HW8 Code

November 23, 2021

[1]:
```
import numpy as np

pid = []

with open("hw8_ids.txt", "r") as ids:
    for line in ids.readlines():
        pid.append(line[:-1])

movie = []
with open("hw8_movies.txt", "r") as mvs:
    for line in mvs.readlines():
        movie.append(line[:-1])

rating = []
with open("hw8_ratings.txt", "r") as rts:
    for line in rts.readlines():
        line = line.strip('\n')
        line = line.split(' ')
        rating.append(line)

rating = np.array(rating)
print(rating.shape)

index = {}
for i in range(rating.shape[1]):
    saw = 0
    rec = 0
    for j in range(rating.shape[0]):
        if rating[j][i] != '?':
            saw += 1
        if rating[j][i] == '1':
            rec += 1
    index[i] = rec / saw

index_sorted = sorted(index.items(), key=lambda x: x[1])
movie_sorted = []
for i in range(len(index_sorted)):
    movie_sorted.append(movie[index_sorted[i][0]])
```

```
print(movie_sorted)

# 8.1.e
probR = np.loadtxt("hw8_probR_init.txt")
probZ = np.loadtxt("hw8_probZ_init.txt").flatten()


# returns Omega_t
def seen(t) -> list:
    res = []
    for rate in rating[t]:
        if rate != '?':
            res.append(1)
        else:
            res.append(0)
    return res


# Z R are ndarray
def prior(Z, R, t):
    K = Z.shape[0]  # 4
    O_t = seen(t)
    J = len(movie)  # 76
    sum = 0

    for i in range(K):
        prod = 1
        for j in range(J):
            if O_t[j] == 0:
                continue
            r_jt = int(rating[t][j])
            if r_jt == 1:
                prod *= R[j][i]
            else:
                prod *= 1 - R[j][i]
        sum += Z[i] * prod
    return sum


def rou_it(Z, R, i, t):
    denom = prior(Z, R, t)
    prod = 1
    O_t = seen(t)
    J = len(movie)
    for j in range(J):
        if O_t[j] == 0:
            continue
        r_jt = int(rating[t][j])
        if r_jt == 1:
            prod *= R[j][i]
        else:
            prod *= 1 - R[j][i]
    num = Z[i] * prod
    return num / denom


def likelihood(Z, R):
    sum = 0
    T = len(pid)
    for t in range(T):
        sum += np.log(prior(Z, R, t))
    return sum / T


def update(Z, R, iter):
    res_Z = np.copy(Z)
    res_R = np.copy(R)
    K = Z.shape[0]  # 4

    T = len(pid)  # 362
    J = len(movie)  # 76

    for k in range(iter):  # 256
        if k in [0, 1, 2, 4, 8, 16, 32, 64, 128, 256]:
            print(k)
            ll = likelihood(res_Z, res_R)
            print(ll)

        # rou_it = rou_it_mat[t][i]
        rou_it_mat = np.zeros((362, 4))
        for t in range(362):
            for i in range(4):
                rou_it_mat[t][i] = rou_it(res_Z, res_R, i, t)

        # update P(Z=i)

        for i in range(K):  # 4
            sum = 0
            for t in range(T):
                sum += rou_it_mat[t][i]
            res_Z[i] = sum / T
        # Update P(R_j=1|Z=i)
        for j in range(J):  # 76
            for i in range(K):  # 4
```

```
                denom = 0
                for t in range(T):
                    denom += rou_it_mat[t][i]
                num = 0
                for t in range(T):  # 362
                    if seen(t)[j] == 1:
                        if int(rating[t][j]) == 1:
                            num += rou_it_mat[t][i]
                        else:
                            continue
                    else:
                        num += rou_it_mat[t][i] * res_R[j][i]
                res_R[j][i] = num / denom
    return res_Z, res_R


# takes about 10 minutes to run :(
final_Z, final_R = update(probZ, probR, 257)

prob = [0] * len(movie)
pid = np.array(pid)

row = int(np.where(pid == 'A15058075')[0])
# row = 344
rate = rating[row]
binary_seen = seen(row)
for k in range(len(prob)):
    if binary_seen[k] == 1:
        continue
    sum = 0
    for i in range(4):
        sum += rou_it(final_Z, final_R, i, row) * final_R[k][i]
    prob[k] = sum
index_mine = {}
for i in range(len(prob)):
    if prob[i] == 0:
        continue
    index_mine[i] = prob[i]

index_mine_sorted = sorted(index_mine.items(), key=lambda x: x[1], reverse=True)
movie_mine_sorted = []
for i in range(len(index_mine_sorted)):
    movie_mine_sorted.append(movie[index_mine_sorted[i][0]])
print(movie_mine_sorted)
```

```
(362, 76)
['I_Feel_Pretty', 'Fifty_Shades_of_Grey', 'Hustlers', 'The_Last_Airbender',
'Magic_Mike', 'Fast_&_Furious:_Hobbs_&_Shaw', 'The_Shape_of_Water',
'Prometheus', 'Phantom_Thread', 'World_War_Z', 'Star_Wars:_The_Force_Awakens',
'Rocketman', 'Chappaquidick', 'Bridemaids', 'Man_of_Steel', 'American_Hustle',
'Terminator:_Dark_Fat', 'Room', 'Good_Boys', 'Pokemon_Detective_Pikachu',
'Fast_Five', 'Mad_Max:_Fury_Road', 'Drive', 'Us', 'The_Help', 'Pitch_Perfect',
'Jurassic_World', 'Frozen', 'X-Men:_First_Class', 'The_Revenant', 'Ex_Machina',
'Avengers:_Age_of_Ultron', 'La_La_Land', 'Midnight_in_Paris',
'Manchester_by_the_Sea', 'Once_Upon_a_Time_in_Hollywood',
'Three_Billboards_Outside_Ebbing', 'Darkest_Hour', 'The_Great_Gatsby',
'Dunkirk', 'Her', 'Captain_America:_The_First_Avenger',
'The_Girls_with_the_Dragon_Tattoo', 'Ready_Player_One', 'Hidden_Figures',
'The_Hateful_Eight', 'Thor', 'Toy_Story_3', 'The_Hunger_Games',
'12_Years_a_Slave', 'Iron_Man_2', 'The_Perks_of_Being_a_Wallflower', 'Joker',
'Les_Miserables', '21_Jump_Street', 'Spiderman:_Far_From_Home', 'Black_Swan',
'Parasite', 'The_Avengers', 'The_Farewell', 'Django_Unchained',
'Now_You_See_Me', 'Avengers:_Endgame', 'Avengers:_Infinity_War',
'Wolf_of_Wall_Street', 'The_Lion_King', 'Gone_Girl',
'Harry_Potter_and_the_Deathly_Hallows:_Part_1', 'The_Social_Network',
'Harry_Potter_and_the_Deathly_Hallows:_Part_2', 'The_Theory_of_Everything',
'Interstellar', 'The_Martian', 'The_Dark_Knight_Rises', 'Shutter_Island',
'Inception']
0
-27.03581500351123
1
-17.5604038243144
2
-16.002362630627825
4
-15.060597317892249
8
-14.501649272824999
16
-14.26378857143746
32
-14.180178075094306
64
-14.170077781591038
128
-14.163960358152185
256
-14.163692439007862
['Shutter_Island', 'The_Social_Network', 'Avengers:_Age_of_Ultron',
'The_Theory_of_Everything', 'The_Farewell', 'Now_You_See_Me', 'The_Martian',
'21_Jump_Street', 'The_Perks_of_Being_a_Wallflower', 'Django_Unchained',
'Midnight_in_Paris', 'The_Hateful_Eight', 'Toy_Story_3', 'Parasite',
'Black_Swan', 'The_Last_Airbender', 'X-Men:_First_Class', 'Us',
'Ready_Player_One', 'The_Girls_with_the_Dragon_Tattoo', 'Pitch_Perfect',
'Frozen', 'Man_of_Steel', 'Hidden_Figures', 'Jurassic_World', 'Fast_Five',
'Rocketman', 'Ex_Machina', 'Chappaquidick', 'American_Hustle', 'The_Help',
'Good_Boys', 'Magic_Mike', 'Darkest_Hour', 'Three_Billboards_Outside_Ebbing',
'The_Revenant', 'Mad_Max:_Fury_Road', 'The_Shape_of_Water', 'Her',
'Fifty_Shades_of_Grey', 'Drive', 'Bridemaids', 'Phantom_Thread', 'Hustlers',
'Prometheus', 'I_Feel_Pretty']
```

```
[ ]:
```