CSE250 HW8

November 27, 2018

```
In [1]: import numpy as np
        import math
        import matplotlib.pyplot as plt
        import pandas as pd
```

```
In [101]: movie_titles = pd.read_csv('hw8_movieTitles_fa18.txt', header=None)
          PID = pd.read_csv('hw8_studentPIDs_fa18.txt', header=None)
          ratings = pd.read_csv('hw8_ratings_fa18.txt', sep = " ", header=None)
          probZ_ini = pd.read_csv('hw8_probZ_init.txt', header=None)
          probRgivenZ_ini = pd.read_csv('hw8_probRgivenZ_init.txt', sep = "  ", header=None)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5: ParserWarning: Falling back to
  """
```

0.1 (a) Sanity Check

```
In [42]: num_student = ratings.shape[0]
         num_movie = ratings.shape[1]
         popularity = np.zeros((num_movie,1))

         for i in range(num_movie):
             not_seen = 0
             for j in range(num_student):
                 if ratings.values[j,i] == '?':
                     not_seen += 1
                 elif ratings.values[j,i] == '1':
                     popularity[i] += 1
             popularity[i] /= (num_student-not_seen)
```

```
In [53]: index = np.squeeze(np.argsort(popularity,axis=0))
```

```
In [61]: movie_sorted = [movie_titles.values[j,0] for j in index]
```

```
In [62]: movie_sorted
```

```
Out[62]: ['The_Last_Airbender',
          'Fifty_Shades_of_Grey',
          'I_Feel_Pretty',
'Chappaquidick',
'Man_of_Steel',
'Prometheus',
'The_Shape_of_Water',
'Phantom_Thread',
'Magic_Mike',
'World_War_Z',
'Bridemaids',
'American_Hustle',
'Drive',
'The_Hunger_Games',
'Thor',
'Pitch_Perfect',
'Fast_Five',
'Avengers:_Age_of_Ultron',
'Jurassic_World',
'The_Hateful_Eight',
'The_Revenant',
'Dunkirk',
'Star_Wars:_The_Force_Awakens',
'Mad_Max:_Fury_Road',
'Captain_America:_The_First_Avenger',
'The_Perks_of_Being_a_Wallflower',
'Iron_Man_2',
'La_La_Land',
'Manchester_by_the_Sea',
'The_Help',
'Midnight_in_Paris',
'The_Girls_with_the_Dragon_Tattoo',
'21_Jump_Street',
'Frozen',
'Now_You_See_Me',
'X-Men:_First_Class',
'Ex_Machina',
'Harry_Potter_and_the_Deathly_Hallows:_Part_1',
'Toy_Story_3',
'Her',
'The_Great_Gatsby',
'The_Avengers',
'The_Theory_of_Everything',
'Room',
'Gone_Girl',
'Three_Billboards_Outside_Ebbing',
'Les_Miserables',
'Harry_Potter_and_the_Deathly_Hallows:_Part_2',
'The_Martian',
'Avengers:_Infinity_War',
'Darkest_Hour',
'Hidden_Figures',
'12_Years_a_Slave',
'Ready_Player_One',
'Black_Swan',
'Django_Unchained',
'Wolf_of_Wall_Street',
'Shutter_Island',
'Interstellar',
'The_Dark_Knight_Rises',
'The_Social_Network',
'Inception']
```

Comments: This list is consisten with my expectations.

0.2 (e) Implementation

```
In [63]: T = ratings.shape[0]  #num of students
```

```
In [71]: def ComputePit(probZ,probRgivenZ,ratings):
             T = ratings.shape[0]
             Pit = np.zeros((4,T))
             for t in range(T):
                 for i in range(4):
                     num = ComputeNum(probZ,probRgivenZ,ratings,i,t)
                     denom = ComputeDenom(probZ,probRgivenZ,ratings,t)
                     Pit[i,t] = num/denom
             return Pit
```

```
In [100]: def ComputeNum(probZ,probRgivenZ,ratings,i,t):
              P_z_i = probZ[i]
              product = 1
              for n in range(ratings.shape[1]):
                  if ratings[t,n] == '1':
                      product *= probRgivenZ[n,i]
                  elif ratings[t,n] == '0':
                      product *= (1 - probRgivenZ[n,i])
              return P_z_i*product

          def ComputeDenom(probZ,probRgivenZ,ratings,t):
              denom = 0
              for i in range(4):
                  P_z_i = probZ[i]
                  product = 1
                  for n in range(ratings.shape[1]):
                      if ratings[t,n] == '1':
                          product *= probRgivenZ[n,i]
                      elif ratings[t,n] == '0':
                          product *= (1 - probRgivenZ[n,i])
                  denom += P_z_i*product
              return denom
```

```
In [86]: def CalcP_Z(Pit):
             T = Pit.shape[1]
             P_z = np.sum(Pit, axis=1)/T
             return P_z

         def CalcRgivenZ(probRgivenZ,Pit,ratings):
             T = Pit.shape[1]
             denom = np.sum(Pit, axis=1)
             result = probRgivenZ.copy()
             for i in range(4):
                 for j in range(probRgivenZ.shape[0]):
                     num = 0
                     for t in range(T):
                         if ratings[t,j] == '1':
                             num += Pit[i,t]
                         elif ratings[t,j] == '0':
                             num += 0
                         else:
                             num += Pit[i,t]*probRgivenZ[j,i]
                     result[j,i] = num/denom[i]
             return result
```

```
In [87]: def CalcLogLikelihood(probZ,probRgivenZ,ratings):
             T = ratings.shape[0]
             log_l = 0
             for t in range(T):
                 log_l += np.log(ComputeDenom(probZ,probRgivenZ,ratings,t))
             log_l /= T
             return log_l
```

```
In [102]: ratings = ratings.values
          probRgivenZ_ini = probRgivenZ_ini.values
          probZ_ini = probZ_ini.values
```

```
In [103]: L = CalcLogLikelihood(probZ_ini,probRgivenZ_ini,ratings)
          Pit = ComputePit(probZ_ini,probRgivenZ_ini,ratings)
          probZ = CalcP_Z(Pit)
          probRgivenZ = CalcRgivenZ(probRgivenZ_ini,Pit,ratings)
          print('iteration: 0' + '  likelihood: %.5f' % L)

          for i in range(128):
              L = CalcLogLikelihood(probZ,probRgivenZ,ratings)
              Pit = ComputePit(probZ,probRgivenZ,ratings)
              probZ = CalcP_Z(Pit)
              probRgivenZ = CalcRgivenZ(probRgivenZ,Pit,ratings)
```

```
            if (i+1) in (0,1,2,4,8,16,32,64,128):
                print('iteration: %d' % (i+1) + '  likelihood: %.5f' % L)
```

```
iteration: 0  likelihood: -26.67883
iteration: 1  likelihood: -16.09467
iteration: 2  likelihood: -14.28779
iteration: 4  likelihood: -13.26508
iteration: 8  likelihood: -12.84731
iteration: 16  likelihood: -12.70600
iteration: 32  likelihood: -12.64074
iteration: 64  likelihood: -12.61607
iteration: 128  likelihood: -12.59119
```

0.3 (f) Personal Movie Recommendation

```
In [118]: my_index = np.where(PID.values == 'A53287481')
          my_index = np.squeeze(my_index)
          my_index = my_index[0]
```

```
In [129]: N = ratings.shape[1]
          size_omega = 0
          for n in range(N):
              if ratings[my_index,n] == '?':
                  size_omega += 1

          R_l = np.zeros((size_omega,2))
          idx = -1

          for l in range(N):
              if ratings[my_index,l] == '?':
                  idx += 1
                  prob = 0
                  for i in range(4):
                      prob += probRgivenZ[l,i]*Pit[i,my_index]
                  R_l[idx,0] = prob
                  R_l[idx,1] = l
```

```
In [143]: index_new = np.squeeze(np.argsort(R_l[:,0],axis=0))
          movie_sorted_index = [R_l[j,1] for j in index_new]
          unseen_movie_sorted = [movie_titles.values[j.astype(int),0] for j in movie_sorted_in
```

```
In [144]: unseen_movie_sorted
```

```
Out[144]: ['I_Feel_Pretty',
           'The_Last_Airbender',
           'Man_of_Steel',
           'The_Hunger_Games',
           'Phantom_Thread',
           'Fast_Five',
           'Bridemaids',
           'American_Hustle',
           'Star_Wars:_The_Force_Awakens',
           'Prometheus',
           'Magic_Mike',
           'Now_You_See_Me',
           'Manchester_by_the_Sea',
           'X-Men:_First_Class',
           'Toy_Story_3',
           'Darkest_Hour',
           'Drive',
           'Room',
           'The_Theory_of_Everything',
           'Harry_Potter_and_the_Deathly_Hallows:_Part_1',
           'The_Hateful_Eight',
           'Ready_Player_One',
           'The_Girls_with_the_Dragon_Tattoo',
           'Hidden_Figures',
           'Chappaquidick',
           'Midnight_in_Paris',
           'Her',
           '12_Years_a_Slave',
           'Harry_Potter_and_the_Deathly_Hallows:_Part_2',
           '21_Jump_Street',
           'Three_Billboards_Outside_Ebbing',
           'Les_Miserables',
           'Gone_Girl',
           'The_Perks_of_Being_a_Wallflower',
           'The_Help',
           'Black_Swan',
           'Django_Unchained']
```