Movie-Recommender-System

November 28, 2017

In [24]:
```
import numpy as np
import pandas as pd

with open('hw8_movieTitles.txt') as f:
    content = f.readlines()
movie_titles = [line.strip() for line in content]

with open('hw8_ratings.txt') as f:
    content = f.readlines()
content = [line.strip() for line in content]
R = np.array([line.split() for line in content])
```

1 8.1 (a) - Sanity Check

In [25]:
```
pop_dict = {}
for i in range(0,R.shape[1]):
    rating = list(R[:,i])
    rating = (rating.count('1')*1.0/(R.shape[0]-rating.count('?')))
    pop_dict[movie_titles[i]] = rating
```

In [26]:
```
labels = ['Movies', 'Mean Popularity Rating']
values = []
for key in sorted(pop_dict, key=pop_dict.__getitem__,reverse=True):
    values.append((key,pop_dict[key]))
pd.DataFrame.from_records(values, columns=labels)
```

Out[26]:

	Movies	Mean Popularity Rating
0	Inception	0.995935
1	The_Dark_Knight_Rises	0.925581
2	Interstellar	0.914414
3	Shutter_Island	0.907801
4	The_Martian	0.901408
5	The_Social_Network	0.894118
6	Now_You_See_Me	0.860606
7	12_Years_a_Slave	0.840909
8	The_Theory_of_Everything	0.840708
9	Gone_Girl	0.839416

10	Harry_Potter_and_the_Deathly_Hallows:_Part_2	0.831897
11	Toy_Story_3	0.810345
12	Black_Swan	0.809917
13	Harry_Potter_and_the_Deathly_Hallows:_Part_1	0.804167
14	Wolf_of_Wall_Street	0.804124
15	The_Avengers	0.800905
16	Midnight_in_Paris	0.797753
17	Frozen	0.790055
18	The_Girls_with_the_Dragon_Tattoo	0.789474
19	Django_Unchained	0.789062
20	Ex_Machina	0.785714
21	Room	0.785714
22	Her	0.779070
23	The_Revenant	0.766990
24	X-Men:_First_Class	0.763736
25	The_Great_Gatsby	0.754839
26	Jurassic_World	0.754011
27	Star_Wars:_The_Force_Awakens	0.748571
28	Les_Miserables	0.745283
29	Captain_America:_The_First_Avenger	0.736041
30	21_Jump_Street	0.735849
31	Avengers:_Age_of_Ultron	0.729885
32	The_Help	0.725000
33	American_Hustle	0.715909
34	The_Perks_of_Being_a_Wallflower	0.695122
35	Iron_Man_2	0.692308
36	The_Hateful_Eight	0.682540
37	Fast_Five	0.678322
38	The_Hunger_Games	0.671296
39	Pitch_Perfect	0.666667
40	Thor	0.664634
41	Drive	0.636364
42	Mad_Max:_Fury_Road	0.633333
43	Man_of_Steel	0.591549
44	World_War_Z	0.590000
45	Bridemaids	0.555556
46	Prometheus	0.543689
47	Magic_Mike	0.524590
48	The_Last_Airbender	0.357143
49	Fifty_Shades_of_Grey	0.330935

2 8.1 (e) - Implementation

In [158]:
```
K = 4 # number of movie-goer types
with open('hw8_probRgivenZ_init.txt') as f:
    content = f.readlines()
content = [line.strip() for line in content]
```

```
        Rz = np.array([list(map(float,d)) \
                       for d in [line.split(' ') for line in content]])
        Z = np.array([0.25]*K)
```

In [159]:
```
        import math

        def compute_likelihood(K,r,Z,Rz,nr_idx):
            likelihood = 0.0
            nr = 0.0
            for i in range(K):
                inner_prod = 1.0
                for j in range(len(r)):
                    if r[j] == '1': inner_prod *= Rz[j][i]
                    if r[j] == '0': inner_prod *= (1-Rz[j][i])
                if i == nr_idx: nr = inner_prod * Z[i] * 1.0
                likelihood += (inner_prod * Z[i] * 1.0)
            return (likelihood,nr)

        def log_likelihood(K,R,Z,Rz):
            ll = 0.0
            for t in range(R.shape[0]):
                ll += math.log(compute_likelihood(K,R[t],Z,Rz,0)[0])
            return ll/R.shape[0]

        def compute_posterior(K,R,Z,Rz):
            P = np.zeros((R.shape[0],K))
            for i in range(K):
                for t in range(R.shape[0]):
                    block = compute_likelihood(K,R[t],Z,Rz,i)
                    P[t][i] = block[1]*1.0/block[0]
            return P

        def getNR(ratings, rz, P_vec):
            Rz_vec = np.array([rz if r == '?' else int(r) for r in ratings])
            return np.dot(P_vec,Rz_vec)

        def em_update(K,R,Z,Rz):
            P = compute_posterior(K,R,Z,Rz)
            for i in range(K):
                sumPi = sum(list(P[:,i]))
                Z[i] = (sumPi*1.0/R.shape[0])
                nr = 0.0
                for j in range(R.shape[1]):
                    Rz[j][i] = getNR(R[:,j], Rz[j][i], P[:,i])*1.0/sumPi
            return Z,Rz
```

In [160]:
```
        table = {}
        p_iter = [0]
```

```
        p_iter.extend([int(math.pow(2,i)) for i in range(7)])
        curr_iter = p_iter[0]
        while curr_iter <= p_iter[len(p_iter)-1]:
            if curr_iter in p_iter:
                table[curr_iter] = log_likelihood(K,R,Z,Rz)
            Z,Rz = em_update(K,R,Z,Rz)
            curr_iter += 1
```

In [163]:
```
labels = ['Iterations', 'Log-Likelihood']
values = []
for key in sorted(table.keys()):
    values.append((key,round(table[key],4)))
pd.DataFrame.from_records(values, columns=labels)
```

Out[163]:

	Iterations	Log-Likelihood
0	0	-23.6819
1	1	-14.3421
2	2	-12.9096
3	4	-12.1506
4	8	-11.8679
5	16	-11.6822
6	32	-11.5655
7	64	-11.5401

3 8.1 (f) - Personal movie recommendations

In [173]:
```
with open('hw8_studentPIDs.txt') as f:
    content = f.readlines()
idx = [line.strip() for line in content].index('A53235626')
unseen = {}
for j in range(len(R[idx])):
    if R[idx][j] == '?':
        hidden_prob = 0.0
        for i in range(K):
            block = compute_likelihood(K,R[idx],Z,Rz,i)
            hidden_prob += Rz[j][i] * (block[1]*1.0/block[0])
        unseen[movie_titles[j]] = hidden_prob
```

The following list actually seems to capture my personal tastes better than the list in 8.1 (a)

In [174]:
```
labels = ['Movies I haven\'t seen', 'Predicted Likability']
values = []
for key in sorted(unseen, key=unseen.__getitem__,reverse=True):
    values.append((key,unseen[key]))
pd.DataFrame.from_records(values, columns=labels)
```

Out[174]:

	Movies I haven't seen	Predicted Likability
0	12_Years_a_Slave	0.937716

1	Black_Swan	0.893647
2	Drive	0.794867
3	Harry_Potter_and_the_Deathly_Hallows:_Part_1	0.734815
4	Star_Wars:_The_Force_Awakens	0.583418
5	World_War_Z	0.499195
6	Man_of_Steel	0.195023