hw5ans

May 27, 2021

1 Maximum Likelihood for Bayesian Networks

[27]:
```
import numpy as np
import pyGM as gm
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
data = np.genfromtxt('RiskFactorData.csv', delimiter=",",names=True)
data
```

[27]:
```
array([(6., 2., 2., 4., 2., 2., 1., 2., 2., 3.),
       (2., 2., 1., 3., 1., 2., 1., 2., 2., 3.),
       (6., 1., 2., 3., 1., 2., 3., 2., 2., 3.), …,
       (2., 2., 2., 3., 1., 2., 3., 2., 2., 3.),
       (1., 2., 2., 4., 1., 2., 1., 2., 2., 3.),
       (1., 2., 2., 2., 1., 2., 3., 2., 2., 3.)],
      dtype=[('income', '<f8'), ('smoke', '<f8'), ('cholesterol', '<f8'),
('bmi', '<f8'), ('exercise', '<f8'), ('attack', '<f8'), ('bp', '<f8'),
('angina', '<f8'), ('stroke', '<f8'), ('diabetes', '<f8')])
```

[28]:
```
data_int = np.array([list(xj) for xj in data], dtype=int)-1
```

[29]:
```
nTrain = int(.75*len(data_int))
train = data_int[:nTrain]
valid = data_int[nTrain:]
```

(b)

[108]:
```
rates = [8,2,2,4,2,2,4,2,2,4]
variables = [gm.Var(i,rates[i]) for i in range(10)]
elements = [train[:,i] for i in range(10)]

pI = gm.Factor([variables[0]],0)
pEgI = gm.Factor([variables[0],variables[4]],0)
pSmokegI = gm.Factor([variables[0],variables[1]],0)
pBMIgIE = gm.Factor([variables[0],variables[3],variables[4]],0)
pBPgIESmoke = gm.Factor([variables[0],variables[1],variables[4],variables[6]],0)
```

```
pCgIESmoke = gm.Factor([variables[0],variables[1],variables[2],variables[4]],0)
pDgBMI = gm.Factor([variables[3],variables[9]],0)
pSgBMIBPC = gm.Factor([variables[2],variables[3],variables[6],variables[8]],0)
pAttackgBMIBPC = gm.
 ↪Factor([variables[2],variables[3],variables[5],variables[6]],0)
pAnginagBMIBPC = gm.
 ↪Factor([variables[2],variables[3],variables[6],variables[7]],0)

'''
pI = gm.Factor([income],0)
pEgI = gm.Factor([income,exercise],0)
pSmokegI = gm.Factor([income,smoke],0)
pBMIgIE = gm.Factor([income,bmi,exercise],0)
pBPgIESmoke = gm.Factor([income,smoke,exercise,bp],0)
pCgIESmoke = gm.Factor([income,smoke,cholesterol,exercise],0)
pDgBMI = gm.Factor([bmi,diabetes],0)
pSgBMIBPC = gm.Factor([cholesterol,bmi,bp,stroke],0)
pAttackgBMIBPC = gm.Factor([cholesterol,bmi,attack,bp],0)
pAnginagBMIBPC = gm.Factor([cholesterol,bmi,bp,angina],0)
'''

for i in range(len(train)):
    pI[int(elements[0][i])]+=1
    pEgI[int(elements[0][i]),int(elements[4][i])]+=1
    pSmokegI[int(elements[0][i]),int(elements[1][i])]+=1
    pDgBMI[int(elements[3][i]),int(elements[9][i])]+=1
    pBMIgIE[int(elements[0][i]),int(elements[3][i]),int(elements[4][i])]+=1
   ␣
 ↪pBPgIESmoke[int(elements[0][i]),int(elements[1][i]),int(elements[4][i]),int(elements[6][i])
   ␣
 ↪pCgIESmoke[int(elements[0][i]),int(elements[1][i]),int(elements[2][i]),int(elements[4][i])]
   ␣
 ↪pSgBMIBPC[int(elements[2][i]),int(elements[3][i]),int(elements[6][i]),int(elements[8][i])]+
   ␣
 ↪pAttackgBMIBPC[int(elements[2][i]),int(elements[3][i]),int(elements[5][i]),int(elements[6][
   ␣
 ↪pAnginagBMIBPC[int(elements[2][i]),int(elements[3][i]),int(elements[6][i]),int(elements[7][
```

[109]:

```
pI/=len(train)
pEgI/=pEgI.sum([variables[4]])
pSmokegI/=pSmokegI.sum([variables[1]])
pDgBMI/=pDgBMI.sum([variables[9]])
pBMIgIE/=pBMIgIE.sum([variables[3]])
pBPgIESmoke/=pBPgIESmoke.sum([variables[6]])
pCgIESmoke/=pCgIESmoke.sum([variables[2]])
pSgBMIBPC/=pSgBMIBPC.sum([variables[8]])
pAttackgBMIBPC/=pAttackgBMIBPC.sum([variables[5]])
```

```
pAnginagBMIBPC/=pAnginagBMIBPC.sum([variables[7]])
factors =␣
↪[pI,pEgI,pSmokegI,pDgBMI,pBMIgIE,pBPgIESmoke,pCgIESmoke,pSgBMIBPC,pAttackgBMIBPC,pAnginagBM
```

[110]:
```
model = gm.GraphModel(factors)
```

[111]:
```
dispI = pd.DataFrame(data=pI.table)
dispI
```

[111]:

	0
0	0.048626
1	0.058928
2	0.073241
3	0.092637
4	0.115643
5	0.150993
6	0.164419
7	0.295514

[132]:
```
dispEgI = pd.DataFrame(data=(pEgI.table))
dispEgI
```

[132]:

	0	1
0	0.624463	0.375537
1	0.609464	0.390536
2	0.633656	0.366344
3	0.663786	0.336214
4	0.702432	0.297568
5	0.749060	0.250940
6	0.791691	0.208309
7	0.851669	0.148331

The total number of probabilities we need to estimate for this network is $7+8+8+48+96+32+12+32+32+32 = 307$.

On the other side, the total number of probabilities in the full joint distribution is $8 \times 2 \times 2 \times 4 \times 4 \times 2 \times 4 \times 2 \times 2 \times 2 - 1 = 32767$.

(c)

[113]:
```
np.mean([model.logValue(x) for x in train])
```

[113]: -6.806920101456448

(d)

[114]:
```
np.mean([model.logValue(x) for x in valid])
```

```
C:\Users\74765\anaconda3\lib\site-packages\pyGM\graphmodel.py:181:
RuntimeWarning: divide by zero encountered in log
```

```
    else:          return sum( [ np.log(f.valueMap(x)) for f in factors ] )
```

[114]: -inf

(e)

```
[118]: pI2 = gm.Factor([variables[0]],1)
pEgI2 = gm.Factor([variables[0],variables[4]],1)
pSmokegI2 = gm.Factor([variables[0],variables[1]],1)
pBMIgIE2 = gm.Factor([variables[0],variables[3],variables[4]],1)
pBPgIESmoke2 = gm.
 ↪Factor([variables[0],variables[1],variables[4],variables[6]],1)
pCgIESmoke2 = gm.Factor([variables[0],variables[1],variables[2],variables[4]],1)
pDgBMI2 = gm.Factor([variables[3],variables[9]],1)
pSgBMIBPC2 = gm.Factor([variables[2],variables[3],variables[6],variables[8]],1)
pAttackgBMIBPC2 = gm.
 ↪Factor([variables[2],variables[3],variables[5],variables[6]],1)
pAnginagBMIBPC2 = gm.
 ↪Factor([variables[2],variables[3],variables[6],variables[7]],1)

for i in range(len(train)):
    pI2[int(elements[0][i])]+=1
    pEgI2[int(elements[0][i]),int(elements[4][i])]+=1
    pSmokegI2[int(elements[0][i]),int(elements[1][i])]+=1
    pDgBMI2[int(elements[3][i]),int(elements[9][i])]+=1
    pBMIgIE2[int(elements[0][i]),int(elements[3][i]),int(elements[4][i])]+=1
    ␣
 ↪pBPgIESmoke2[int(elements[0][i]),int(elements[1][i]),int(elements[4][i]),int(elements[6][i]
    ␣
 ↪pCgIESmoke2[int(elements[0][i]),int(elements[1][i]),int(elements[2][i]),int(elements[4][i])
    ␣
 ↪pSgBMIBPC2[int(elements[2][i]),int(elements[3][i]),int(elements[6][i]),int(elements[8][i])]
    ␣
 ↪pAttackgBMIBPC2[int(elements[2][i]),int(elements[3][i]),int(elements[5][i]),int(elements[6]
    ␣
 ↪pAnginagBMIBPC2[int(elements[2][i]),int(elements[3][i]),int(elements[6][i]),int(elements[7]

pI2/=len(train)
pEgI2/=pEgI2.sum([variables[4]])
pSmokegI2/=pSmokegI2.sum([variables[1]])
pDgBMI2/=pDgBMI2.sum([variables[9]])
pBMIgIE2/=pBMIgIE2.sum([variables[3]])
pBPgIESmoke2/=pBPgIESmoke2.sum([variables[6]])
pCgIESmoke2/=pCgIESmoke2.sum([variables[2]])
pSgBMIBPC2/=pSgBMIBPC2.sum([variables[8]])
pAttackgBMIBPC2/=pAttackgBMIBPC2.sum([variables[5]])
pAnginagBMIBPC2/=pAnginagBMIBPC2.sum([variables[7]])
```

[119]:
```
factors2 =␣
↪[pI2,pEgI2,pSmokegI2,pDgBMI2,pBMIgIE2,pBPgIESmoke2,pCgIESmoke2,pSgBMIBPC2,pAttackgBMIBPC2,p
model2 = gm.GraphModel(factors2)
```

[120]:
```
np.mean([model2.logValue(x) for x in train])
```

[120]: -6.806922505441412

[121]:
```
np.mean([model2.logValue(x) for x in valid])
```

[121]: -6.852253937548911

2 Learning Graph Structure

(a)

[122]:
```
phat = {}
for i in model2.X:          # estimate single-variable probabilities
    phat[i] = gm.Factor([i],1)
    for xs in train: phat[i][xs[i]] += 1.0
    phat[i] /= len(train)

for i in model2.X:          # estimate pairwise probabilities
    for j in model2.X:
        if j<=i: continue
        phat[i,j] = gm.Factor([i,j],1)
        for xs in train: phat[i,j][xs[i],xs[j]] += 1.0
        phat[i,j] /= len(train)
```

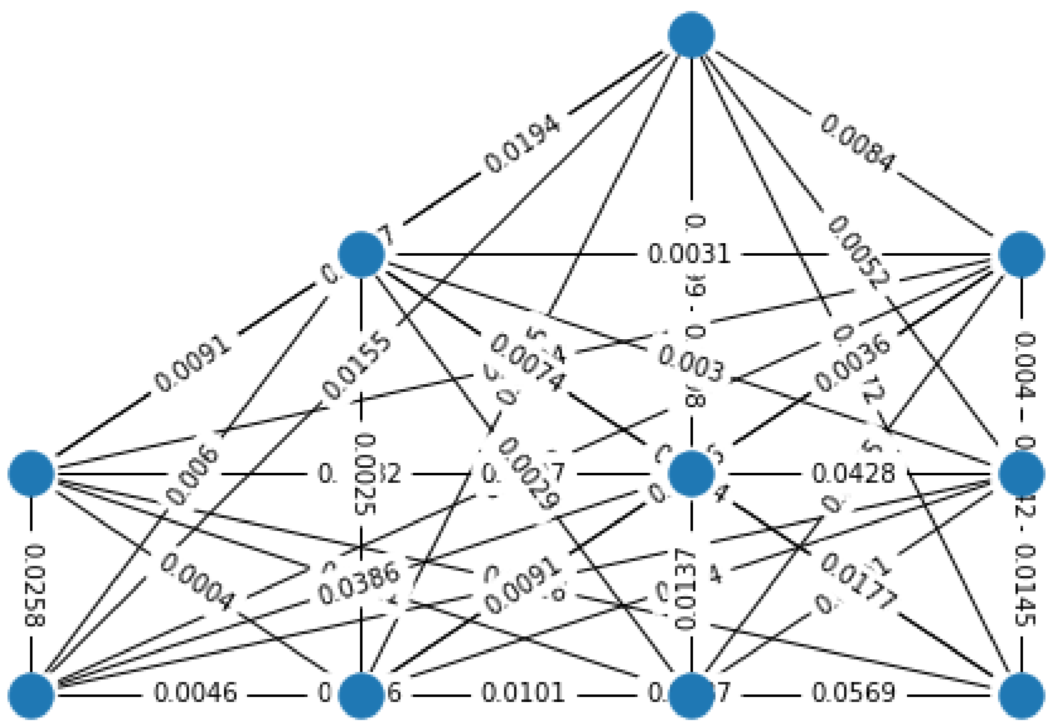
[123]:
```
wts = np.zeros((10,10))
for i in model2.X:          # estimate pairwise probabilities
    for j in model2.X:
        if j<=i: continue   # estimate (empirical) mutual information:
        wts[i,j] = (phat[i,j] * (phat[i,j]/phat[i]/phat[j]).log() ).sum()

np.set_printoptions(precision=4, suppress=True)
print(np.round(wts,4))
```

```
[[0.     0.0084 0.0052 0.0067 0.0194 0.0098 0.0199 0.0072 0.0095 0.0155]
 [0.     0.     0.004  0.0004 0.0031 0.0055 0.0036 0.0042 0.0016 0.0013]
 [0.     0.     0.     0.0097 0.003  0.0101 0.0428 0.0145 0.004  0.0196]
 [0.     0.     0.     0.     0.0091 0.0011 0.0282 0.0016 0.0004 0.0258]
 [0.     0.     0.     0.     0.     0.0029 0.0074 0.0024 0.0025 0.006 ]
 [0.     0.     0.     0.     0.     0.     0.0137 0.0569 0.0115 0.0096]
 [0.     0.     0.     0.     0.     0.     0.     0.0177 0.0091 0.0386]
 [0.     0.     0.     0.     0.     0.     0.     0.     0.0087 0.0101]
 [0.     0.     0.     0.     0.     0.     0.     0.     0.     0.0046]
 [0.     0.     0.     0.     0.     0.     0.     0.     0.     0.    ]]
```

(b)

[124]:

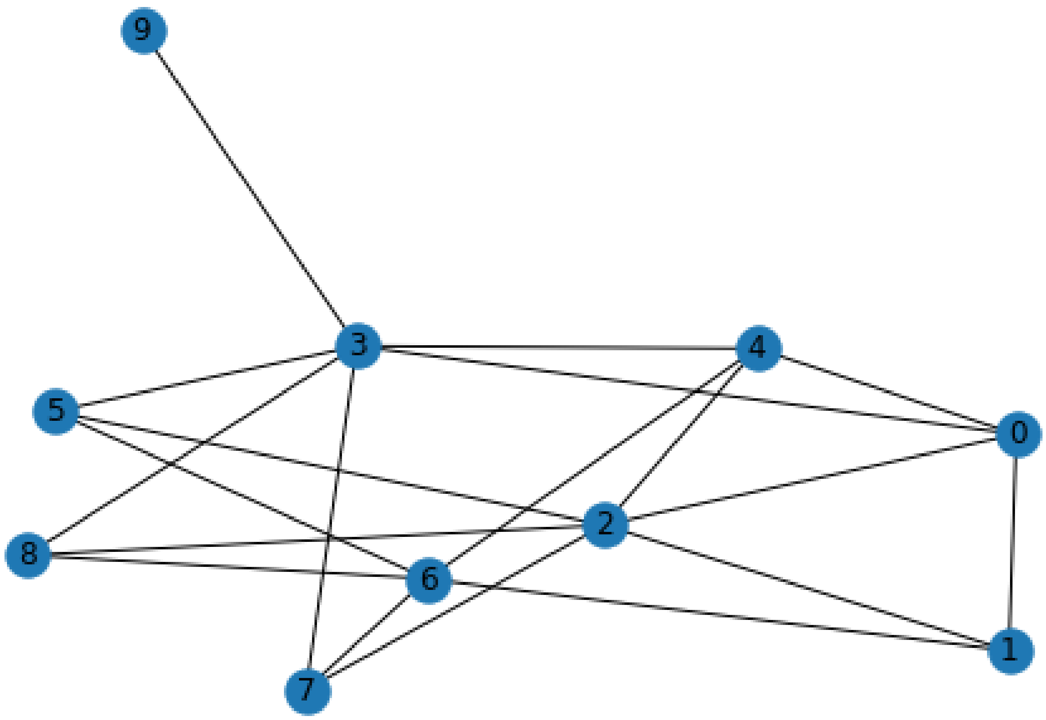
```
G=nx.Graph()
pos = {variables[0]:(2,3), variables[4]:(1,2), variables[1]:(3,2), variables[3]:
↪(0,1), variables[6]:(2,1), variables[2]:(3,1), variables[9]:(0,0),␣
↪variables[8]:(1,0), variables[5]:(2,0), variables[7]:(3,0)}

labels={}

for Xi in variables: G.add_node( Xi.label )
for i in range(10):
    for j in range(i,10):
        G.add_edge(i,j, weight=wts[i,j])
        labels[(i,j)] = np.round(wts[i,j],4)

nx.draw(G,pos);
nx.draw_networkx_edge_labels(G,pos,edge_labels=labels);
```



[125]:

```
factorsCL = [phat[i] for i in range(10)] + [phat[i,j]/phat[i]/phat[j] for i,j␣
↪in␣
↪[(0,1),(0,2),(0,3),(0,4),(1,2),(1,6),(2,4),(2,5),(2,7),(2,8),(3,4),(3,5),(3,7),(3,8),(3,9),
CL_model = gm.GraphModel(factorsCL)
```

[126]:
```
G2=nx.Graph()
CL_model.drawMarkovGraph(var_color='w', factor_color=(0.2, 0.2, 0.8))
nx.draw(G2,pos)
```



[127]:
```
(CL_model.nfactors)**2
```

[127]: 841

(c)

[128]:
```
np.mean([CL_model.logValue(x) for x in train])
```

[128]: -6.8006682013121615

[129]:
```
np.mean([CL_model.logValue(x) for x in valid])
```

[129]: -6.845340997624409

(d)

[130]:
```
(np.mean([model2.logValue(x) for x in train])-(307/2)*(np.log(len(train))/
↪len(train)))
```

[130]: -6.814455784522677

[131]:
```
(np.mean([CL_model.logValue(x) for x in train])-(841/2)*(np.log(len(train))/
↪len(train)))
```

[131]: -6.821304969088525

Thus we prefer the hand-designed model when counting the penalty in.