CSE 150 Homework 5

Pedro Sousa Meireles
A15677282

Fall 2018

1 Maximum Likelihood Estimation

(a) **Complete data**

$$P(B=b|A=a) = \frac{\sum_{t=1}^{T} I(a,a_t)\cdot I(b,b_t)}{\sum_{t=1}^{T} I(a,a_t)}$$

$$P(C=c|A=a,B=b) = \frac{\sum_{t=1}^{T} I(a,a_t)\cdot I(b,b_t)\cdot I(c,c_t)}{\sum_{t=1}^{T} I(a,a_t)\cdot I(b,b_t)}$$

$$P(D=d|A=a,C=c) = \frac{\sum_{t=1}^{T} I(a,a_t)\cdot I(c,c_t)\cdot I(d,d_t)}{\sum_{t=1}^{T} I(a,a_t)\cdot I(d,d_t)}$$

(b) **Posterior probability**

$$\begin{aligned}P(a,c|b,d) &= \frac{P(a,b,c,d)}{P(b,d)}\\ &= \frac{P(a)\cdot P(b|a)\cdot P(c|a,b)\cdot P(d|a,c)}{\sum_{a'}\sum_{c'} P(a=a')\cdot P(b|a=a')\cdot P(c=c'|a=a',b)\cdot P(d|a=a',c=c')}\end{aligned}$$

(c) **Posterior probability**

$$P(a|b,d) = \sum_{c'} P(a,c=c'|b,d)$$

$$P(c|b,d) = \sum_{a'} P(a=a',c|b,d)$$

(d) **Log-likelihood**

$$\begin{aligned}L &= \sum_t \log P(B=b_T, D=d_t)\\ &= \sum_t \log \sum_a \sum_c P(A=a,B=b_T,C=c,D=d_t)\\ &= \sum_t \log \sum_a \sum_c P(A=a)\cdot P(B=b_T|A=a)\cdot P(C=c|A=a,B=b_t)\cdot P(D=d_t|A=a,C=c)\end{aligned}$$

(e) **EM algorithm**

$$\begin{aligned}P(A=a) &\longleftarrow \frac{\widehat{count}(A=a)}{T}\\ &\longleftarrow \frac{\sum_{t=1}^{T} P(A=a|B=b_t,D=d_t)}{T}\\ &\longleftarrow \frac{\sum_{t=1}^{T}\sum_c P(A=a,C=c|B=b_t,D=d_t)}{T}\end{aligned}$$

$$
\begin{aligned}
P(B=b|A=a) &\longleftarrow \frac{\widehat{count}(A=a,B=b)}{\widehat{count}(A=a)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} P(A=a,B=b|B=b_t,D=d_t)}{\widehat{count}(A=a)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} I(b,b_t)\cdot P(A=a|B=b_t,D=d_t)}{\widehat{count}(A=a)} \\
&\longleftarrow \frac{\sum_{t=1}^{T}\sum_c I(b,b_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\widehat{count}(A=a)} \\
&\longleftarrow \frac{\sum_{t=1}^{T}\sum_c I(b,b_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\sum_{t=1}^{T}\sum_c P(A=a,C=c|B=b_t,D=d_t)}
\end{aligned}
$$

$$
\begin{aligned}
P(C=c|A=a,B=b) &\longleftarrow \frac{\widehat{count}(A=a,B=b,C=c)}{\widehat{count}(A=a,B=b)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} P(A=a,B=b,C=c|B=b_t,D=d_t)}{\widehat{count}(A=a,B=b)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} I(b,b_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\widehat{count}(A=a,B=b)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} I(b,b_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\sum_{t=1}^{T}\sum_{c'} I(b,b_t)\cdot P(A=a,C=c'|B=b_t,D=d_t)}
\end{aligned}
$$

$$
\begin{aligned}
P(D=d|A=a,C=c) &\longleftarrow \frac{\widehat{count}(A=a,C=c,D=d)}{\widehat{count}(A=a,C=c)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} P(A=a,C=c,D=d|B=b_t,D=d_t)}{\widehat{count}(A=a,C=c)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} I(d,d_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\widehat{count}(A=a,C=c)} \\
&\longleftarrow \frac{\sum_{t=1}^{T} I(d,d_t)\cdot P(A=a,C=c|B=b_t,D=d_t)}{\sum_{t=1}^{T} P(A=a,C=c|B=b_t,D=d_t)}
\end{aligned}
$$

2 EM algorithm for noisy-OR

(a) **Equivalence of models**

(i) Marginalization over $\overrightarrow{z}$

(ii) Product/chain rule was applied along with conditional independence between every z_i since all paths between different z_i are d-separated due to rule #3.

(iii) $P_B(Y=0|\overrightarrow{Z}=\overrightarrow{z})=0$ for any $z_i=1$, so the only remaining term is for $\overrightarrow{Z}=0$.

(iv) $P_B(Z_i=0|X_i=1)=1-p_i$ and $P_B(Z_i=0|X_i=0)=1$. Then $P_B(Z_i=0|X_i=x_i)=(1-p_i)^{x_i}$

(b) **EM Implementation: Per-iteration statistics**

iteration	number of mistakes M	log conditional likelihood L
0	175	-0.9581
1	56	-0.4959
2	43	-0.4082
4	42	-0.3646
8	44	-0.3475
16	40	-0.1146
32	37	-0.3226
64	37	-0.3148
128	36	-0.3112
256	36	-0.3102
512	36	-0.3100

(c) **EM Implementation: Estimated values for p_i**

i	p_i
1	7.952606636558421e-05
2	0.00481741209369718
3	2.5702471621528087e-11
4	0.26533320243361963
5	1.4919335171144578e-05
6	0.009464229795569459
7	0.24030074435506368
8	0.11345165109907844
9	0.0001434658634790401
10	0.5234804065754611
11	0.4072853322527083
12	9.074568784373926e-08
13	0.6157947609276805
14	5.954608022848239e-06
15	0.04490171238688918
16	0.5899773388133327
17	0.9999999999999938
18	0.9999999821990931
19	4.154500914170821e-09
20	0.4629914874582514
21	0.3531984001127537
22	0.5248644134908856
23	0.19475859952835034

(d) **EM Implementation: Source code**

```
In [99]: import math
```

```
In [100]: X = []
          for l in open("hw5_noisyOr_x.txt"):
              numbers = l.split()
              numbers = [int(s) for s in numbers]
              X.append(numbers)
```

```
In [101]: y = []
          for l in open("hw5_noisyOr_y.txt"):
              #numbers = l.split()
              #numbers = [int(s) for s in numbers]
              y.append(int(l))
```

In [102]:
```
p = []
for i in range(0, 23):
    p.append(0.05)
```

In [103]:
```
def updatePi():
    newP = []
    for i in range(0, 23):
        pi = 0
        Ti = 0
        for t in range(0, len(X)):
            prod = 1
            for j in range(0, len(X[0])):
                prod *= (1-p[j])**X[t][j]
            pi += y[t]*X[t][i]*p[i]/(1-prod)
            Ti += X[t][i]
        pi = pi/Ti
        newP.append(pi)
    return newP
```

In [104]:
```
def computePYX():
    PYX = []
    for i in range(0,len(X)):
        pyx = 1
        for j in range(0, len(X[0])):
            pyx *= (1-p[j])**X[i][j]
        PYX.append(1-pyx)
    return PYX
```

In [105]:
```
def loglikelihood(probabilities):
    ll = 0
    for i in range(len(y)):
        if y[i] == 0:
            ll += math.log(1-probabilities[i])
        else:
            ll += math.log(probabilities[i])
    ll /= len(y)
    return ll
```

In [106]:
```
iterations = []

probabilities = computePYX()
ll = loglikelihood(probabilities)

errors = 0
for i in range(0, len(y)):
    if (probabilities[i] >= 0.5 and y[i] == 0)
    or (probabilities[i] < 0.5 and y[i] == 1):
        errors +=1

iterations.append((errors, ll))

for i in range(0,512):
    p = updatePi();
    probabilities = computePYX()
    ll = loglikelihood(probabilities)
```

```
        errors = 0
        for i in range(0, len(y)):
            if (probabilities[i] >= 0.5 and y[i] == 0)
            or (probabilities[i] < 0.5 and y[i] == 1):
                errors +=1

        iterations.append((errors, ll))
```

In [120]:
```
table = []
table.append((0, iterations[0][0], iterations[0][1]))

i = 1
while i <= 512:
    table.append((i, iterations[i][0], iterations[i][1]))
    i *= 2
```

In [121]:
```
table
```

Out[121]:
```
[(0, 175, -0.9580854082157914),
 (1, 56, -0.49591639407753635),
 (2, 43, -0.40822081705839114),
 (4, 42, -0.3646149825001877),
 (8, 44, -0.34750061620878253),
 (16, 40, -0.33461704895854844),
 (32, 37, -0.3225814031674978),
 (64, 37, -0.3148266983628557),
 (128, 36, -0.3111558472151897),
 (256, 36, -0.3101613534740759),
 (512, 36, -0.30999030298497576)]
```

In [122]:
```
p
```

Out[122]:
```
[7.952606636558421e-05,
 0.004817412093697185,
 2.5702471621528087e-11,
 0.26533320243361963,
 1.4919335171144578e-05,
 0.009464229795569459,
 0.24030074435506368,
 0.11345165109907844,
 0.0001434658634790401,
 0.5234804065754611,
 0.4072853322527083,
 9.074568784373926e-08,
 0.6157947609276805,
 5.954608022848239e-06,
 0.04490171238688918,
 0.5899773388133327,
 0.9999999999999938,
 0.9999999821990931,
 4.154500914170821e-09,
 0.4629914874582514,
 0.3531984001127537,
 0.5248644134908856,
 0.19475859952835034]
```

3 EM algorithm for binary matrix completion

(a) **Sanity check**

['The_Last_Airbender', 'Batman_v_Superman:_Dawn_of_Justice', 'Justice_League', 'Suicide_Squad', 'It', 'Terminator_Genisys', 'World_War_Z', 'The_Shape_of_Water', 'Venom', 'Man_of_Steel', 'Tron', 'Blade_Runner_2049', 'The_Lego_Movie', 'Star_Wars:_The_Phantom_Menace', 'Jurassic_World', 'The_Greatest_Showman', 'The_Hunger_Games', 'Terminator_2', 'Oceans_8', 'Star_Wars:_The_Last_Jedi', 'La_La_Land', 'Mad_Max:_Fury_Road', 'Get_Out', 'Furious_7', 'Jumanji:_Welcome_to_the_Jungle', '2001:_A_Space_Odyssey', 'Wonder_Woman', 'Frozen', 'Star_Trek_Beyond', 'Captain_America:_Civil_War', 'Guardians_of_the_Galaxy_Vol._2', 'Ex_Machina', 'Harry_Potter_and_the_Deathly_Hallows:_Part_2', 'Ant-Man_and_the_Wasp', 'Fantastic_Beasts_and_Where_To_Find_Them', 'Rogue_One', 'Logan', 'Zootopia', 'The_Lord_of_the_Rings:_The_Fellowship_of_the_Ring', 'Thor:_Ragnarok', 'Deadpool_2', 'The_Imitation_Game', 'Guardians_of_the_Galaxy', 'Iron_Man_3', 'Black_Panther', 'The_Martian', 'The_Wolf_of_Wall_Street', 'The_Dark_Knight', 'Jurassic_Park_(1993)', 'Avengers:_Infinity_War', 'Mission:_Impossible_-_Fallout', 'Coco', 'Moana', 'Interstellar', 'The_Matrix', 'The_Avengers', 'WALL-E', 'Inception', 'Solo', 'Doctor_Strange']

(b) **Likelihood**

$$\begin{aligned}
P(\{R_j = r_j^{(t)}\}_{j\in\Omega_t}) &= \sum_{i=1}^{k} P(\{R_j = r_j^{(t)}\}_{j\in\Omega_t}, Y = i) && \text{(Marginalization)}\\
&= \sum_{i=1}^{k} P(Y=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = i, R_1 = r_1^{(t)}, ..., R_{j-1} = r_{j-1}^{(t)}) && \text{(Product rule)}\\
&= \sum_{i=1}^{k} P(Y=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = i) && \text{(Cond. Ind. in Bayes' Net)}
\end{aligned}$$

(c) **E-step**

$$\begin{aligned}
P(Y = i | \{R_j = r_j^{(t)}\}_{j\in\Omega_t}) &= \frac{P(Y = i, \{R_j = r_j^{(t)}\}_{j\in\Omega_t})}{P(\{R_j = r_j^{(t)}\}_{j\in\Omega_t})} && \text{(Product rule)}\\
&= \frac{P(Y=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = i, R_1 = r_1^{(t)}, ..., R_{j-1} = r_{j-1}^{(t)})}{P(\{R_j = r_j^{(t)}\}_{j\in\Omega_t})} && \text{(Product rule)}\\
&= \frac{P(Y=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = i)}{P(\{R_j = r_j^{(t)}\}_{j\in\Omega_t})} && \text{(Cond. Ind. in Bayes' Net)}\\
&= \frac{P(Y=i) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = i)}{\sum_{y=1}^{k} P(Y=y) \prod_{j\in\Omega_t} P(R_j = r_j^{(t)} | Y = y)} && \text{(Replacing denominator)}
\end{aligned}$$

(d) **M-step**

$$
\begin{aligned}
P(Y=i) &\longleftarrow \sum_{t=1}^{T} P(Y=i,\{R_j=r_j^{(t)}\}_{j\in\Omega_t}) && \text{(Marginalization over samples)}\\
&\longleftarrow \sum_{t=1}^{T} P(Y=i|\{R_j=r_j^{(t)}\}_{j\in\Omega_t})\cdot P(\{R_j=r_j^{(t)}\}_{j\in\Omega_t}) && \text{(Product rule)}\\
&\longleftarrow \frac{1}{T}\sum_{t=1}^{T} P(Y=i|\{R_j=r_j^{(t)}\}_{j\in\Omega_t}) && (P(\{R_j=r_j^{(t)}\}_{j\in\Omega_t})=\frac{1}{T})\\
&\longleftarrow \frac{1}{T}\sum_{t=1}^{T} p_{it}
\end{aligned}
$$

$$
\begin{aligned}
P(R_j=1|Y=1) &\longleftarrow \frac{P(Y=i|R_j=1)\cdot P(R_j=1)}{P(Y=1)}\\
&\longleftarrow \frac{\sum_t P(Y=i,\{R_j=r_j^{(t)}\}|R_j=1)\cdot P(R_j=1)}{P(Y=1)}\\
&\longleftarrow \frac{\sum_t P(Y=i,R_j=1|\{R_j=r_j^{(t)}\})\cdot P(\{R_j=r_j^{(t)}\})}{P(Y=1)}\\
&\longleftarrow \frac{\frac{1}{T}(\sum_{t:j\in\Omega_t} P(Y=i|\{R_j=r_j^{(t)}\})\cdot I(r_j^{(t)},1)+\sum_{t:j\notin\Omega_t} P(Y=i|\{R_j=r_j^{(t)}\})\cdot P(R_j=1|Y=i))}{P(Y=1)}\\
&\longleftarrow \frac{\frac{1}{T}(\sum_{t:j\in\Omega_t} p_{it}\cdot I(r_j^{(t)},1)+\sum_{t:j\notin\Omega_t} p_{it}\cdot P(R_j=1|Y=i))}{P(Y=1)}\\
&\longleftarrow \frac{\frac{1}{T}(\sum_{t:j\in\Omega_t} p_{it}\cdot I(r_j^{(t)},1)+\sum_{t:j\notin\Omega_t} p_{it}\cdot P(R_j=1|Y=i))}{\frac{1}{T}\sum_{t=1}^{T} p_{it}}\\
&\longleftarrow \frac{\sum_{t:j\in\Omega_t} p_{it}\cdot I(r_j^{(t)},1)+\sum_{t:j\notin\Omega_t} p_{it}\cdot P(R_j=1|Y=i)}{\sum_{t=1}^{T} p_{it}}
\end{aligned}
$$

(e) **Implementation**

iteration	**log-likelihood** L
0	-27.9848
1	-15.5730
2	-13.6614
4	-12.5566
8	-12.1332
16	-12.0195
32	-12.0040
64	-12.0005
128	-12.0000

(f) **Personal categorization** Given my personal ratings, the i that maximizes $P(Y=i|\mathit{my\ ratings})$ is $i=1$, which gives a probability of 1.

(g) **Personal movie recommendations**

unseen movies	$P(R_j = 1 \mid my\ ratings)$
Black Panther	0.9436
The Last Airbender	0.2642
2001: A Space Odyssey	0.9494
Terminator Genisys	0.5840
Terminator 2	0.8577
Coco	0.9357
Ant-Man and the Wasp	0.8815
Venom	0.6962
Oceans 8	0.9449
The Lego Movie	0.7885
Jumanji: Welcome to the Jungle	0.8222
Solo	1.0000
Furious 7	0.8332

This list reflects my personal tastes better than the list in item (a).

(h) **Source code**

3.0.1 Question 3

(a)

```
In [138]: X = []
          for l in open("hw5_movieRatings_fa18.txt"):
              v = []
              for x in l.split():
                  if x != '?':
                      v.append(int(x))
                  else:
                      v.append(x)
              X.append(v)
```

```
In [132]: titles = []
          for l in open("hw5_movieTitles_fa18.txt"):
              titles.append(l.split()[0])
```

```
In [154]: moviesAvg = []
          for i in range(0, len(titles)):
              moviesAvg.append([0,0,0])

          for i in range(0, len(X)):
              for j in range(0, len(titles)):
                  if X[i][j] == 1:
                      moviesAvg[j][0] += 1
                  if X[i][j] != '?':
                      moviesAvg[j][1] += 1

          for i in range(0, len(titles)):
              moviesAvg[i][2] = moviesAvg[i][0]/moviesAvg[i][1]

          moviesAvg = list(zip(titles, moviesAvg))
```

```
In [160]: moviesAvg = sorted(moviesAvg, key=lambda x:x[1][2])
```

```
In [164]: [v[0] for v in moviesAvg]

Out[164]: ['The_Last_Airbender',
           'Batman_v_Superman:_Dawn_of_Justice',
           'Justice_League',
           'Suicide_Squad',
           'It',
           'Terminator_Genisys',
           'World_War_Z',
           'The_Shape_of_Water',
           'Venom',
           'Man_of_Steel',
           'Tron',
           'Blade_Runner_2049',
           'The_Lego_Movie',
           'Star_Wars:_The_Phantom_Menace',
           'Jurassic_World',
           'The_Greatest_Showman',
           'The_Hunger_Games',
           'Terminator_2',
           'Oceans_8',
           'Star_Wars:_The_Last_Jedi',
           'La_La_Land',
           'Mad_Max:_Fury_Road',
           'Get_Out',
           'Furious_7',
           'Jumanji:_Welcome_to_the_Jungle',
           '2001:_A_Space_Odyssey',
           'Wonder_Woman',
           'Frozen',
           'Star_Trek_Beyond',
           'Captain_America:_Civil_War',
           'Guardians_of_the_Galaxy_Vol._2',
           'Ex_Machina',
           'Harry_Potter_and_the_Deathly_Hallows:_Part_2',
           'Ant-Man_and_the_Wasp',
           'Fantastic_Beasts_and_Where_To_Find_Them',
           'Rogue_One',
           'Logan',
           'Zootopia',
           'The_Lord_of_the_Rings:_The_Fellowship_of_the_Ring',
           'Thor:_Ragnarok',
           'Deadpool_2',
           'The_Imitation_Game',
           'Guardians_of_the_Galaxy',
           'Iron_Man_3',
           'Black_Panther',
           'The_Martian',
           'The_Wolf_of_Wall_Street',
           'The_Dark_Knight',
           'Jurassic_Park_(1993)',
           'Avengers:_Infinity_War',
           'Mission:_Impossible_-_Fallout',
           'Coco',
```

```
 'Moana',
 'Interstellar',
 'The_Matrix',
 'The_Avengers',
 'WALL-E',
 'Inception',
 'Solo',
 'Doctor_Strange']
```

(e)

```
In [170]: pY = []
          for l in open("hw5_probType_init_fa18.txt"):
              pY.append(float(l.split()[0]))

          pRY = []
          for l in open("hw5_probRatingGivenType_init_fa18.txt"):
              probs = [float(x) for x in l.split()]
              pRY.append(probs)

In [189]: def computePit():
              vPit = []
              for t in range(0, len(X)):
                  vPt = []
                  for i in range(0, len(pY)):
                      prod = pY[i]
                      for j in range(0, len(pRY)):
                          if X[t][j] == 1:
                              prod *= pRY[j][i]
                          elif X[t][j] == 0:
                              prod *= (1-pRY[j][i])
                      vPt.append(prod)
                  vPt = [x/sum(vPt) for x in vPt]
                  vPit.append(vPt)
              return list(map(list, zip(*vPit)))

In [208]: def updatePY(Pit):
              newPY = []
              for i in range(0, len(pY)):
                  sum = 0
                  for t in range(0, len(X)):
                      sum += Pit[i][t]
                  sum /= len(X)
                  newPY.append(sum)
              return newPY

In [203]: def updatePRY(Pit):
              newPRY = []
              for j in range(0, len(pRY)):
                  newPy = []
                  for i in range(0, len(pY)):
                      sumT = 0
                      sumNotT = 0
                      for t in range(0, len(X)):
```

```
                    if X[t][j] == 1:
                        sumT += Pit[i][t]
                    elif X[t][j] == '?':
                        sumNotT += Pit[i][t]*pRY[j][i]
                newPy.append((sumT + sumNotT)/sum(Pit[i]))
            newPRY.append(newPy)
        return newPRY
```

In [214]:
```
def computePR(Pit, t):
    sum = 0
    for i in range(0, len(pY)):
        prod = pY[i]
        for j in range(0, len(pRY)):
            if X[t][j] == 1:
                prod *= pRY[j][i]
            elif X[t][j] == 0:
                prod *= (1-pRY[j][i])
        sum += prod
    return sum
```

In [215]:
```
def loglikelihood(Pit):
    ll = 0
    for t in range(0, len(X)):
        ll += math.log(computePR(Pit, t))
    ll /= len(X)
    return ll
```

In [220]:
```
loglikelihood(computePit())
```

Out[220]: -27.98479373624261

In [221]:
```
iterations = []

pit = computePit()
ll = loglikelihood(pit)

iterations.append([0, ll])

for i in range(0,128):
    pY = updatePY(pit)
    pRY = updatePRY(pit)
    pit = computePit()
    ll = loglikelihood(pit)

    iterations.append([i+1, ll])
```

In [223]:
```
table = []
table.append(iterations[0])
i=1
while i <= 128:
    table.append(iterations[i])
    i *= 2
```

In [224]:
```
table
```

```
Out[224]: [[0, -27.98479373624261],
           [1, -15.572996571571787],
           [2, -13.66141481085908],
           [4, -12.556587236861157],
           [8, -12.13316136441514],
           [16, -12.019531087089582],
           [32, -12.004032227091116],
           [64, -12.000469747436933],
           [128, -12.00004068559657]]
```

(f)

```
In [238]: myRatings = ['?', 1, '?', 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, '?', 1, '?',
  '?', 1, '?', '?', '?', '?', 1, '?', 1, 1, 1, '?', 1, 0, 1, 1, '?', 1, 0, 1, 1, '?']

In [239]: X.index(myRatings)

Out[239]: 111

In [240]: myProbs = [p[111] for p in pit]

In [241]: myProbs

Out[241]: [1.0, 5.151841717383405e-86, -3.573574433350554e-63, 0.0]
```

(g)

```
In [244]: unseenMovies = []
          for j in range(0, len(myRatings)):
              sum = 0
              if myRatings[j] == '?':
                  for i in range(0, len(pY)):
                      sum += pit[i][111]*pRY[j][i]
                  unseenMovies.append([titles[j], sum])

In [245]: unseenMovies

Out[245]: [['Black_Panther', 0.9435799253823884],
           ['The_Last_Airbender', 0.2641657488701842],
           ['2001:_A_Space_Odyssey', 0.9494494359912142],
           ['Terminator_Genisys', 0.5840336593899693],
           ['Terminator_2', 0.8577013829345648],
           ['Coco', 0.935731301447267],
           ['Ant-Man_and_the_Wasp', 0.8815382276129842],
           ['Venom', 0.6962129331476755],
           ['Oceans_8', 0.944858959209109],
           ['The_Lego_Movie', 0.7885033530939412],
           ['Jumanji:_Welcome_to_the_Jungle', 0.8222240618014134],
           ['Solo', 0.9999999999999997],
           ['Furious_7', 0.8331588178649978]]
```