hw5

October 24, 2021

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
"""
Task:
Predict which song a user would like based on his ratings
Assign 3 values of 1 for songs you recommend, and 3 values of 0 for songs you␣
 ↪don't recommend
Find AUC

Data format:
Train: User | Item | Rating
Note that item could be anything (track, album, artist, genre)
Test: User | Track | Album | Artist | Genres...

0th Approach:
- add if trainUserID > userID: section
- this is enough for over 0.8 rating on Kaggle


First Approach:
- Use mean data to find best rated songs
- Use median
- Use Max/Min (the 3 highest overall score)

Second Approach:
- Use weighed approach. If Track > Artist > Album > Genre. Perhaps a 30/25/20/15␣
 ↪weight for all of these and add up numbers

Third Approach:
- YOLO and give give everything random numbers? 50% auc theoretically
"""
```

[2]:
```
"\nTask:\nPredict which song a user would like based on his ratings\nAssign 3
values of 1 for songs you recommend, and 3 values of 0 for songs you don't
recommend\nFind AUC\n\nData format:\nTrain: User | Item | Rating\nNote that item
could be anything (track, album, artist, genre)\nTest: User | Track | Album |
Artist | Genres...\n\nFirst Approach:\n- Use mean data to find best rated
```

```
songs\n- Use median\n- Use Max/Min (the 3 highest overall score)\n\nSecond
Approach:\n- Use weighed approach. If Track > Artist > Album > Genre. Perhaps a
30/25/20/15 weight for all of these and add up numbers\n\nThird Approach:\n-
YOLO and give give everything random numbers? 50% auc theoretically\n"
```

[3]:

```
file_test = 'testTrack_hierarchy.txt'
file_train = 'trainIdx2_matrix.txt'
output_file= 'output.txt'
format_output = 'submission.txt'
```

[4]:

```
fTest = open(file_test, 'r')
fTrain = open(file_train, 'r')
Trainline = fTrain.readline()
fOutput = open(output_file, 'w')
fFormat = open(format_output,'w')
fOutput.write('userID'+ '|' +'trackID'+ '|' + 'recommendation'+ '|'
           + 'album'+ '|' + 'artist'+ '|' + 'num_genre_ratings' + '|'
           +  'mean' + '\n')
fFormat.write('TrackID' + '|' + 'Predictor' + '\n')
```

[4]:

```
18
```

[5]:

```
trackID_vec=[0]*6
albumID_vec=[0]*6
artistID_vec=[0]*6
lastUserID=-1
mean_vec=[0]*6
num_genres_vec=[0]*6
```

[6]:

```
user_rating_inTrain=np.zeros(shape=(6,3))
user_rating_inTrain
```

[6]:

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

[7]:

```
for line in fTest:
    arr_test=line.strip().split('|') #this strips the line at | and makes into array
    userID= arr_test[0]
    trackID= arr_test[1]
    albumID= arr_test[2]
    artistID=arr_test[3]
    mean = 0
    sum = 0
    num_genres = 0
    genres = []
    #ii = 0
#Problem: genre may exceed 1, need to append to array
    if len(arr_test) > 4:
        num_genres = len(arr_test) - 4 #total num genres
        #genres = [] #create empty array
        for i in range(4, len(arr_test)):
            genres.append([arr_test[i]])

    if userID!= lastUserID: #resets the userId
        ii=0
        user_rating_inTrain=np.zeros(shape=(6,3))

    trackID_vec[ii]=trackID
    albumID_vec[ii]=albumID
    artistID_vec[ii]=artistID
    num_genres_vec[ii]=genres #won't show actual rating
    mean_vec[ii] = mean
    ii=ii+1 #increases until 6. How does it know to stop at 6? If statement␣
 ↪below? Or user_rating shape
    lastUserID=userID

    if ii==6:
        while (Trainline):
            # for Trainline in fTrain:
            arr_train = Trainline.strip().split('|')
            #userId in test, trainUserID in train files
            trainUserID=arr_train[0]
            trainItemID=arr_train[1]
            trainRating=arr_train[2]
            Trainline=fTrain.readline()

            if trainUserID < userID: #train is less than userId, meaning that␣
 ↪userId doesn't have that itemId, goes to next
                continue
            if trainUserID == userID:
                for nn in range(0, 6):
                    if trainItemID==albumID_vec[nn]:
                        user_rating_inTrain[nn, 0]=trainRating
                    if trainItemID==artistID_vec[nn]:
                        user_rating_inTrain[nn, 1]=trainRating
            if trainUserID > userID:
                #[int(num, base=16) for num in fTest]
                #int(arr_train.translate(str.maketrans({'|':" "})), 16)
                #sum = (user_rating_inTrain[nn,0]) + (user_rating_inTrain[nn,␣
↪1]) + (num_genres_vec[nn]) #+ (mean_vec[nn])
                for nn in range(0, 6):
                    if user_rating_inTrain[nn,0] > 30 or␣
↪user_rating_inTrain[nn,1] > 30: #if album and artist rating > 35
                    #change to and for next submission
                    #if sum > 100:
                        outStr=str(userID) + '|' + str(trackID_vec[nn])+ '|' +␣
↪'yes' + '|' + str(user_rating_inTrain[nn,0]) + '|' +␣
↪str(user_rating_inTrain[nn, 1]) + '|' + str(num_genres_vec[nn]) #'|' +␣
↪str(mean_vec[nn]
                        formatStr = str(userID) + '_' + str(trackID_vec[nn]) +␣
↪'|' + str(1)
                        fOutput.write(outStr + '\n')
                        fFormat.write(formatStr + '\n')
                    else:
                        outStr=str(userID) + '|' + str(trackID_vec[nn])+ '|' +␣
↪'no' + '|' + str(user_rating_inTrain[nn,0]) + '|' +␣
↪str(user_rating_inTrain[nn, 1]) + '|' + str(num_genres_vec[nn]) #+  '|' +␣
↪str(mean_vec[nn]
                        formatStr = str(userID) + '_' + str(trackID_vec[nn]) +␣
↪'|' + str(0)
                        fOutput.write(outStr + '\n')
                        fFormat.write(formatStr + '\n')
                break
```

[8]:

```
user_rating_inTrain
```

[8]:

```
array([[90., 90.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [90., 90.,  0.],
       [ 0.,  0.,  0.],
       [90., 90.,  0.]])
```

[9]:

```
fTest.close()
fTrain.close()
fOutput.close()
fFormat.close()
```

[10]:

```
reader = pd.read_csv(format_output, delimiter = '|')
reader
```

[10]:

```
              TrackID  Predictor
0       199810_208019          0
1        199810_74139          0
```

```
2          199810_9903          0
3        199810_242681          0
4         199810_18515          0
...                ...        ...
119995    249010_72192          0
119996    249010_86104          0
119997   249010_186634          1
119998   249010_293818          0
119999   249010_262811          1

[120000 rows x 2 columns]
```

[11]:

```
reader.to_csv('submission2.csv', index = False)
```

[ ]: