Tingyu Shi A59023729

8.1 (a)

	Movie	Popularity
0	Inception	0.980198
1	The_Dark_Knight_Rises	0.931217
2	The_Social_Network	0.930233
3	Harry_Potter_and_the_Deathly_Hallows:_Part_2	0.920000
4	Interstellar	0.919048
...	...	...
71	Bridemaids	0.553191
72	Magic_Mike	0.508475
73	Fast_&_Furious:_Hobbs_&_Shaw	0.485507
74	I_Feel_Pretty	0.413793
75	Chappaquidick	0.400000

8.1 (b)

$$P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t}\right)$$

$$= \sum_{i=1}^{k} P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t},\ Z=i\right)$$ Marginalization

$$= \sum_{i=1}^{k} P(Z=i)\cdot P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t} \mid Z=i\right)$$ Product Rule

$$= \sum_{i=1}^{k} P(Z=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} \mid Z=i)$$ Conditional Independence.

8.1(c)

$$P\left(Z=i \,\middle|\, \{R_j = r_j^{(t)}\}_{j\in\Omega_t}\right)$$

$$= \frac{P\left(Z=i, \{R_j = r_j^{(t)}\}_{j\in\Omega_t}\right)}{P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t}\right)}$$

$$= \frac{P(Z=i)\cdot P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t} \,\middle|\, Z=i\right)}{P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t}\right)}$$

product Rule

$$= \frac{P(Z=i)\, P\left(\{R_j = r_j^{(t)}\}_{j\in\Omega_t} \,\middle|\, Z=i\right)}{\sum_{i'=1}^{k} P(Z=i') \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} \mid Z=i')}$$

substitute denominator with answer of 8.1(b)

$$= \frac{P(Z=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} \mid Z=i)}{\sum_{i'=1}^{k} P(Z=i') \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} \mid Z=i')}$$

C.I.

8.1(d)

(1) The general equation for root node is

$$P(X_i = x) \leftarrow \frac{1}{T}\sum_t P(X_i = x \mid V_t = v_t)$$

Here, Z is the root node and $R_1 \cdots R_{76}$ are visible nodes.

$$\therefore \quad P(Z=i) = \frac{1}{T}\sum_t^T P(Z=i \mid \{R_j = r_j^{(t)}\}_{j\in\Omega_t})$$

$$= \frac{1}{T}\sum_t^T P_{it}$$

(2) For nodes with parents, the general equation is

$$P(X_i = x \mid Pa_i = \pi) \leftarrow \frac{\sum_t P(X_i = x, Pa_i = \pi \mid V_t = v_t)}{\sum_t P(Pa_i = \pi \mid V_t = v_t)}$$

$$\therefore \quad P(R_j = 1 \mid Z = i) \leftarrow \frac{\sum_{t=1}^T P(R_j = 1, Z = i \mid \{R_j = r_j^{(t)}\}_{j\in\Omega_t})}{\sum_{t=1}^T P(Z = i \mid \{R_j = r_j^{(t)}\}_{j\in\Omega_t})}$$

Numerator:

$$\sum_{t=1}^{T} P(R_j=1, Z=i \mid \{R_j = r_j^{(t)}\}_{j\in\Omega_t})$$

$$= \sum_{\{t \mid j\in\Omega_t\}} P(R_j=1, Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t}) +$$

$$\sum_{\{t \mid j\notin\Omega_t\}} P(R_j=1, Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$\sum_{\{t \mid j\in\Omega_t\}} P(R_j=1, Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$= \sum_{\{t \mid j\in\Omega_t\}} P(Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$\cdot P(R_j=1 \mid Z=i, \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$= \sum_{\{t \mid j\in\Omega_t\}} I(r_j^{(t)}, 1)\ P_{it}$$

$$\sum_{\{t \mid j\notin\Omega_t\}} P(R_j=1, Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$= \sum_{\{t \mid j\notin\Omega_t\}} P(Z=i \mid \{R_k = r_k^{(t)}\}_{k\in\Omega_t}) \cdot P(R_j=1 \mid Z=i, \{R_k = r_k^{(t)}\}_{k\in\Omega_t})$$

$$= \sum_{\{t \mid j\notin\Omega_t\}} P_{it} \cdot P(R_j=1 \mid Z=i)$$ CI.

$\therefore$

$$\text{Numerator} = \sum_{\{t | j \in \Lambda_t\}} I(r_j^{(t)}, 1)\, \rho_{it} + \sum_{\{t | j \notin \Lambda_t\}} \rho_{it}\, P(R_j = 1 \mid Z = i)$$

$$\therefore P(R_j = 1 \mid Z = i) \leftarrow \frac{\sum_{\{t | j \in \Lambda_t\}} I(r_j^{(t)}, 1)\, \rho_{it} + \sum_{\{t | j \notin \Lambda_t\}} \rho_{it}\, P(R_j = 1 \mid Z = i)}{\sum_{t=1}^{T} \rho_{it}}$$

8.1(e)

Iteration	Log-Likelyhood
0	-27.6244
1	-18.4767
2	-16.7949
4	-15.5518
8	-14.9802
16	-14.6801
32	-14.5675
64	-14.5544
128	-14.5525
256	-14.5521

Log-likelyhood increases at each iteration.

8.1(f)

Unseen Movies	Rating Prediction
Shutter_Island	0.999428
Her	0.999134
Midnight_in_Paris	0.998961
Black_Swan	0.998906
21_Jump_Street	0.973956
Three_Billboards_Outside_Ebbing	0.926648
Us	0.919494
Once_Upon_a_Time_in_Hollywood	0.898950
Thor	0.889440
Hustlers	0.888378
Dunkirk	0.883974
Manchester_by_the_Sea	0.871827
The_Perks_of_Being_a_Wallflower	0.862390
Rocketman	0.857618
The_Farewell	0.855130
Good_Boys	0.831695
The_Shape_of_Water	0.828470
Ready_Player_One	0.820728
La_La_Land	0.805954
Pitch_Perfect	0.786311
The_Help	0.773986
Pokemon_Detective_Pikachu	0.754673
Ex_Machina	0.729328
The_Hateful_Eight	0.726279
Drive	0.708248
The_Last_Airbender	0.670534
Bridemaids	0.652035
Chappaquidick	0.651731
I_Feel_Pretty	0.620234

This prediction is not so accurate. But this due to the lack of data.

8.1(g)

HW8_Code

December 6, 2023

1 Homework 8 (Tingyu Shi)

1.0.1 Import Packages

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from progressbar import progressbar
```

1.0.2 Read Data

[2]:
```
# read users
ids = []
fileName = 'hw8_data/hw8_ids.txt'
with open(fileName, 'r') as file:
    for line in file:
        s = line.strip()
        slist = s.split()
        ids.append(slist[0])

# read movies
movies = []
fileName = 'hw8_data/hw8_movies.txt'
with open(fileName, 'r') as file:
    for line in file:
        s = line.strip()
        slist = s.split()
        movies.append(slist[0])

# R init
rinit = []
fileName = 'hw8_data/hw8_probR_init.txt'
with open(fileName, 'r') as file:
    for line in file:
        s = line.strip()
        slist = s.split()
        temp = [float(x) for x in slist]
        rinit.append(temp)
```

```
rinit = np.array(rinit)


# Z init
zinit = []
fileName = 'hw8_data/hw8_probZ_init.txt'
with open(fileName, 'r') as file:
    for line in file:
        s = line.strip()
        slist = s.split()
        temp = [float(x) for x in slist]
        zinit.append(temp)
zinit = np.array(zinit) ; zinit = np.squeeze(zinit)


# read interactions

d = {
    "1": 1,  # recommend
    "0": 0,  # does not recommend
    "?": -1  # haven't seen
}

inters = []
fileName = 'hw8_data/hw8_ratings.txt'
with open(fileName, 'r') as file:
    for line in file:
        s = line.strip()
        slist = s.split()
        temp = [d[x] for x in slist]
        inters.append(temp)
inters = np.array(inters)
```

1.0.3 (a)

[3]:
```
pop_rating = []
for i in range(len(movies)):
    rec = np.sum(inters[:,i] == 1)
    notrec = np.sum(inters[:,i] == 0)
    pop_rating.append(rec / (rec + notrec))

data = {
    "Movie": movies,
    "Popularity": pop_rating
}

df = pd.DataFrame(data)
```

```
df = df.sort_values(by='Popularity', ascending=False)
df.reset_index(drop=True, inplace=True)
df
```

[3]:

```
                                        Movie  Popularity
0                                   Inception    0.980198
1                       The_Dark_Knight_Rises    0.931217
2                          The_Social_Network    0.930233
3   Harry_Potter_and_the_Deathly_Hallows:_Part_2    0.920000
4                                Interstellar    0.919048
..                                        ...         ...
71                                 Bridemaids    0.553191
72                                 Magic_Mike    0.508475
73              Fast_&_Furious:_Hobbs_&_Shaw    0.485507
74                              I_Feel_Pretty    0.413793
75                               Chappaquidick    0.400000

[76 rows x 2 columns]
```

1.0.4 (e)

[4]:

```
def helper1(i, t, rmatrix):
    """
    i: movie type
    t: a user

    calculate P(Rj = rj | Z = i) for j in Sigma t
    """

    res = 1
    interaction = inters[t, :]
    interaction = list( np.squeeze(interaction) )
    for idx, interaction_ in enumerate(interaction):
        if interaction_ == -1:
            continue
        if interaction_ == 1:
            res *= rmatrix[idx, i]
        if interaction_ == 0:
            res *= (1 - rmatrix[idx, i])
    return res
```

[5]:

```
def get_rho_it(i, t, zmatrix, rmatrix):
    """
    i: movie type
    t: a user
    zmatrix: P(Z = i)
    rmatrix: P(Rj = 1 | Z = i)
```

```
    """
    numer = zmatrix[i] * helper1(i, t, rmatrix)

    deno = 0
    for ip in range(len(zmatrix)):
        deno += (zmatrix[ip] * helper1(ip, t, rmatrix))

    return numer / deno
```

[6]:
```
def LL(zmatrix, rmatrix):
    T, _ = inters.shape
    res = 0
    for t in range(T):
        temp = 0
        for i in range(len(zmatrix)):
            temp += (zmatrix[i] * helper1(i, t, rmatrix))
        res += np.log(temp)
    return res / T
```

[7]:
```
def updateZ(zmatrix, rmatrix):
    newzmatrix = np.zeros_like(zmatrix)
    T, _ = inters.shape
    for i in range(len(newzmatrix)):
        temp = 0
        for t in range(T):
            temp += get_rho_it(i, t, zmatrix, rmatrix)
        newzmatrix[i] = temp / T
    return newzmatrix
```

[8]:
```
def helper2(j, i, zmatrix, rmatrix):
    T, _ = inters.shape

    numer = 0
    for t in range(T):
        if inters[t, j] == 1:
            numer += get_rho_it(i, t, zmatrix, rmatrix)
        if inters[t, j] == -1:
            numer += (get_rho_it(i, t, zmatrix, rmatrix) * rmatrix[j, i])

    return numer

def updateR(zmatrix, rmatrix):
    T, _ = inters.shape

    newrmatrix = np.zeros_like(rmatrix)

    # calculate denos in advance
```

```
    denos = []
    for i in range(len(zmatrix)):
        deno = 0
        for t in range(T):
            deno += get_rho_it(i, t, zmatrix, rmatrix)
        denos.append(deno)


    for j in range(rmatrix.shape[0]):
        for i in range(rmatrix.shape[1]):
            newrmatrix[j, i] = helper2(j, i, zmatrix, rmatrix) / denos[i]

    return newrmatrix
```

[9]:

```
recordAt = [0, 1, 2, 4, 8, 16, 32, 64, 128, 256]
records = []

for i in progressbar( range(257) ):
    if i == 0:
        records.append(LL(zinit, rinit))
        continue

    # update
    if i == 1:
        z = updateZ(zinit, rinit) ; r = updateR(zinit, rinit)
        pz = np.copy(z) ; pr = np.copy(r)
    else:
        z = updateZ(pz, pr) ; r = updateR(pz, pr)
        pz = np.copy(z) ; pr = np.copy(r)

    # record
    if i in recordAt:
        records.append(LL(z, r))
```

```
100% (257 of 257) |######################| Elapsed Time: 0:14:30 Time:  0:14:30
```

[10]:

```
newRecords = [round(x, 4) for x in records]

data = {
    "Iteration": recordAt,
    "Log-Likelyhood": newRecords
}
df = pd.DataFrame(data)
df
```

[10]:

	Iteration	Log-Likelyhood
0	0	-27.6244
1	1	-18.4767
2	2	-16.7949
3	4	-15.5518
4	8	-14.9802
5	16	-14.6801
6	32	-14.5675
7	64	-14.5544
8	128	-14.5525
9	256	-14.5521

1.0.5 (f)

[11]:
```
def predict(userIndex, movieIndex):
    res = 0
    for i in range(len(z)):
        res += (get_rho_it(i, userIndex, z, r) * r[movieIndex, i] )
    return res
```

[13]:
```
userIndex = ids.index("A59023729")
myHistory = list(np.squeeze(inters[userIndex]))
probs = []
for movieIndex, history in enumerate(myHistory):
    if history == -1:
        probs.append( (predict(userIndex, movieIndex) , movies[movieIndex]) )
```

[14]:
```
probs.sort(reverse=True)
ratings = [x[0] for x in probs]
unseen_movies = [x[1] for x in probs]

data = {
    "Unseen Movies": unseen_movies,
    "Rating Prediction": ratings
}
df = pd.DataFrame(data)
df
```

[14]:

	Unseen Movies	Rating Prediction
0	Shutter_Island	0.999428
1	Her	0.999134
2	Midnight_in_Paris	0.998961
3	Black_Swan	0.998906
4	21_Jump_Street	0.973956
5	Three_Billboards_Outside_Ebbing	0.926648
6	Us	0.919494
7	Once_Upon_a_Time_in_Hollywood	0.898950

8	Thor	0.889440
9	Hustlers	0.888378
10	Dunkirk	0.883974
11	Manchester_by_the_Sea	0.871827
12	The_Perks_of_Being_a_Wallflower	0.862390
13	Rocketman	0.857618
14	The_Farewell	0.855130
15	Good_Boys	0.831695
16	The_Shape_of_Water	0.828470
17	Ready_Player_One	0.820728
18	La_La_Land	0.805954
19	Pitch_Perfect	0.786311
20	The_Help	0.773986
21	Pokemon_Detective_Pikachu	0.754673
22	Ex_Machina	0.729328
23	The_Hateful_Eight	0.726279
24	Drive	0.708248
25	The_Last_Airbender	0.670534
26	Bridemaids	0.652035
27	Chappaquidick	0.651731
28	I_Feel_Pretty	0.620234

[ ]: