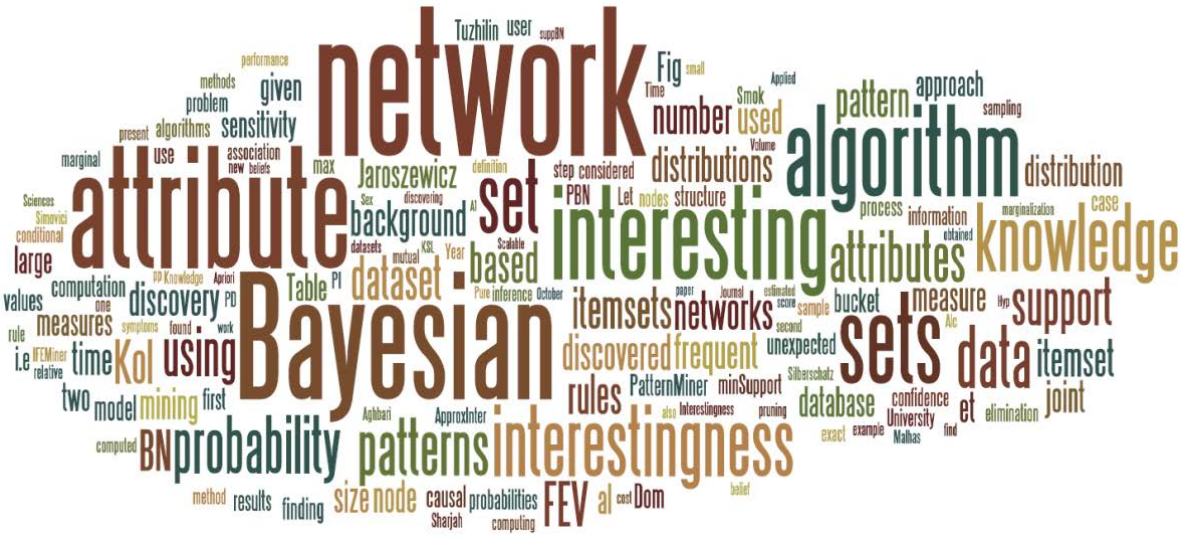
AI Final Project

Bayesian Network – MCMC

Cancer Analysis



Group 15

x1052047 李曄彤

103062336 莊若琪

103031111 許琨杰

Content

Preface

This is the final project report written by the group of five students who are taking Artificial Intelligence in autumn semester,2016.We are Jo-Chi Chuang,Kun Jieh Hsu and Li YeTong from National Tsing Tua University.

We will be explaining the idea of our project and the implementation process in detail and systematically. So as for clearer understanding of our project, the report is divided into several parts; each part with a specific focus.

Abstract

The goal of our project is focused on the probabilistic inference of the reasons contributing to lung cancer and breast cancer.The bayesian belief network is set up based on conditional probability tables,which are grabbed from professional essays and statistics from government and judged subjectively.In the Markov monte carlo algorithm,we use the known CPT to do more inferences about different nodes in the network.

Introduction

A Bayesian belief network, Bayesian model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Our project is aimed to make a cancer analysis.We find data from quantities of medical essays and ideas of experts online in terms of cancer and the likely elements leading to these cancers.After setting up the network,we use the MCMC algorithm to do inferences.When assuming some nodes in the same level to be true,we can get other nodes probability,so that we can achieve the goal of cancer analysis.

Dataset

We got data mainly from medical essays and statistics from government and other organizations.

There are 71 persons infected tuberculosis per 100,000 persons in China.

The possibility of people who are used to smoke infected tuberculosis is two to three times than normal people.

The possibility of people who are living in the environment full of air and particles is ten to thirteen times than normal people.

There are 28.9 percent of fat people who are easily caught in set-health.

There are 24 percent of people who are used to sleep less than 6 hours per day who are easily caught in set-health.

There are 30 percent of people who have had tuberculosis infecting lung cancer easily.

There is 1 person infected breast cancer every 300 persons in China.

The possibility of people who are in the sub-health state is sixteen times more than those healthy people.

According to the following formulas:

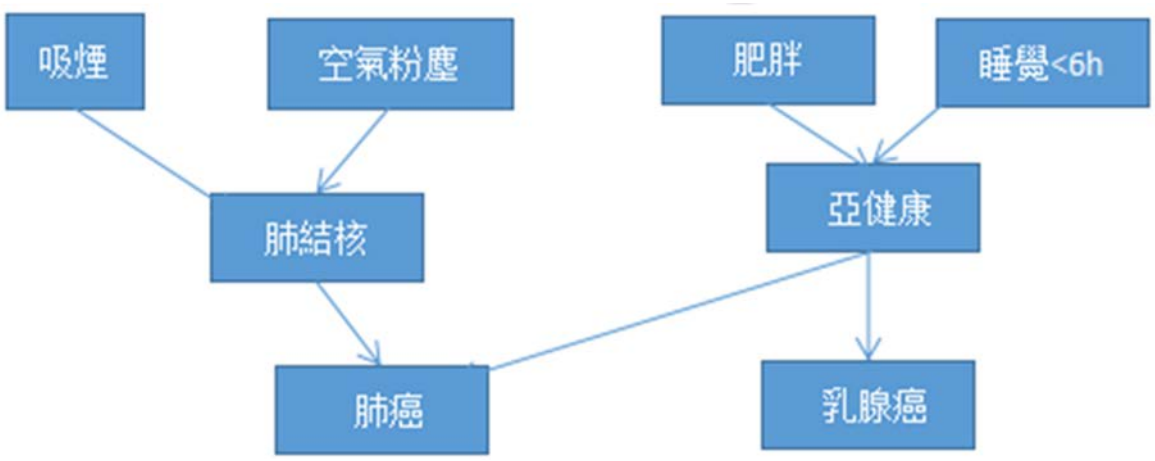
$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

$$P(A_i|B) = \frac{P(B|A_i)\,P(A_i)}{\sum_j P(B|A_j)\,P(A_j)},$$

Bayesian Belief Network

Our topic is cancer analysis based on many reasons.We choose the most common type of cancers——lung cancer and breast cancer as target nodes.These two cancers are proved to be the most possible cancer that human being may be infected.Which may contributes this disease?We think that unhealthy lifestyles and unhealthy living environment may lead to

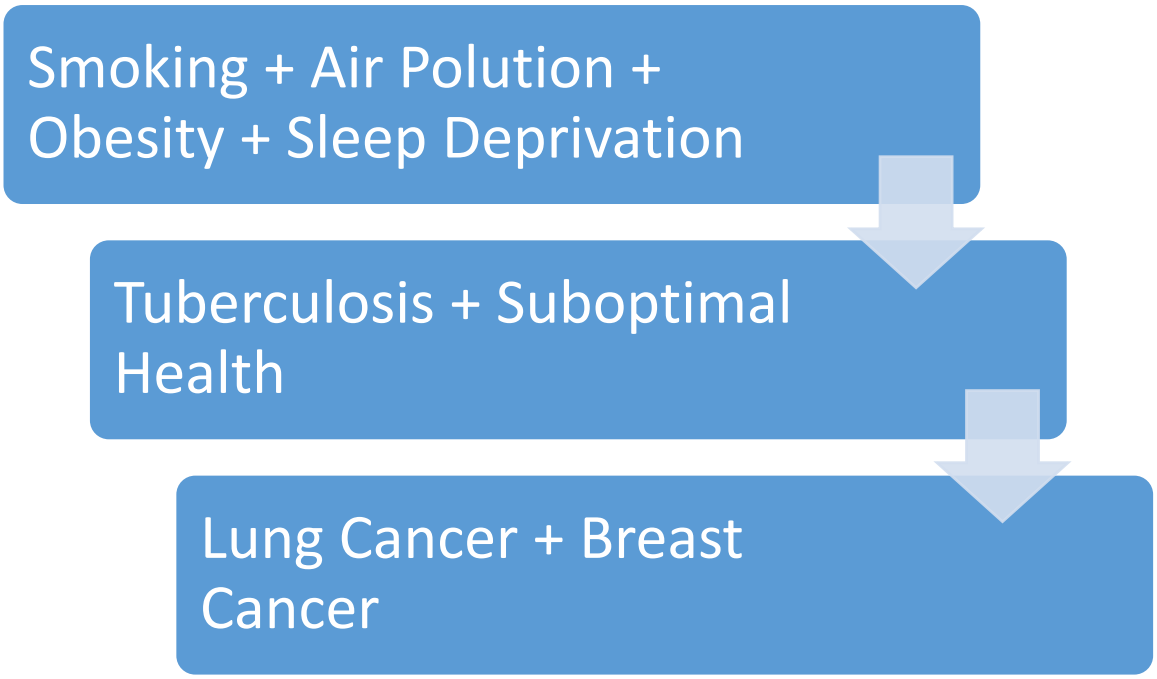
diseases and the state of sub-health.So we choose “smoking”,”air”,”fat”,”sleeping less than 6 hours” as the initial elements.Then we choose “tuberculosis” and “sub-health” as the second level node as the transition between the targets and initial elements.The bayesian belief network shows:



Smoking, Air Pollution, Obesity, Sleep	TT	TF	FT	FF

This is a DAG.We use MCMC to do inferences based on the above network.

In order to fit the requirements of MCMC algorithm,we gather nodes in the same level as one node,the network is turned to the following style:



According to the statics and formulas from the part “Dataset”,we can get the CPT:

TTTT	0.0878	0.0782	0.4412	0.3928
TTTF	0.0480	0.1180	0.2410	0.5930
TTFT	0.0398	0.1262	0.2002	0.6338
TTFF	0.0166	0.1494	0.0834	0.7506
TFTT	0.0074	0.0066	0.5216	0.4644
TFTF	0.0040	0.0100	0.2850	0.7010
TFFT	0.0034	0.0106	0.2366	0.7494
TFFF	0.0014	0.0126	0.0986	0.8874
FTTT	0.0503	0.0447	0.4787	0.4263
FTTF	0.0275	0.0675	0.2615	0.6435
FTFT	0.0228	0.0722	0.2172	0.6878
FTFF	0.0095	0.0855	0.0905	0.8145
FFTT	0.0033	0.0030	0.5257	0.4680
FFTF	0.0018	0.0045	0.2872	0.7065
FFFT	0.0015	0.0048	0.2385	0.7552
FFFF	0.0006	0.0057	0.0994	0.8943

Table1:

The column represents the joint probability of tuberculosis and sub-health.

The row represents the joint probability of “smoking”,”air”,”fat” and “sleeping less than 6 hours”.

Tuberculosis,	TT	TF	FT	FF
TT	0.0184	0.3266	0.0349	0.6201
TF	0.0010	0.2990	0.0023	0.6977
FT	0.0024	0.0426	0.0509	0.9041
FF	0.0003	0.0053	0.0530	0.9414

Table2:

The column represents the joint probability of tuberculosis and sub-health.

The row represents the joint probability of lung cancer and breast cancer.

MCMC

Below is the working principle of our self-designed MCMC algorithm:

1. User choose an “known node” and a “desired node”, then input the value of the known node.

2. MCMC uses CPT to get the values of other nodes from the known node.

3. MCMC records which variable in the value of the desired node has been selected in the “score”.

4. Step 2 to 3 is a cycle. Because CPT is composed by probabilities, the selected variable of each cycle may differ. However, after repeating at least ten thousand times and normalizing the score, the result converges to a probability distribution of the desired node.

Code Overview

```
class Node:
    def __init__(self, num_row, num_col):
        self.CPT = [[0 for x in range(num_row)] for y in range(num_col)]
    def getCPT(self, row, col):
        return self.CPT[row][col]
    def updateCPT(self, list):
        self.CPT = list;
    def printCPT(self):
        print(self.CPT)
    def getNumOfNextStates(self):
        return len(self.CPT[0])
    def getNumOfCurrentStates(self):
        return len(self.CPT)
nodes = []

input_layer = int(input("Input your known node (1 - 3): "))
output_layer = int(input("Input your desired node: "))
n1_state = int(input("Input the value of the known node (from CPT): "))
up_to_down = True;

n1 = Node(16, 4)
CPT_temp = [[0.0878,0.0782,0.4412,0.3928],[0.0480,0.1180,0.2410,0.5930],[0.0398,0.126
n1.updateCPT(CPT_temp)
n2 = Node(4, 4)
CPT_temp = [[0.0184,0.3266,0.0349,0.6201],[0.0010,0.2990,0.0023,0.6977],[0.0024,0.042
n2.updateCPT(CPT_temp)
```

Console Terminal Run TODO

```
if input_layer == 1:
    nodes.append(n1)
    num_of_targets = 4
    if output_layer == 3:
        nodes.append(n2)
elif input_layer == 2:
    if output_layer == 1:
        up_to_down = False
        nodes.append(n1)
        num_of_targets = 16
    elif output_layer == 3:
        nodes.append(n2)
        num_of_targets = 4
elif input_layer == 3:
    up_to_down = False
    if output_layer == 1:
        nodes.append(n1)
        nodes.append(n2)
        num_of_targets = 16
    elif output_layer == 2:
        nodes.append(n2)
        num_of_targets = 4

def probability(p): # p is a list
    p2 = 0.0
    for i in range(len(p)):
        p2 += p[i]
```

Console Terminal Run TODO

```
current_state = 0
next_state = 0
total_run = 0
print(99)
if up_to_down:
    for i in range(0, num_of_loop, 1):
        current_state = n1_state
        for n in range(0, len(nodes), 1):
            p = []
            for k in range(0, nodes[n].getNumOfNextStates(), 1):
                p.append(nodes[n].getCPT(current_state, k))
            next_state = probability(p)
            current_state = next_state
        score[next_state] += 1
        total_run += 1



else:
    un = 0
    while total_run < num_of_loop:
        first_input = random.randint(0, num_of_targets-1)
        current_state = first_input
        for n in range(0, len(nodes), 1):
            p = []
            for k in range(0, nodes[n].getNumOfNextStates(), 1):
                p.append(nodes[n].getCPT(current_state, k))
            next_state = probability(p)
```

Console Terminal Run TODO

art keyboard internationalization for PyCharm: We have found out that you are using a non-english keyboard layout. You can enable smart layout support for German languag

Demo

1. Run the MCMC.py

2. Input your known node. However, we only have three nodes at the moment, so the input can only be “1, 2 or 3”.

3. Input your desired node which is one of the other two nodes.

4. Input the value of the known node which can be found in the CPT.

5. Output is the probability distribution of the desired node.

For example, if you have “smoking”, “air pollution”, “obesity” and “sleep deprivation” problem, then the input value of the known node value will be “0” which can be found at the second row of the CPT as “T T T T”. If you want to know the probability of getting cancer, then you should input “1” which means the first node for the known node, “3” which means the third node for the desired node and “0” for the known node’s value.

Analysis

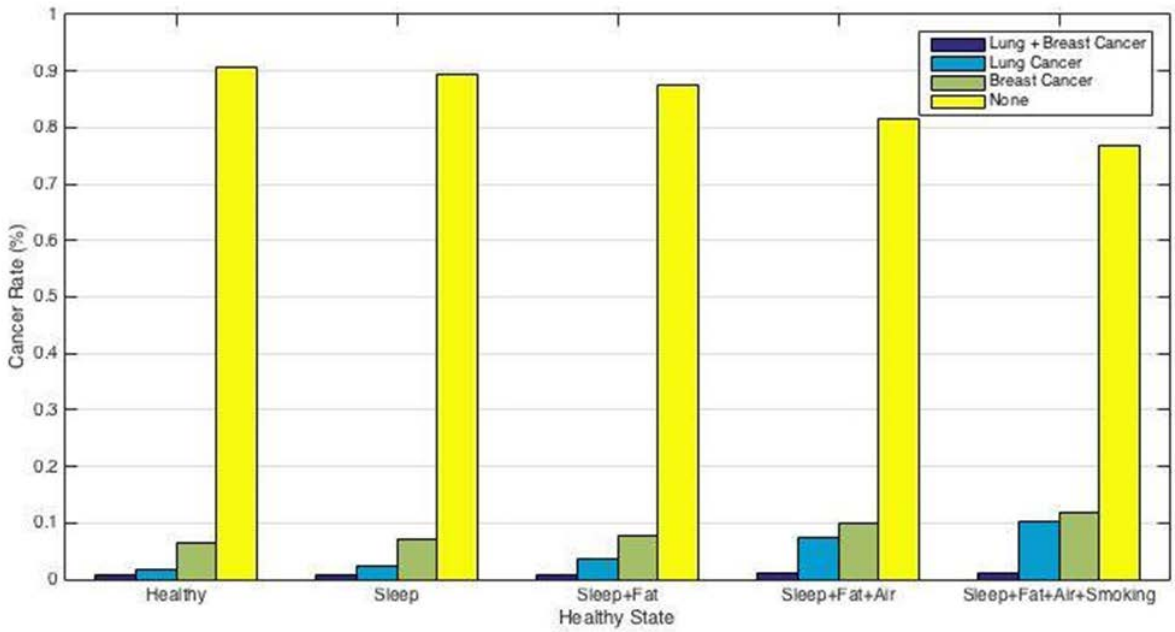


Figure 1

This figure shows the relationship between node 1 and node 3. In “Healthy”, we can see that if we input 0 which means no health problem to node 1, the user is least likely to get cancer. However, with more and more health problems are added, the probability of getting cancer rises. The most significant change occurs between “Sleep + Fat” and “Sleep + Fat + air”. In these two data, the probability of getting lung cancer has a considerable increment when the “air pollution” problem is added.

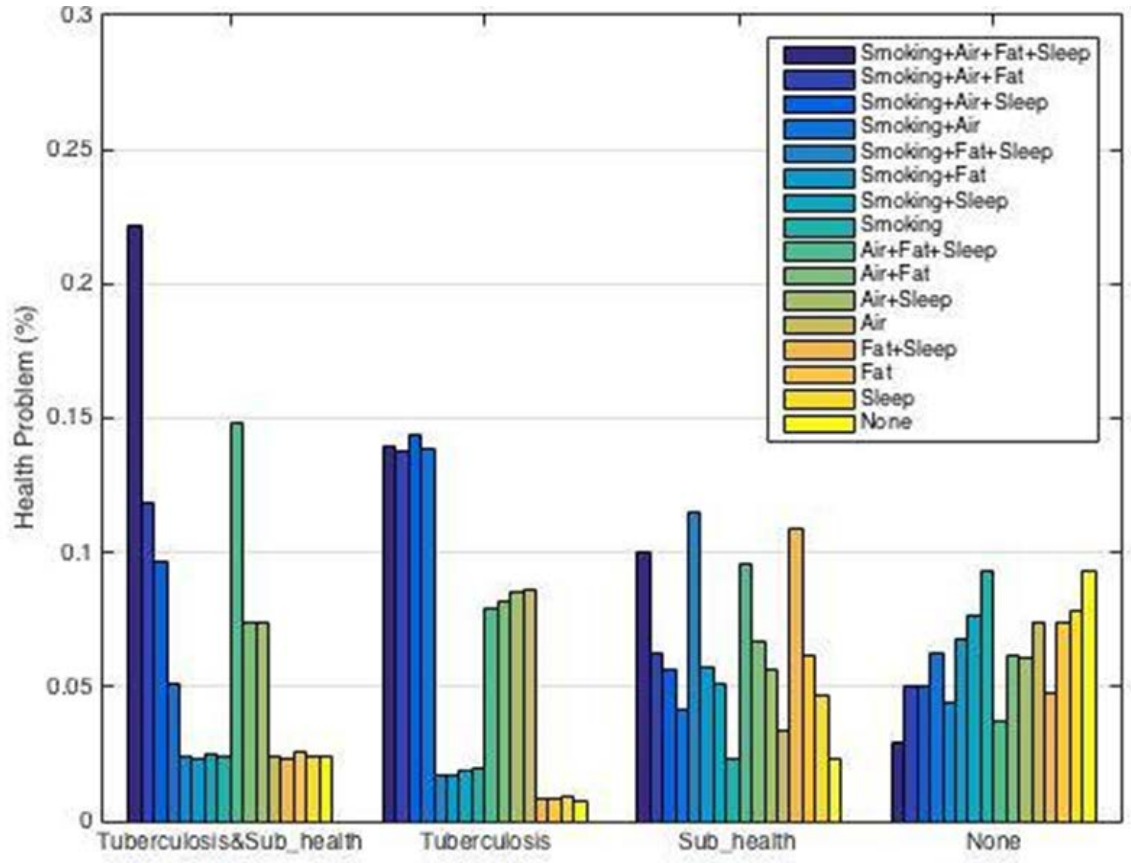
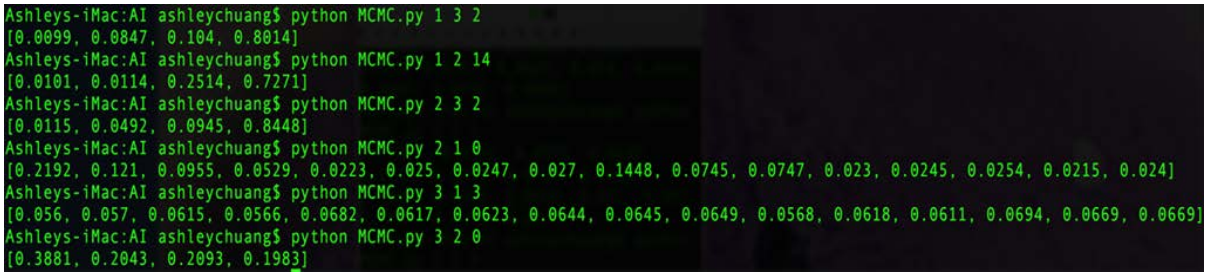


Figure 2

This figure shows the relationship between node 2 and node 1. In “Tuberculosis & Sub_health”, we can see that “Smoking + Air + Fat + Sleep” has the highest probability causing this problem and “Air + Fat + Sleep” is the second major. With the symptom getting less serious (from left to right), the probability distribution is getting uniform. Still, there’s an intersting thing. In the “Tuberculosis”, the first four and the ninth to twelfth columns has higher probability and all of them has a common factor, air pollution.

Evaluation cases



Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 15

[0.0088, 0.0214, 0.0698, 0.9]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 15

[0.0094, 0.0187, 0.0635, 0.9084]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 15

[0.0093, 0.0211, 0.064, 0.9056]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 15

[0.0094, 0.0209, 0.0635, 0.9062]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 14

[0.0099, 0.0242, 0.0702, 0.8957]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 14

[0.0094, 0.0248, 0.0724, 0.8934]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 14

[0.0091, 0.0241, 0.0693, 0.8975]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 14

[0.0094, 0.0264, 0.0689, 0.8953]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 14

[0.0107, 0.0268, 0.0715, 0.891]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 12

[0.0089, 0.0368, 0.0775, 0.8768]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 12

[0.0107, 0.0383, 0.0824, 0.8686]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 12

[0.0091, 0.0404, 0.0785, 0.872]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 12

[0.0112, 0.0411, 0.0756, 0.8721]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 12

[0.0095, 0.0353, 0.0767, 0.8785]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 8

[0.0098, 0.0777, 0.1016, 0.8109]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 8

[0.0105, 0.0756, 0.0995, 0.8144]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 8

[0.0109, 0.0792, 0.0978, 0.8121]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 8

[0.0098, 0.0744, 0.0985, 0.8173]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 8

[0.0117, 0.0756, 0.1015, 0.8112]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 0

[0.012, 0.1035, 0.1171, 0.7674]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 0

[0.0117, 0.1076, 0.1119, 0.7688]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 0

^[[A[0.0102, 0.1013, 0.119, 0.7695]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 0

^[[A[0.0117, 0.0968, 0.1157, 0.7758]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 1 3 0

[0.0119, 0.0994, 0.112, 0.7767]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 0

[0.2218, 0.1183, 0.0963, 0.0508, 0.0242, 0.0235, 0.0248, 0.0241, 0.1485, 0.0735, 0.0735, 0.024, 0.0231, 0.0259, 0.0237, 0.024]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 0

[0.2171, 0.1187, 0.0969, 0.05, 0.0247, 0.0236, 0.0256, 0.0244, 0.1488, 0.0723, 0.0766, 0.0254, 0.0227, 0.0242, 0.0248, 0.0242]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 0

[0.2212, 0.1294, 0.0933, 0.0443, 0.0238, 0.0247, 0.0248, 0.0247, 0.1482, 0.0742, 0.0703, 0.0253, 0.0227, 0.0238, 0.0238, 0.0255]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 0

[0.2211, 0.1256, 0.0934, 0.0519, 0.0221, 0.022, 0.0262, 0.0221, 0.1484, 0.0738, 0.0721, 0.0254, 0.0239, 0.0244, 0.0233, 0.0243]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 0

[0.2149, 0.119, 0.1027, 0.0477, 0.026, 0.0268, 0.0228, 0.0251, 0.1468, 0.0764, 0.0727, 0.024, 0.0239, 0.0224, 0.0241, 0.0247]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 1

[0.1399, 0.138, 0.1439, 0.1387, 0.0174, 0.0168, 0.0185, 0.0196, 0.0792, 0.0822, 0.0854, 0.0863, 0.0085, 0.0086, 0.0092, 0.0078]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 1

[0.1308, 0.137, 0.1396, 0.1487, 0.019, 0.0176, 0.0203, 0.0163, 0.0842, 0.0853, 0.0837, 0.0838, 0.0081, 0.009, 0.0088, 0.0078]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 1

[0.1325, 0.1426, 0.1401, 0.1446, 0.0192, 0.0179, 0.0162, 0.0165, 0.0844, 0.0855, 0.0822, 0.0873, 0.0082, 0.0087, 0.0075, 0.0066]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 1

[0.1368, 0.1401, 0.1434, 0.1426, 0.0149, 0.0172, 0.016, 0.0169, 0.0829, 0.0803, 0.0856, 0.091, 0.0076, 0.0092, 0.0083, 0.0072]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 1

[0.1382, 0.1454, 0.1389, 0.1462, 0.0167, 0.0181, 0.0145, 0.0189, 0.0776, 0.0778, 0.0817, 0.0871, 0.0111, 0.0093, 0.0107, 0.0078]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 2

[0.0998, 0.0622, 0.0565, 0.0419, 0.1147, 0.0575, 0.0512, 0.0231, 0.0959, 0.0669, 0.0566, 0.0334, 0.1086, 0.0614, 0.0471, 0.0232]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 2

[0.0887, 0.0704, 0.0604, 0.0426, 0.0979, 0.0613, 0.0539, 0.0233, 0.1024, 0.0655, 0.0591, 0.0336, 0.1074, 0.0607, 0.0516, 0.0212]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 2

2[0.0945, 0.0656, 0.0579, 0.0433, 0.1012, 0.0633, 0.0481, 0.0232, 0.1056, 0.0629, 0.0584, 0.0321, 0.1117, 0.0596, 0.0486, 0.024]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 2

2[0.0925, 0.0641, 0.0603, 0.0445, 0.101, 0.061, 0.0515, 0.0264, 0.0975, 0.0653, 0.0564, 0.0341, 0.1105, 0.0598, 0.053, 0.0221]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 2

[0.093, 0.0674, 0.0573, 0.0429, 0.1025, 0.0617, 0.0561, 0.0268, 0.1035, 0.0612, 0.0555, 0.0329, 0.1079, 0.0591, 0.05, 0.0222]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 3

[0.0292, 0.0504, 0.0504, 0.0623, 0.0443, 0.0682, 0.0762, 0.0931, 0.0371, 0.0614, 0.0608, 0.0742, 0.048, 0.0736, 0.078, 0.0928]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 3

[0.0316, 0.0505, 0.0513, 0.0653, 0.0444, 0.0735, 0.0749, 0.0883, 0.0362, 0.0576, 0.0622, 0.0812, 0.0459, 0.0685, 0.0808, 0.0878]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 3

[0.0319, 0.0466, 0.0527, 0.0661, 0.0452, 0.0709, 0.0804, 0.0867, 0.036, 0.0583, 0.0609, 0.074, 0.0484, 0.0685, 0.0792, 0.0942]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 3

[0.0327, 0.0506, 0.0573, 0.0642, 0.0479, 0.0715, 0.0737, 0.0908, 0.0369, 0.0599, 0.0653, 0.072, 0.047, 0.0651, 0.0776, 0.0875]

Ashleys-iMac:AI ashleychuang$ python MCMC.py 2 1 3

[0.0309, 0.05, 0.0518, 0.0663, 0.0432, 0.0733, 0.0742, 0.0925, 0.0363, 0.0626, 0.0602, 0.0731, 0.0482, 0.0704, 0.0778, 0.0892]

Eample:

CASE 1:

Assume that someone wants to know the probability of being diagnosed with what kinds of cancers (Layer 3), given that we already know the health state of the body (Layer 1).

>> python MCMC.py

>> Input your known node (1~3): <- You should enter 1 (Already know health condition) in this case

>> Input your desired node: <- 3 in this case (Would like to know the probability of getting cancers)

>> Input the value of the know node (from CPT): <- Please check CPT and select what health condition you are in. Since there're 16 states you may be in, enter 0~15, each represents one health condition in the first CPT.

OUTPUT:

The probability of being diagnosed with both lung cancer and breast: 0.0088

The probability of being diagnosed with only lung cancer: 0.027

The probability of being diagnosed with only breast cancer: 0.0752

The probability of not being diagnosed with both lung cancer and breast: 0.889

CASE 2:

Assume that someone already know that he or she has Tuberculosis or Sub-health (Layer 2), and wants to know which health condition he or she may be in (Layer 1).

>> python MCMC.py

>> Input your known node (1~3): <- You should enter 2 (Already know has Tuberculosis/Sub-health or not) in this case

>> Input your desired node: <- 1 in this case (Would like to know the in which health condition he or she may be)

>> Input the value of the know node (from CPT): <- Please check CPT and select which state of tuberculosis or sub-health do you match. Since there're 4 states you may be in, enter 0~3, each represents a state in the second CPT of tuberculosis or sub-health.

OUTPUT:

The probability of having problem with smoking, air pollution, fat and sleeping: 0.0296

The probability of having problem with smoking, air pollution and fat: 0.0483

The probability of having problem with smoking, air pollution, and sleeping: 0.0519

The probability of having problem with smoking and air pollution: 0.0661

The probability of having problem with smoking, fat and sleeping: 0.0494

The probability of having problem with smoking, and fat: 0.0716

The probability of having problem with smoking, and sleeping: 0.0688

The probability of having problem with smoking: 0.0885

The probability of having problem with air pollution, fat and sleeping: 0.0375

The probability of having problem with air pollution, and fat: 0.0562

The probability of having problem with air pollution, and sleeping: 0.0613

The probability of having problem with air pollution: 0.0804

The probability of having problem with fat and sleeping: 0.0475

The probability of having problem with fat: 0.076

The probability of having problem with sleeping: 0.0756

The probability of not having any problem with smoking, air pollution, fat and sleeping: 0.0913

Further Improvement

We now have the function of the same level inference.In other words, given a node,we can only have probabilities of nodes in a same level through MCMC. We wanna get the function that whichever node probability we assumes,any of the nodes probabilities we can achieve(eg.given the “breast cancer” true,we can get the “fat” probability and “ sub-health” probability at a time,rather than execute several times.

Conclusion

The purpose and objectives of our bayesian belief network inferences and MCMC simulation is achieved. Now, users can do the cancer analysis using the correct demo way.

Although we didn’t generate the CPTs by ourselves, both of the MCMC and CPTs still did a great job by giving us a clear and resonable view on the relationships between these factors. And among of them, the air pollution seems to be the most important one which contributes significant changes in both figures.

Reference and Bibliography

《亞健康人群的中醫體質特點分析》，《廣州中醫藥大學學報》編號 R211

《高校青年人群亚健康调查及中医基本证候分析》，《中國醫藥科學》2012 年 12 期

《中國癌症流行病學與防治研究現狀》，知網《化學進展》

《2003～2007 年中國癌症發病分析》，知網《中國腫瘤》

Task allocation

李曄彤：Data Collecion, Bayesian Belief Network, CPT, Report

莊若琪：MCMC Emulation, Coding, Report

許琨杰：MCMC Emulation, Coding, Report