CSE 250a Assignment 8

November 27, 2017

1 8.1 EM algorithm for binary matrix completion

In [76]:
```
import numpy as np
import pandas as pd
```

In [77]:
```
# load files
movieTitles = open('hw8_movieTitles.txt').read().splitlines()
studentPIDs = open('hw8_studentPIDs.txt').read().splitlines()
movieRatings = np.genfromtxt('hw8_ratings.txt', dtype='str')
probZ_init = np.loadtxt('hw8_probZ_init.txt', dtype='float32')
probRgivenZ_init = np.loadtxt('hw8_probRgivenZ_init.txt', dtype='float32')
```

1.0.1 (a) Sanity check

In [93]:
```
meanRatings = []
for j in range(len(movieTitles)):
    ratingsVec = movieRatings[:,j]
    numRecommended = (ratingsVec == '1').sum()
    numSeen = (ratingsVec != '?').sum()
    ratio = numRecommended*1.0/numSeen
    meanRatings.append(ratio)

rankedMovies = [x for x,y in sorted(zip(movieTitles, meanRatings), key=lambda x: x[1])]
for movie in rankedMovies:
    print(movie)
```

```
Fifty_Shades_of_Grey
The_Last_Airbender
Magic_Mike
Prometheus
Bridemaids
World_War_Z
Man_of_Steel
Mad_Max:_Fury_Road
Drive
Thor
Pitch_Perfect
```

```
The_Hunger_Games
Fast_Five
The_Hateful_Eight
Iron_Man_2
The_Perks_of_Being_a_Wallflower
American_Hustle
The_Help
Avengers:_Age_of_Ultron
21_Jump_Street
Captain_America:_The_First_Avenger
Les_Miserables
Star_Wars:_The_Force_Awakens
Jurassic_World
The_Great_Gatsby
X-Men:_First_Class
The_Revenant
Her
Ex_Machina
Room
Django_Unchained
The_Girls_with_the_Dragon_Tattoo
Frozen
Midnight_in_Paris
The_Avengers
Wolf_of_Wall_Street
Harry_Potter_and_the_Deathly_Hallows:_Part_1
Black_Swan
Toy_Story_3
Harry_Potter_and_the_Deathly_Hallows:_Part_2
Gone_Girl
The_Theory_of_Everything
12_Years_a_Slave
Now_You_See_Me
The_Social_Network
The_Martian
Shutter_Island
Interstellar
The_Dark_Knight_Rises
Inception
```

1.0.2 (e) Implementation

```
In [21]: k = 4 #number  of types of movie-goers
         T = len(studentPIDs) #number of students
         ITERS = 64
         ITERS_PRINT = [0,1,2,4,8,16,32,64]
         PID = 'A11381871'
```

```
In [73]: # functions

         ''' E-step
         posterior prob that student corresponds to movie-goer type i
         pass:
         i in k for P(Z=i)
         t for student
         pz and priors for current iteration
         - compute numer and denom separately for efficiency in EM algorithm
         '''
         def estep_numer(i, t, pz, priors):
             j_rec, = np.where(movieRatings[t,:] == '1')
             j_notrec, = np.where(movieRatings[t,:] == '0')
             numer = pz[i]*np.prod(priors[j_rec,i])*np.prod(1-priors[j_notrec,i])
             return numer
         def estep_denom(t, pz, priors):
             denom = 0
             j_rec, = np.where(movieRatings[t,:] == '1')
             j_notrec, = np.where(movieRatings[t,:] == '0')
             for i in range(k):
                 denom += pz[i]*np.prod(priors[j_rec,i])*np.prod(1-priors[j_notrec,i])
             return denom

         ''' M-step
         re-estimate P(Z=i) and P(R_j=1|Z=i)
         pass:
         i in k for P(Z=i)
         j for movie with rating R_j=1
         posteriors and priors for current iteration
         - update P(Z=i) and P(R_j=1|Z=i) separately
         - P(Z=i) and denominator of prior computed in EM_algorithm()'''
         def mstep_prz(i, j, posteriors, priors):
             # sum over students who recommended movie j (I(r_j,1))
             t_seen, = np.where(movieRatings[:,j] == '1')
             numer_seen = np.sum(posteriors[i,t_seen])
             # sum over students who have not seen movie j
             t_unseen, = np.where(movieRatings[:,j] == '?')
             numer_unseen = priors[j,i]*np.sum(posteriors[i,t_unseen])
             return numer_seen+numer_unseen

         '''likelihood of student t's ratings
         pass:
         t for student
         pz and priors for current iteration'''
         def likelihood(t, pz, priors):
             cumsum = 0
             for i in range(k):
                 j_rec, = np.where(movieRatings[t,:] == '1')
```

```
                j_notrec, = np.where(movieRatings[t,:] == '0')
                cumsum += pz[i]*np.prod(priors[j_rec,i])*np.prod(1-priors[j_notrec,i])
            return cumsum

        '''run EM algorithm'''
        def EM_algorithm():
            # initialize CPTs and posteriors with initial values, update each iteration
            pz = np.copy(probZ_init)
            priors = np.copy(probRgivenZ_init)
            posteriors = np.empty([k,T], dtype='float32')
            L = [] #log-likelihoods for each iteration
            for iteration in range(ITERS+1):
                L_iter = 0
                pz_temp = np.empty(4)
                priors_temp = np.empty([50,4])
                # e-step & likelihood calculation
                for t in range(T):
                    L_iter += np.log(likelihood(t, pz, priors))
                    e_denom = estep_denom(t, pz, priors)
                    # e-step - update posteriors
                    for i in range(k):
                        posteriors[i,t] = estep_numer(i,t,pz,priors)/e_denom
                # m-step
                for i in range(k):
                    temp = np.sum(posteriors[i,:])
                    pz_temp[i] = temp/T
                    for j in range(len(movieTitles)):
                        priors_temp[j,i] = mstep_prz(i,j,posteriors,priors)/temp
                L.append(L_iter/T) #append normalized log-likelihood for current iter
                pz = pz_temp #update P(Z=i)
                priors = priors_temp #update priors
                if iteration in ITERS_PRINT:
                    print('iteration: %d \t log-likelihood L = %f' % (iteration, L[iteration]))
            return L, posteriors, pz, priors
```

```
In [43]: log_likelihoods, posteriors_mtx, pz_vec, priors_mtx = EM_algorithm()
```

```
iteration: 0          log-likelihood L = -23.681943
iteration: 1          log-likelihood L = -14.342139
iteration: 2          log-likelihood L = -12.909592
iteration: 4          log-likelihood L = -12.150620
iteration: 8          log-likelihood L = -11.867861
iteration: 16          log-likelihood L = -11.682204
iteration: 32          log-likelihood L = -11.565450
iteration: 64          log-likelihood L = -11.540129
```

1.0.3 (f) Personal movie recommendations

In [94]:
```
my_idx = studentPIDs.index(PID)
my_data = movieRatings[my_idx,:] #my ratings
unseen, = np.where(my_data == '?') #movies I haven't seen
expected_ratings = []

for l in unseen:
    exp_rating = 0
    for i in range(k):
        estep_term = estep_numer(i,my_idx,pz_vec,priors_mtx)/estep_denom(my_idx,pz_vec,
        mstep_term = mstep_prz(i,l,posteriors_mtx, priors_mtx)/np.sum(posteriors_mtx[i,
        exp_rating += estep_term*mstep_term
    expected_ratings.append(exp_rating)
#     print('Movie: %s \t Expected rating: %f' % (movieTitles[l], exp_rating))

pd.DataFrame(list(zip([movieTitles[l] for l in unseen], expected_ratings)), columns=['M
```

Out[94]:

	Movie	Expected rating
0	Shutter_Island	0.838750
1	The_Last_Airbender	0.111544
2	Iron_Man_2	0.299696
3	Fast_Five	0.389798
4	Captain_America:_The_First_Avenger	0.340816
5	X-Men:_First_Class	0.448799
6	Drive	0.765643
7	Midnight_in_Paris	0.821305
8	Prometheus	0.425693
9	The_Perks_of_Being_a_Wallflower	0.645255
10	The_Avengers	0.515682
11	The_Dark_Knight_Rises	0.916737
12	Django_Unchained	0.801189
13	Les_Miserables	0.608445
14	21_Jump_Street	0.640580
15	Magic_Mike	0.278354
16	The_Great_Gatsby	0.615251
17	Frozen	0.855208
18	Now_You_See_Me	0.643127
19	Her	0.765667
20	12_Years_a_Slave	0.935811
21	World_War_Z	0.541566
22	American_Hustle	0.631415
23	Man_of_Steel	0.263388
24	Ex_Machina	0.786971
25	The_Theory_of_Everything	0.711867
26	Star_Wars:_The_Force_Awakens	0.625181
27	Mad_Max:_Fury_Road	0.597657
28	Jurassic_World	0.630645
29	Fifty_Shades_of_Grey	0.191532

30	Avengers:_Age_of_Ultron	0.168732
31	The_Martian	0.930492
32	The_Hateful_Eight	0.769271
33	The_Revenant	0.659578