between BC is: 0.99999 radians and 57.29508 degrees
[[3.06, 5.0, 3.0], [2.68, 3.2, 2.0], [2.92, 6.4, 3.0]]
The angle between AB is: 0.99088 radians and 56.77333 degrees
The angle between AC is: 0.99155 radians and 56.8119 degrees
The angle between BC is: 0.96918 radians and 55.5298 degrees
```

[2]: '\nD\n'

Not sure if I interpretted this part correctly. In fact I know I didn't. Don't really remember covering this, and I have no clue why my answers are this way, as computers A and C seem to be the closest, yet I'm getting B and C.

Really no idea where to get started on this for **A.**, **B.**, **C.**, **D.**. Just need thanksgiving break to be here already.

Back to top # Problem 3 (25 pts; Practice: Recommendations)

This problem is about recommender systems. There is a joke recommender dataset from this site:

http://eigentaste.berkeley.edu/dataset/

`jokeratings.csv` contains data from 24,983 users who have rated 36 or more jokes from this set. The text of the jokes themselves is contained in `jokes.csv`. Each are saved as UTF-8 files.

WARNING NOTE: A number of the jokes are distasteful at best and may be offensive. You may skip any prompt that required you actually read/print actual jokes if you wish, and fabricate ranking values when applicable in part **D**. In practice, finding objects that are polarizing (such as bad/offensive jokes) is part of the point of a recommendation system, so instead I recommend you rank such jokes very low and see if a similar user patten is be manifested in the data!

[3]:
```
ratings=pd.read_csv('jokeratings.csv', encoding='UTF-8', header=None)
ratings.head(3)
```

[3]:

	0	1	2	3	4	5	6	7	8	9	…	91	\
0	74	-7.82	8.79	-9.66	-8.16	-7.52	-8.50	-9.85	4.17	-8.98	…	2.82	
1	100	4.08	-0.29	6.36	4.37	-2.38	-9.66	-0.73	-5.34	8.88	…	2.82	
2	49	99.00	99.00	99.00	99.00	9.03	9.27	9.03	9.27	99.00	…	99.00	

	92	93	94	95	96	97	98	99	100
0	99.00	99.00	99.00	99.00	99.00	-5.63	99.00	99.00	99.00
1	-4.95	-0.29	7.86	-0.19	-2.14	3.06	0.34	-4.32	1.07
2	99.00	99.00	9.08	99.00	99.00	99.00	99.00	99.00	99.00

[3 rows x 101 columns]

[4]:
```
jokes=pd.read_csv('jokes.csv', encoding='UTF-8', header=None)
len(jokes)
```

[4]: 100

Part A:

Read the dataset and ensure it is clean. That is, that it contains no odd or fill entries such as NAN or nonsensical numerical values. Perform any adjustments to set up the data frame for a k-nearest neighbors recommendation system.

(Note that the first column of the CSV is the count of jokes rated by that user, the other 100 columns are their actual ratings.)

[5]:
```
#YOUR CODE HERE
lb = -10
ub = 10
fill = 99

initial_size = ratings.shape[0]
print(f"Number of users with valid ratings: {initial_size}\n")
```

```
# because we simply want to get rid of NAN rows...
print("Dropping rows with NANs...")
print(f"We dropped {initial_size - ratings.shape[0]} rows\n")
ratings.dropna()

# we also drop rows with values outside the range -10...10 and that are not␣
 ↪equal to our null value of 99
print(f"Dropping rows ratings less than {lb}, greater than {ub}, and unequal to␣
 ↪the null value ({fill} in our case)...")
print(f"We dropped {initial_size - ratings.shape[0]} rows\n")
ratings = ratings[((lb <= ratings.iloc[:,1:]) & (ratings.iloc[:,1:] <= ub) |␣
 ↪(ratings.iloc[:,1:] == fill)).all(1)]

final_size = ratings.shape[0]
print(f"Number of users with valid ratings: {final_size}")
print(f"We dropped {initial_size - final_size} rows in total!")
```

```
Number of users with valid ratings: 24983

Dropping rows with NANs…
We dropped 0 rows

Dropping rows ratings less than -10, greater than 10, and unequal to the null
value (99 in our case)…
We dropped 0 rows

Number of users with valid ratings: 24983
We dropped 0 rows in total!
```

Part B:

For a naive recommender system, we have enough data to implement $k-$ nearest neighbors for a collaborative filtering.

Assume there is an user who has rated only one joke and that happens to be joke 8. The user has given the joke a rating of 5.

- Print joke 8.

Now we have to recommend a set of jokes to the user.

- You will need to find a list of `k` people who have given a similar rating on joke 8. The value of `k` is your choice.

Print these users (by their index/identifiers).

- You will also need to find the top `m` jokes rated by these `k` users. Again `m` is your choice.

Since we only have one user and one rating, we may have had numerous "perfect match" other users. Even in this case, you should include terms to *weight* the averages of the nearest-neighbors by their similarities to the new user.

Print these jokes (both by their identifiers and the joke itself)

[6]:
```
# nansum returns 0 if all values in array are nan, and I wanted it to return nan
def true_nansum(arr):
    a, mask = np.lib.nanfunctions._replace_nan(arr,0)
    if np.all(mask):
        return np.nan
    return np.nansum(arr)

def create_new_user(jokes, scores, fill, n_col):
    l = [len(jokes)]
    for col in range(n_col):
        flag = True
        for i, joke in enumerate(jokes):
            if joke == col:
                l.append(scores[i])
                flag = False
        if flag:
            l.append(fill)
    return l

def neighbors(df, user, k, fill_value=np.nan, cols=[]):
    nan_df       = df.replace(fill_value, np.nan)
    nan_user     = pd.Series(user).replace(fill_value, np.nan)
    nan_diff     = nan_df - nan_user
    cols_to_drop = list(set(nan_diff.columns) - set(cols))
    nan_diff     = nan_diff.drop(cols_to_drop, axis=1)
    abs_sums     = []
    for index, row in nan_diff.iterrows():
        abs_sums.append(np.abs(true_nansum(row)))
    return list(pd.Series(abs_sums).sort_values()[:k].index)

def get_jokes(df, user, neighbors, m, fill_value=np.nan, cols=[]):
    n_neighbors  = len(neighbors)
    cols_to_drop = list(set(df.columns) - set(cols))
    z_df         = df.replace(fill_value, 0)
    z_user       = pd.Series(user).replace(fill_value, 0)
    sim_XY       = []
    cols         = np.zeros(len(df.columns))
    for neighbor in neighbors:
        sim = stats.pearsonr(z_df.iloc[neighbor,:], z_user)[0]
        sim_XY.append(sim)
        cols += (sim * z_df.iloc[neighbor,:])
    cols *= (1 / np.sum(sim_XY))
    jokes = pd.Series(cols[1:]).sort_values(ascending=False)[:m].index
    return jokes
```

[7]:
```
'''
Joke 8
'''
print(f"Joke 8: {jokes.iloc[8].values[0]}\n")

'''
Choosing m
'''
# percentage of users to use
percent = 0.1
n_jokes = len(jokes)
# number of neighbors for our new user
m       = int(n_jokes * percent)
print(f"m = {m}\n")

'''
Choosing k
'''
# percentage of users to use
percent = 0.01
# number of neighbors for our new user
k       = int(ratings.shape[0] * percent)
print(f"k = {k}\n")

'''
Creating a new user
'''
n_columns    = 100
fill_value   = 99.0
joke_indices = [8]
new_scores   = [5]
new_user     = create_new_user(joke_indices, new_scores, fill_value, n_columns)
print(f"Our new user looks like:\n{new_user}\n")

'''
Getting the k-nearest neighbors
'''
# The actual columns the jokes are stored in
columns     = [i for i in range(1, n_columns + 1)]
k_neighbors_complex = neighbors(ratings, new_user, k, fill_value, columns)
print(f"{k} nearest neighbors are:\n{k_neighbors_complex}")

'''
Getting m jokes
'''
recommended_jokes = get_jokes(ratings, new_user, k_neighbors_complex, m,␣
↪fill_value, columns)
```

```python
for joke in recommended_jokes:
    print(f"Joke {joke}: {jokes.iloc[joke].values[0]}\n")
```

```
Joke 8: A country guy goes into a city bar that has a dress code, and the maitre
d'  demands he wear a tie. Discouraged, the guy goes to his car to sulk when
inspiration strikes: He's got jumper cables in the trunk! So he wraps them
around his neck, sort of like a string tie (a bulky string tie to be sure) and
returns to the bar. The maitre d' is reluctant, but says to the guy, "Okay,
you're a pretty resourceful fellow, you can come in… but just don't start
anything"!

m = 10

k = 249

Our new user looks like:
[1, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 5, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0]

249 nearest neighbors are:
[17947, 18086, 13115, 5403, 6988, 20470, 13855, 19944, 24121, 4199, 15063,
22917, 23619, 5000, 8332, 13619, 1332, 23661, 17038, 11727, 10780, 3771, 13438,
22389, 6961, 22101, 23224, 5394, 18509, 16394, 2121, 12553, 18933, 24140, 13109,
21833, 4020, 15528, 17167, 9153, 21349, 19361, 1605, 14890, 12652, 9949, 17116,
4981, 4946, 14578, 6845, 5990, 13642, 12425, 14898, 20844, 24387, 19142, 4262,
7036, 8223, 757, 3886, 15808, 17743, 4327, 15798, 4821, 14303, 819, 13174, 3445,
21071, 83, 19335, 21408, 3263, 16433, 22500, 24896, 6709, 2141, 9737, 6213, 399,
13884, 15702, 10842, 2063, 11302, 14679, 9853, 674, 13408, 24643, 18790, 771,
20717, 20518, 2578, 6448, 19173, 14436, 10967, 13977, 20486, 2383, 8940, 21081,
23505, 13593, 6831, 13561, 13314, 15604, 13477, 2181, 14606, 3367, 22736, 6029,
19388, 2244, 23755, 1558, 17598, 17016, 18390, 7356, 17940, 15257, 7358, 8726,
19545, 11971, 17390, 18263, 9493, 20055, 1258, 22502, 11901, 22453, 12400, 1789,
1797, 23132, 3915, 7243, 16787, 16760, 17707, 7129, 21831, 7656, 1124, 4326,
15469, 8113, 7781, 7180, 20619, 2788, 11943, 15135, 19998, 6681, 10353, 9990,
6382, 20489, 23577, 12100, 17415, 50, 7729, 8556, 5724, 36, 6728, 4413, 19853,
19828, 14132, 20699, 24808, 13267, 6762, 18738, 21903, 11223, 12884, 9864,
13387, 21992, 309, 2163, 24152, 6079, 17071, 12737, 14865, 9258, 10321, 14774,
7009, 20375, 5469, 5470, 10722, 7495, 23929, 11874, 22323, 12571, 23005, 2930,
14519, 5524, 19754, 23775, 20214, 5331, 5712, 8908, 19835, 3447, 5886, 15038,
8843, 20325, 14969, 5850, 704, 15244, 22734, 10535, 977, 17141, 4982, 24867,
9136, 11207, 2144, 9856, 13415, 232, 18710, 11990]
```

Joke 9: Two cannibals are eating a clown, one turns to other and says: "Does this taste funny to you?

Joke 50: Did you hear that Clinton has announced there is a new national bird? The spread eagle.

Joke 36: A Jewish young man was seeing a psychiatrist for an eating and sleeping disorder. "I am so obsessed with my mother… As soon as I go to sleep, I start dreaming, and everyone in my dream turns into my mother. I wake up in such a state, all I can do is go downstairs and eat a piece of toast." The psychiatrist replies: "What, just one piece of toast, for a big boy like you?"

Joke 27: A mechanical, electrical and a software engineer from Microsoft were driving through the desert when the car broke down. The mechanical engineer said "It seems to be a problem with the fuel injection system, why don't we pop the hood and I'll take a look at it." To which the electrical engineer replied, "No I think it's just a loose ground wire, I'll get out and take a look." Then, the Microsoft engineer jumps in. "No, no, no. If we just close up all the windows, get out, wait a few minutes, get back in, and then reopen the windows everything will work fine."

Joke 62: An engineer, a physicist and a mathematician are sleeping in a room. There is a fire in the room. The engineer wakes up, sees the fire, picks up the bucket of water and douses the fire and goes back to sleep. Again there is fire in the room. This time, the physicist wakes up, notices the bucket, fills it with water, calculates the optimal trajectory and douses the fire in minimum amount of water and goes back to sleep. Again there is fire. This time the mathematician wakes up. He looks at the fire, looks at the bucket and the water and exclaims, "A solution exists" and goes back to sleep.

Joke 66: Once upon a time, two brooms fell in love and decided to get married. Before the ceremony, the bride broom informed the groom broom that she was expecting a little whiskbroom. The groom broom was aghast! "How is this possible?" he asked. "We've never swept together!

Joke 69: Employer to applicant: "In this job we need someone who is responsible." Applicant: "I'm the one you want. On my last job, every time anything went wrong, they said I was responsible."

Joke 35: A guy walks into a bar, orders a beer and says to the bartender, "Hey, I got this great Polish Joke…" The barkeep glares at him and says in a warning tone of voice: "Before you go telling that joke you better know that I'm Polish, both bouncers are Polish and so are most of my customers" "Okay" says the customer,"I'll tell it very slowly."

Joke 68: This guys wife asks, "Honey if I died would you remarry?" and he replies, "Well, after a considerable period of grieving, we all need companionship, I guess I would." She then asks, "If I died and you remarried,

would she live in this house?" and he replies, "We've spent a lot of time and money getting this house just the way we want it. I'm not going to get rid of my house, I guess she would." "If I died and you remarried, and she lived in this house, would she sleep in our bed?" and he says, "That bed is brand new, we just paid two thousand dollars for it, it's going to last a long time, I guess she would." So she asks, "If I died and you remarried, and she lived in this house, and slept in our bed, would she use my golf clubs?" "Oh no, she's left handed."

Joke 10: Q. What do a hurricane, a tornado, and a redneck divorce all have in common? A. Someone's going to lose their trailer…

[8]:
```
'''
Joke 8
'''
print(f"Joke 8: {jokes.iloc[8].values[0]}\n")

'''
Choosing m
'''
# percentage of users to use
percent = 0.1
n_jokes = len(jokes)
# number of neighbors for our new user
m       = int(n_jokes * percent)
print(f"m = {m}\n")

'''
Choosing k
'''
# percentage of users to use
percent = 0.01
# number of neighbors for our new user
k       = int(ratings.shape[0] * percent)
print(f"k = {k}\n")

'''
Creating a new user
'''
n_columns    = 100
fill_value   = 99.0
joke_indices = [8]
new_scores   = [5]
new_user     = create_new_user(joke_indices, new_scores, fill_value, n_columns)
print(f"Our new user looks like:\n{new_user}\n")

'''
```

```
Getting the k-nearest neighbors
'''
k_neighbors_simple = list(abs(ratings.iloc[:,joke_indices[0]+1] -␣
 ↪new_scores[0]).sort_values()[:k].index)
print(f"{k} nearest neighbors are:\n{k_neighbors_simple}")

'''
Getting m jokes
'''
recommended_jokes = get_jokes(ratings, new_user, k_neighbors_simple, m,␣
 ↪fill_value, columns)
for joke in recommended_jokes:
    print(f"Joke {joke}: {jokes.iloc[joke].values[0]}\n")
```

```
Joke 8: A country guy goes into a city bar that has a dress code, and the maitre
d'  demands he wear a tie. Discouraged, the guy goes to his car to sulk when
inspiration strikes: He's got jumper cables in the trunk! So he wraps them
around his neck, sort of like a string tie (a bulky string tie to be sure) and
returns to the bar. The maitre d' is reluctant, but says to the guy, "Okay,
you're a pretty resourceful fellow, you can come in… but just don't start
anything"!

m = 10

k = 249

Our new user looks like:
[1, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 5, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0]

249 nearest neighbors are:
[19944, 18086, 24121, 22917, 4199, 13115, 13438, 15063, 11727, 23661, 6988,
5403, 10780, 23619, 17038, 5000, 1332, 8332, 3771, 13855, 13619, 20470, 17947,
15528, 22389, 13109, 24387, 399, 20844, 13174, 22500, 15702, 2121, 3263, 5990,
17743, 14898, 14890, 21349, 83, 4821, 2141, 1605, 819, 21833, 4946, 10842, 4981,
12425, 18933, 21071, 17116, 12553, 16433, 16394, 17167, 8223, 15798, 18509,
12652, 19142, 24140, 757, 5394, 19335, 3445, 21408, 4020, 19361, 9153, 7036,
15808, 22101, 4327, 3886, 13884, 6845, 23224, 13642, 9949, 4262, 6709, 14578,
9737, 24896, 14303, 6961, 6213, 19173, 8726, 23755, 14606, 22453, 23132, 17598,
19545, 11901, 14436, 15257, 13314, 7781, 2578, 22502, 9493, 3367, 7356, 2383,
19388, 1558, 24643, 18390, 1258, 15604, 6831, 23505, 6448, 17390, 15469, 7656,
```

9853, 2063, 11302, 18263, 7358, 771, 13977, 22736, 7243, 13477, 20518, 17016, 8113, 12400, 18790, 2244, 2181, 8940, 17940, 20486, 16760, 13561, 13593, 20717, 20055, 3915, 10967, 16787, 6029, 1797, 4326, 11971, 7180, 1124, 13408, 7129, 21831, 17707, 674, 21081, 14679, 1789, 6681, 21903, 11223, 17415, 12100, 12884, 309, 20619, 5469, 5470, 12571, 18738, 6762, 11943, 10722, 20375, 17071, 6728, 2788, 7729, 21992, 2163, 23005, 14132, 12737, 20489, 24152, 23929, 11874, 36, 13387, 19828, 19853, 50, 13267, 6079, 9864, 9990, 5724, 8556, 4413, 9258, 14865, 14774, 19998, 10321, 22323, 23577, 10353, 15135, 7495, 6382, 20699, 14519, 24808, 2930, 7009, 9322, 14151, 4982, 19754, 15038, 20804, 11990, 17737, 13415, 977, 1368, 20214, 13555, 22734, 8843, 3877, 13664, 16810, 12331, 18710, 16780, 8908, 18188, 20681, 283, 8498, 23074, 12374, 10997, 19835, 14969]
Joke 9: Two cannibals are eating a clown, one turns to other and says: "Does this taste funny to you?

Joke 50: Did you hear that Clinton has announced there is a new national bird? The spread eagle.

Joke 36: A Jewish young man was seeing a psychiatrist for an eating and sleeping disorder. "I am so obsessed with my mother… As soon as I go to sleep, I start dreaming, and everyone in my dream turns into my mother. I wake up in such a state, all I can do is go downstairs and eat a piece of toast." The psychiatrist replies: "What, just one piece of toast, for a big boy like you?"

Joke 62: An engineer, a physicist and a mathematician are sleeping in a room. There is a fire in the room. The engineer wakes up, sees the fire, picks up the bucket of water and douses the fire and goes back to sleep. Again there is fire in the room. This time, the physicist wakes up, notices the bucket, fills it with water, calculates the optimal trajectory and douses the fire in minimum amount of water and goes back to sleep. Again there is fire. This time the mathematician wakes up. He looks at the fire, looks at the bucket and the water and exclaims, "A solution exists" and goes back to sleep.

Joke 27: A mechanical, electrical and a software engineer from Microsoft were driving through the desert when the car broke down. The mechanical engineer said "It seems to be a problem with the fuel injection system, why don't we pop the hood and I'll take a look at it." To which the electrical engineer replied, "No I think it's just a loose ground wire, I'll get out and take a look." Then, the Microsoft engineer jumps in. "No, no, no. If we just close up all the windows, get out, wait a few minutes, get back in, and then reopen the windows everything will work fine."

Joke 66: Once upon a time, two brooms fell in love and decided to get married. Before the ceremony, the bride broom informed the groom broom that she was expecting a little whiskbroom. The groom broom was aghast! "How is this possible?" he asked. "We've never swept together!

Joke 10: Q. What do a hurricane, a tornado, and a redneck divorce all have in common? A. Someone's going to lose their trailer…

Joke 54: A woman has twins, and gives them up for adoption. One of them goes to a family in Egypt and is named "Amal." The other goes to a family in Spain; they name him "Juan." Years later, Juan sends a picture of himself to his mom. Upon receiving the picture, she tells her husband that she wishes she also had a picture of Amal. Her husband responds, "But they are twins-if you've seen Juan, you've seen Amal.

Joke 69: Employer to applicant: "In this job we need someone who is responsible." Applicant: "I'm the one you want. On my last job, every time anything went wrong, they said I was responsible."

Joke 68: This guys wife asks, "Honey if I died would you remarry?" and he replies, "Well, after a considerable period of grieving, we all need companionship, I guess I would." She then asks, "If I died and you remarried, would she live in this house?" and he replies, "We've spent a lot of time and money getting this house just the way we want it. I'm not going to get rid of my house, I guess she would." "If I died and you remarried, and she lived in this house, would she sleep in our bed?" and he says, "That bed is brand new, we just paid two thousand dollars for it, it's going to last a long time, I guess she would." So she asks, "If I died and you remarried, and she lived in this house, and slept in our bed, would she use my golf clubs?" "Oh no, she's left handed."

[9]:
```
'''
Making sure both solutions match, and while they may order the things␣
 ↪differently, they are both correct solutions
I would say.
'''
s   = set(k_neighbors_simple)
print("S:")
for i in k_neighbors_simple:
    print(ratings.iloc[i,joke_indices[0]+1], end=" ")
c   = set(k_neighbors_complex)
print("\nC:")
for i in k_neighbors_complex:
    print(ratings.iloc[i,joke_indices[0]+1], end=" ")
s_n = len(s)
c_n = len(c)
d   = s - c
d_n = len(d)

'''As it turns out, both methods work well, however, my functional solution␣
 ↪will be easier to implement with the weights'''
```

```
S:
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.05 4.95 5.05 5.05 5.05 4.95 4.95 5.05 4.95 4.95 5.05 5.05 4.95
```

```
5.05 5.05 5.05 5.05 4.95 4.95 4.95 5.05 4.95 4.95 5.05 4.95 4.95 4.95 5.05 4.95
5.05 5.05 4.95 5.05 5.05 4.95 5.05 5.05 5.05 4.95 5.05 4.95 4.95 4.95 5.05 4.95
4.95 5.05 4.95 4.95 4.95 5.05 4.95 4.95 5.05 5.05 4.95 4.95 4.95 5.05 4.95 4.95
4.95 4.95 4.95 5.05 5.1 4.9 5.1 4.9 4.9 5.1 4.9 5.1 5.1 5.1 4.9 5.1 4.9 4.9 5.1
5.1 4.9 5.1 4.9 4.9 4.9 5.1 5.1 4.9 5.1 4.9 5.1 5.1 4.9 4.9 4.9 4.9 4.9 4.9 5.1
4.9 4.9 4.9 5.1 5.1 4.9 5.1 4.9 4.9 4.9 4.9 5.1 4.9 4.9 5.1 4.9 4.9 4.9 4.9 5.1
5.1 4.9 5.1 5.1 5.1 5.1 4.9 5.1 5.1 5.1 5.1 4.9 4.9 4.9 4.9 4.9 5.1 4.9 4.85
5.15 5.15 5.15 5.15 4.85 4.85 4.85 5.15 5.15 5.15 5.15 4.85 5.15 4.85 5.15 5.15
5.15 4.85 5.15 4.85 4.85 5.15 5.15 4.85 4.85 4.85 5.15 4.85 5.15 4.85 4.85 5.15
5.15 4.85 4.85 5.15 5.15 5.15 5.15 5.15 4.85 5.15 4.85 4.85 4.85 5.15 5.15 5.15
4.85 4.85 5.15 4.85 5.15 4.85 5.15 4.85 4.81 4.81 5.19 5.19 4.81 4.81 5.19 4.81
4.81 4.81 4.81 4.81 4.81 5.19 4.81 4.81 4.81 4.81 4.81 5.19 4.81 4.81 5.19 4.81
5.19 5.19 5.19 4.81 4.81 4.81 4.81
C:
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 4.95 4.95 4.95 5.05 4.95 5.05 4.95 4.95 5.05 4.95 4.95 5.05 4.95
4.95 5.05 5.05 5.05 5.05 4.95 4.95 5.05 5.05 4.95 4.95 4.95 4.95 4.95 5.05 5.05
4.95 4.95 5.05 4.95 5.05 5.05 4.95 4.95 5.05 5.05 4.95 4.95 4.95 5.05 4.95 4.95
4.95 5.05 4.95 4.95 5.05 5.05 4.95 5.05 5.05 5.05 5.05 4.95 5.05 4.95 4.95 5.05
5.05 4.95 4.95 5.05 4.9 4.9 5.1 4.9 4.9 5.1 5.1 4.9 4.9 5.1 5.1 4.9 5.1 5.1 5.1
5.1 4.9 4.9 4.9 4.9 4.9 5.1 4.9 4.9 4.9 5.1 5.1 4.9 4.9 4.9 4.9 5.1 5.1 4.9 5.1
5.1 4.9 4.9 4.9 5.1 5.1 5.1 4.9 4.9 4.9 5.1 5.1 4.9 5.1 5.1 5.1 4.9 5.1 5.1 4.9
4.9 4.9 5.1 5.1 4.9 5.1 5.1 4.9 4.9 4.9 4.9 4.9 5.1 4.9 4.9 4.9 4.9 5.1 4.85
4.85 5.15 4.85 4.85 4.85 5.15 5.15 5.15 4.85 5.15 5.15 5.15 5.15 5.15 5.15 5.15
5.15 5.15 5.15 5.15 4.85 5.15 4.85 4.85 4.85 4.85 5.15 5.15 5.15 4.85 5.15 4.85
4.85 4.85 4.85 4.85 4.85 5.15 4.85 5.15 4.85 4.85 4.85 4.85 5.15 5.15 5.15 4.85
4.85 5.15 4.85 5.15 5.15 5.15 5.15 5.15 4.81 5.19 4.81 4.81 4.81 4.81 4.81 4.81
5.19 5.19 4.81 4.81 4.81 4.81 4.81 4.81 5.19 5.19 5.19 4.81 5.19 5.19 4.81 4.81
5.19 5.19 4.81 4.81 5.19 5.19 5.19
```

[9]:
```
'As it turns out, both methods work well, however, my functional solution will
 be easier to implement with the weights'
```

Part C:

Justify why you picked these values for `m` and `k`.

In general, how would your recommendation change with changing `k` and keeping `m` constant? What effects do increases in `k` cause? What about decreases?

Well, I've chosen `k` differently, using a percentage of users. I've tried 0.1 which took a while to compute, 0.01 which I think to be the sweespot, and 0.005 which seemed too small a sample size. Larger `k` takes more time, but theoretically, I'd think it would be a bigger 'battle' between jokes when we raise `k` and keep `m` constant, and inversely, we'd have less competition when we decrease `k`.

As for `m`, I also chose to take a percentage of all the jokes, and I chose 0.1 or 10 of jokes simply because I wanted to read them. I think a smaller choice of `m` might've made more sense because

not all of the jokes were necessarily all that similar in style.

Part D: Now try it yourself. Read jokes 1-4 and rate them all. Create a collaborative filter for yourself. What are the top 3 jokes recommended to you? What do you think of them?

[10]:
```
'''
Jokes 1, 2, 3, 4
'''
joke_nums = [1,2,3,4]
for joke_i in joke_nums:
    print(f"Joke {joke_i}: {jokes.iloc[joke_i].values[0]}\n")

'''
Choosing m
'''
m = 3 # we're given this
print(f"m = {m}\n")

'''
Choosing k
'''
# percentage of users to use
percent = 0.01
# number of neighbors for our new user
k       = int(ratings.shape[0] * percent)
print(f"k = {k}\n")

'''
Creating a new user
'''
n_columns    = 100
fill_value   = 99.0
joke_indices = [i for i in joke_nums]
print(joke_indices)

'''NOTE'''
# Although some of these jokes can definitely be viewed as offensive,
# I rated them by which ones ellicited the largest response.
# Plus I happen to enjoy some good 'ole fashoined dark humor.
new_scores   = [7, 3.5, 0, 6]


new_user     = create_new_user(joke_indices, new_scores, fill_value, n_columns)
print(f"Our new user looks like:\n{new_user}\n")

'''
```

```python
Getting the k-nearest neighbors
'''
# The actual columns the jokes are stored in
columns     = [i for i in range(1, n_columns + 1)]
k_neighbors = neighbors(ratings, new_user, k, fill_value, columns)
print(f"{k} nearest neighbors are:\n{k_neighbors_complex}")

'''
Getting m jokes
'''
recommended_jokes = get_jokes(ratings, new_user, k_neighbors, m, fill_value,␣
 ↪columns)
for joke in recommended_jokes:
    print(f"Joke {joke}: {jokes.iloc[joke].values[0]}\n")
```

```
Joke 1: This couple had an excellent relationship going until one day he came
home from work to find his girlfriend packing. He asked her why she was leaving
him and she told him that she had heard awful things about him.   "What could
they possibly have said to make you move out?"   "They told me that you were a
pedophile."   He replied, "That's an awfully big word for a ten year old."

Joke 2: Q. What's 200 feet long and has 4 teeth?   A. The front row at a Willie
Nelson Concert.

Joke 3: Q. What's the difference between a man and a toilet?   A. A toilet
doesn't follow you around after you use it.

Joke 4: Q.      What's O. J. Simpson's Internet address?  A.    Slash, slash,
backslash, slash, slash, escape.

m = 3

k = 249

[1, 2, 3, 4]
Our new user looks like:
[4, 99.0, 7, 3.5, 0, 6, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0]

249 nearest neighbors are:
[17947, 18086, 13115, 5403, 6988, 20470, 13855, 19944, 24121, 4199, 15063,
```

```
22917, 23619, 5000, 8332, 13619, 1332, 23661, 17038, 11727, 10780, 3771, 13438,
22389, 6961, 22101, 23224, 5394, 18509, 16394, 2121, 12553, 18933, 24140, 13109,
21833, 4020, 15528, 17167, 9153, 21349, 19361, 1605, 14890, 12652, 9949, 17116,
4981, 4946, 14578, 6845, 5990, 13642, 12425, 14898, 20844, 24387, 19142, 4262,
7036, 8223, 757, 3886, 15808, 17743, 4327, 15798, 4821, 14303, 819, 13174, 3445,
21071, 83, 19335, 21408, 3263, 16433, 22500, 24896, 6709, 2141, 9737, 6213, 399,
13884, 15702, 10842, 2063, 11302, 14679, 9853, 674, 13408, 24643, 18790, 771,
20717, 20518, 2578, 6448, 19173, 14436, 10967, 13977, 20486, 2383, 8940, 21081,
23505, 13593, 6831, 13561, 13314, 15604, 13477, 2181, 14606, 3367, 22736, 6029,
19388, 2244, 23755, 1558, 17598, 17016, 18390, 7356, 17940, 15257, 7358, 8726,
19545, 11971, 17390, 18263, 9493, 20055, 1258, 22502, 11901, 22453, 12400, 1789,
1797, 23132, 3915, 7243, 16787, 16760, 17707, 7129, 21831, 7656, 1124, 4326,
15469, 8113, 7781, 7180, 20619, 2788, 11943, 15135, 19998, 6681, 10353, 9990,
6382, 20489, 23577, 12100, 17415, 50, 7729, 8556, 5724, 36, 6728, 4413, 19853,
19828, 14132, 20699, 24808, 13267, 6762, 18738, 21903, 11223, 12884, 9864,
13387, 21992, 309, 2163, 24152, 6079, 17071, 12737, 14865, 9258, 10321, 14774,
7009, 20375, 5469, 5470, 10722, 7495, 23929, 11874, 22323, 12571, 23005, 2930,
14519, 5524, 19754, 23775, 20214, 5331, 5712, 8908, 19835, 3447, 5886, 15038,
8843, 20325, 14969, 5850, 704, 15244, 22734, 10535, 977, 17141, 4982, 24867,
9136, 11207, 2144, 9856, 13415, 232, 18710, 11990]
Joke 5: Bill & Hillary are on a trip back to Arkansas. They're almost out of
gas, so Bill pulls into a service station on the outskirts of town. The
attendant runs out of the station to serve them when Hillary realizes it's an
old boyfriend from high school. She and the attendant chat as he gases up their
car and cleans the windows. Then they all say good-bye.   As Bill pulls the car
onto the road, he turns to Hillary and says, 'Now aren't you glad you married me
and not him ? You could've been the wife of a grease monkey !'   To which
Hillary replied, 'No, Bill. If I would have married him, you'd be pumping gas
and he would be the President !'

Joke 50: Did you hear that Clinton has announced there is a new national bird?
The spread eagle.

Joke 36: A Jewish young man was seeing a psychiatrist for an eating and sleeping
disorder.   "I am so obsessed with my mother… As soon as I go to sleep, I
start dreaming, and everyone in my dream turns into my mother. I wake up in such
a state, all I can do is go downstairs and eat a piece of toast."  The
psychiatrist replies:  "What, just one piece of toast, for a big boy like you?"
```

Part E:

Read some more of the jokes. Repeat your favorite(s) here:

[11]:
```
favs = [1, 4, 27, 32, 53]
for joke in favs:
    print(f"Joke {joke}: {jokes.iloc[joke].values[0]}\n")
```

```
Joke 1: This couple had an excellent relationship going until one day he came
```

home from work to find his girlfriend packing. He asked her why she was leaving him and she told him that she had heard awful things about him. "What could they possibly have said to make you move out?" "They told me that you were a pedophile." He replied, "That's an awfully big word for a ten year old."

Joke 4: Q. What's O. J. Simpson's Internet address? A. Slash, slash, backslash, slash, slash, escape.

Joke 27: A mechanical, electrical and a software engineer from Microsoft were driving through the desert when the car broke down. The mechanical engineer said "It seems to be a problem with the fuel injection system, why don't we pop the hood and I'll take a look at it." To which the electrical engineer replied, "No I think it's just a loose ground wire, I'll get out and take a look." Then, the Microsoft engineer jumps in. "No, no, no. If we just close up all the windows, get out, wait a few minutes, get back in, and then reopen the windows everything will work fine."

Joke 32: What do you call an American in the finals of the world cup? "Hey Beer Man!"

Joke 53: The Pope dies and, naturally, goes to heaven. He's met by the reception committee, and after a whirlwind tour he is told that he can enjoy any of the myriad of recreations available. He decides that he wants to read all of the ancient original text of the Holy Scriptures, so he spends the next eon or so learning languages. After becoming a linguistic master, he sits down in the library and begins to pour over every version of the Bible, working back from most recent "Easy Reading" to the original script. All of a sudden there is a scream in the library. The Angels come running in only to find the Pope huddled in his chair, crying to himself and muttering, "An 'R'! The scribes left out the 'R'." A particularly concerned Angel takes him aside, offering comfort, asks him what the problem is and what does he mean. After collecting his wits, the Pope sobs again, "It's the letter 'R'. They left out the 'R'. The word was supposed to be CELEBRATE!"

Back to top # Extra Credit (Running BigCLAM on Big Data: up to +25 pts). This problem may be turned in either with HW7 or HW8.

Consider the data set `Marvel_Network`. This set consists of two columns, hero1 and hero2. Every row is filled with co-occurrence of two such marvel characters in a comic.

1.1.1 Part A) (10 pts) Cleaning and setup

(i) Use item baskets to count how many times each character appears. You may also want to count how many times each *edge* between pairs of characters appears (use your code from A Priori!) Print the most popular 5 characters' names.

[12]:
```
#Names and counts
```

(ii) Remove any characters having degree = 1. What do these nodes represent?

[ ]:

(iii) For speed of computation, you should also remove any *edges* with count 1. What do these *edges* represent?

[ ]:

(iv) Create an adjacency matrix (or probably a compact/sparse representation of such a matrix!) for the data after accounting for parts (ii), (iii). How many characters are there? How many edges?

[ ]:

1.1.2 Part B) (10 pts) Detect communities in the graph generated above using BigCLAM

For this problem, take the number of communities to find as 4 (plus a background community!). Use your choice of initialization. If you have **prior knowledge** of any comic book characters, consider setting 4 characters from "different" Marvel universes as your seeds. If not, either use PageRank or properties of your adjacency matrix to decide where to initialize affiliations.

Are the final communities ones you've seen in popular culture? Verify by checking the community affiliation scores of two characters that you believe should share a dominant community, like Wolverine and Professor X.

[ ]:

1.1.3 Part C) (5 pts) Detect communities in the graph generated above using *weighted* BigCLAM

It turns out that some edges occurred multiple times in the data set. We can adjust our model to count the edges proportionately to the number of times they occured: we just have to weight the partial derivatives in our gradient calculation by multiplying each term v in ∇F_u by the number of times u and v were seen together.

Run the model accordingly, and report the community affiliations of the same pair of characters you looked at in Part B). They should be even closer now, right?

[ ]: