Assignment 4.1 Kevin Thompson

March 14, 2024

0.0.1 Assignment 4 DSC 478

Kevin Thompson This notebook is an example of recommender systems using two CSV files. One with a list of 100 jokes and another with 1000 user ratings on these 100 jokes.

[66]:
```
import pandas as pd
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
from numpy import linalg as la
```

[83]:
```
jokes = pd.read_csv('C:/Users/kthom/Downloads/jokes.csv', usecols=[1], header =␣
↪None)
joke_rating = pd.read_csv('C:/Users/kthom/Downloads/modified_jester_data.csv',␣
↪delimiter=',', header = None)
```

[84]:
```
print(jokes.head())
jokes.shape
```

```
                                                   1
0  A man visits the doctor. The doctor says "I ha…
1  This couple had an excellent relationship goin…
2  Q. What's 200 feet long and has 4 teeth? A. Th…
3  Q. What's the difference between a man and a t…
4  Q. What's O. J. Simpson's Internet address? A…
```

[84]:
```
(100, 1)
```

[85]:
```
print(joke_rating.head(10))
joke_rating.shape
```

```
      0      1      2      3      4      5      6      7      8      9   … \
0   3.18  19.79   1.34   2.84   3.48   2.50   1.15  15.17   2.02   6.24  …
1  15.08  10.71  17.36  15.37   8.62   1.34  10.27   5.66  19.88  20.22  …
2   0.00   0.00   0.00   0.00  20.03  20.27  20.03  20.27   0.00   0.00  …
3   0.00  19.35   0.00   0.00  12.80  19.16   8.18  17.21   0.00  12.84  …
4  19.50  15.61   6.83   5.61  12.36  12.60  18.04  15.61  10.56  16.73  …
5   4.83   7.46  11.44   2.50   3.91   6.68   2.31  10.13   4.35   9.20  …
6   0.00   0.00   0.00   0.00  19.59   1.15  18.72  19.79   0.00   0.00  …
7  17.84  14.16  20.17   4.79   2.84   9.30  20.27  12.41   5.81   6.58  …
```

```
8   7.21   7.46   1.58   4.11   2.26  10.71   5.71   2.07   3.14   9.40  …
9  14.01  16.15  16.15  14.01  17.41  16.15  19.93  13.52  14.01  19.16  …

      90     91     92     93     94     95     96     97     98     99
0  13.82   0.00   0.00   0.00   0.00   0.00   5.37   0.00   0.00   0.00
1  13.82   6.05  10.71  18.86  10.81   8.86  14.06  11.34   6.68  12.07
2   0.00   0.00   0.00  20.08   0.00   0.00   0.00   0.00   0.00   0.00
3   0.00   0.00   0.00  11.53   0.00   0.00   0.00   0.00   0.00   0.00
4  16.19  16.58  15.27  16.19  16.73  12.55  14.11  17.55  12.80  12.60
5   7.46   4.11  10.32   8.04   8.82   7.65  11.05   1.92   5.95   7.55
6   0.00   0.00   0.00   0.00   0.00  13.33   0.00   0.00   0.00   0.00
7  18.23   9.88  10.90   5.32   7.84   7.65  13.14  10.95  12.31  11.00
8  15.37  10.71  15.17  10.71  10.71  10.71  10.71  10.71   7.60   6.05
9   0.00  15.47   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00

[10 rows x 100 columns]
```

[85]:
```
(1000, 100)
```

[86]:
```
users_rating = joke_rating.to_numpy() # change to np array #could also do:␣
↪genfromtxt('modified_jester_data.csv',delimiter=',') from start
```

[44]:
```
#different simularity functions
def ecludSim(inA,inB):
    return 1.0 / (1.0 + la.norm(inA - inB))

def pearsSim(inA,inB):
    if len(inA) < 3 : return 1.0
    return 0.5 + 0.5 * corrcoef(inA, inB, rowvar = 0)[0][1]

def cosSim(inA,inB):
    num = float(inA.T * inB)
    denom = la.norm(inA)*la.norm(inB)
    return 0.5 + 0.5 * (num / denom)
```

[61]:
```
def standEst(dataMat, user, simMeas, item):
    n = np.shape(dataMat)[1]
    simTotal = 0.0; ratSimTotal = 0.0
    for j in range(n):
        userRating = dataMat[user,j]
        if userRating == 0: continue
        overLap = nonzero(logical_and(dataMat[:,item]>0, \
                                      dataMat[:,j]>0))[0]
        if len(overLap) == 0: similarity = 0
        else: similarity = simMeas(dataMat[overLap,item], \
                                   dataMat[overLap,j])
        #print 'the %d and %d similarity is: %f' % (item, j, similarity)
        simTotal += similarity
```

```
        ratSimTotal += similarity * userRating
    if simTotal == 0: return 0
    else: return ratSimTotal/simTotal
```

[46]:
```
#cross validate user to other users in a data matrix
def cross_validate_user(dataMat, user, test_ratio, estMethod=standEst,␣
 ↪simMeas=pearsSim):
        number_of_items = np.shape(dataMat)[1]
        rated_items_by_user = np.array([i for i in range(number_of_items) if␣
 ↪dataMat[user,i]>0])
        test_size = round(test_ratio * len(rated_items_by_user))
        test_indices = np.random.randint(0, len(rated_items_by_user), test_size)
        withheld_items = rated_items_by_user[test_indices]
        original_user_profile = np.copy(dataMat[user])
        dataMat[user, withheld_items] = 0 # So that the withheld test items is␣
 ↪not used in the rating estimation below
        error_u = 0.0
        count_u = len(withheld_items)

        # Compute absolute error for user u over all test items
        for item in withheld_items:
                # Estimate rating on the withheld item
                estimatedScore = estMethod(dataMat, user, simMeas, item)
                error_u = error_u + abs(estimatedScore -␣
 ↪original_user_profile[item])

        # Now restore ratings of the withheld items to the user profile
        for item in withheld_items:
                dataMat[user, item] = original_user_profile[item]

        # Return sum of absolute errors and the count of test cases for this␣
 ↪user
        # Note that these will have to be accumulated for each user to compute␣
 ↪MAE
        return error_u, count_u
```

[64]:
```
def test(dataMat, test_ratio, estMethod):
    """Write this function to iterate over all users and for each perform␣
 ↪evaluation by calling
        the above cross_validate_user function on each user. MAE will be the␣
 ↪ratio of total error
        across all test cases to the total number of test cases, for all␣
 ↪users"""
    totalError = 0.0 # initialize variables to count errors for MAE
    totalCount = 0.0
    for i in range(users_rating.shape[0]):
```

```
        error_u, count_u = cross_validate_user(dataMat, i, test_ratio,␣
↪estMethod=standEst, simMeas=pearsSim)
        totalError += error_u
        totalCount += count_u
    Mae= totalError/totalCount


    print('Mean Absoloute Error for ',str(estMethod),' : ', Mae)
    return Mae
```

[68]:
```
test(users_rating, .2, standEst)
```

```
Mean Absoloute Error for  <function standEst at 0x000001ECF075E2A0>  :
3.7441410338925483
```

[68]:
```
3.7441410338925483
```

MAE of 3.7441 is decent given the context of our rating scale. On a scale of 1-21, our prediction will be within 3.7… of the actual rating. While this isn't extremely accurate, it would definitely give a pretty close estimate of one's opion. Also, it is hard to accurately give a rating on a scale of something subjective like a joke anyway, so I would argue one's actual rating is already within a ballpark range – so in that context this mae is sufficient

[199]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.neighbors import NearestNeighbors
def print_most_similar_jokes(rating_data, jokeID, k, simularity_function):
    """This function will take one of the jokes' rating data and 5 the k␣
↪nearest neighbors. It will then print out the joke
    we are using to compare as well as the k nearest neighbors' joke text."""

    def get_joke_text(jokes, id):
        """function to get joke from a list of jokes with a specified id"""
        return np.array(jokes)[id]

    k = k+1 # i dont want to include the joke we are comparing, as that will␣
↪obviously be the most similar (it to itself)
    neigh = NearestNeighbors(n_neighbors=k)
    neigh.fit(rating_data.T)
    distances, indices = neigh.kneighbors([rating_data.T[jokeID]])
    nearest_neighbors = rating_data.T[indices[0]]

    similarJokes = [] #initialize empty list to collect which joke #s we have␣
↪found
    for i in nearest_neighbors[1:]:
        for a in range(len(users_rating.T)):
            if np.array_equal(users_rating.T[a], i):
                similarJokes.append(a)
```

```
    print("Selected Joke: \n\n", get_joke_text(jokes,jokeID),'\n\nTop 5 Similar␣
↪Jokes:\n')
    for i in similarJokes:
        if get_joke_text(jokes,i)[0][0] == 'Q':
            print(get_joke_text(jokes,i)[0],'\n')
        else:
            print('Q:  '+get_joke_text(jokes,i)[0],'\n')
```

[200]:
```
print_most_similar_jokes(users_rating, 15, 5, ecludSim)
```

```
Selected Joke:

 ['Q. What is orange and sounds like a parrot?  A. A carrot.']

Top 5 Similar Jokes:

Q:  A dog walks into Western Union and asks the clerk to send a telegram. He
fills out a form on which he writes down the telegram he wishes to send: "Bow
wow wow Bow wow wow."The clerk says "You can add another 'Bow wow' for the same
price."The dog responded "Now wouldn't that sound a little silly?"

Q:  How many men does it take to screw in a light bulb? One…men will screw
anything.

Q:  What did the blind person say when given some matzah?A:  Who the hell wrote
this?

Q. Did you hear about the dyslexic devil worshipper? A. He sold his soul to
Santa.

Q:  They asked the Japanese visitor if they have elections in his country.
"Every Morning" he answers.
```