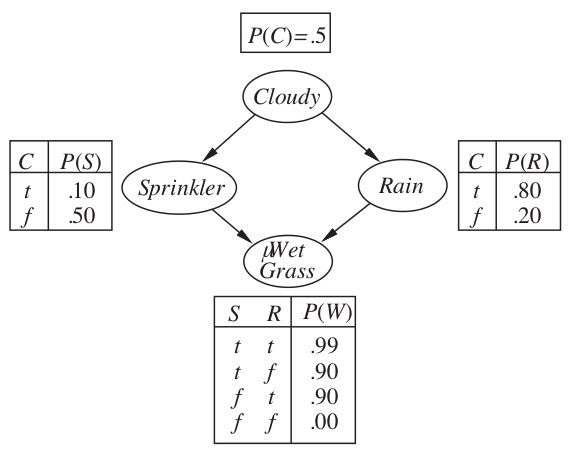
**实验二 采样算法之草地浇水问题**

**姓名：杨志烨 学号：16S103168**

1. **问题描述**
2. **待解决问题的解释**

“草地浇水”问题由多联通贝叶斯网络表示，如下图所示。本实验使用直接采样算法、拒绝采样算法、似然加权采样算法以及吉布斯采样算法计算该贝叶斯网络的查询，以计算条件概率为例。



**(2)采样方法如何应用于该问题**

采样是根据一个概率分布模型得到一个随机样本的过程，使用采样方法求解基于贝叶斯网络的条件概率查询问题的思想非常简单，即使用频率来近似概率。因此，使用采样算法求解条件概率是一种近似方法，由概率论中的大数定理可知，采样的事件越多，得到的解就越精确。

1. **算法介绍**

**(1)采样算法的一般介绍**

由于采样方法差别不大，之前对采样也不理解，所以借助本次实验机会实现了老师提到的四种采样方法。

**直接采样算法**：任何采样算法中最基本的要素是根据已知概率分布生成样本。对于贝叶斯网络而言，直接采样算法即按照网络的拓扑结构依次对每个变量进行采样，被采样变量值的概率分布依赖于父节点已得到的赋值，最终得到一个完整的服从贝叶斯网络表示的概率分布的样本。

**拒绝采样算法**：拒绝采样算法是一类由一个易于采样的分布出发，为一个难以直接采样的分布产生采样样本的通用算法，可以被用于计算条件概率。拒绝采样算法首先根据贝叶斯网络指定的概率分布生成采样样本，然后，它拒绝所有与证据e不匹配的样本，最后通过在剩余的样本中对时间X=x的出现频繁程度计数从而得到估计概率。

**加权采样算法**：加权采样算法固定证据变量E的值，只对除证据以外的其余变量进行采样，这保证了生成的每个采样样本都与证据一致。同时，在对查询变量的分布进行计数之前，把根据证据得到的事件的似然作为每个时间的权值，这个权值通过每个证据变量在给定父节点取值下的条件概率的乘积进行度量。

**吉布斯采样算法**：吉布斯采样算法与拒绝采样算法和加权采样算法不同，它没有为每个事件都重新生成样本，而是通过对前一个事件进行随机改变而生成每个事件样本。因此可以认为网络处于为每个变量指定了值的一个特定的当前状态，而下一个状态则通过对某个非证据变量进行采样来产生，取决与这个变量的马尔可夫覆盖中的变量的当前值。

**(2)算法伪代码**

**直接采样算法**：

**function** direct-sample(bn) **returns** an event sampled from the bayesian network bn

**inputs**: bn, a Bayesian network

x = an event with n elements

**for** i = 1 **to** n **do**

x[i] = a random sample from P(Xi|parents(Xi))

**return** x

**拒绝采样算法**：

**function** rejection-sample(X, e, bn, N) **returns** an estimate of P(X|e)

**inputs**: X, the query variable

e, evidence specified as an event

bn, a Bayesian network

N, the total number of samples to be generated

**local variable**: C, a vector of counts over X

**for** j = 1 **to** N **do**

x = direct-sample(bn)

if x is consistent with e then

C[x] = C[x] + 1 where x is the value of X in x

**return** normalize(C)

**加权采样算法**：

**function** likehood-weighting(X, e, bn, N) **returns** an estimate of P(X|e)

**inputs**: X, the query variable

e, evidence specified as an event

bn, a Bayesian network

N, the total number of samples to be generated

**local variable**: C, a vector of counts over X

**for** j = 1 **to** N **do**

x, w = weighted-sample(bn)

if x is consistent with e then

C[x] = C[x] + w where x is the value of X in x

**return** normalize(C)

**function** weighted-sample(bn, e) **returns** an event and a weight

x = an event with n elements;

w = 1

**for** i = 1 **to** n do

**if** X[i] has a value x[i] in e **then**

w = w \* P(X[i] = x[i]|parents(X[i]))

**else**

x[i] = a random sample from P(X[i]|parents(X[i]))

**return** x, w

**吉布斯采样算法**：

**function** gibbs-sample(X, e, bn, N) **returns** an estimate of P(X|e)

**inputs**: X, the query variable

e, evidence specified as an event

bn, a Bayesian network

N, the total number of samples to be generated

**local variable**: C, a vector of counts over X

Z, the nonevidence variable in bn

x, the current state of the network, initial copied from e

initialize x with random values from the variable in Z

**for** j = 1 **to** N **do**

C[x] = C[x] + 1 where x in the values of X

**for each** Z[i] **in** Z **do**

sample the value of Z[i] in x from P(Z[i]|mb(Z[i])) given the values of MB(Z[i]) in x

**return** normalize(C)

**3.算法实现**

**(1)实验环境与问题规模**

实验环境:

系统:Macbook Pro , Mac OS 10.11.5

语言及编译器:python,python2.7

IDE:PyChram

1. **数据结构：**

主要数据结构为BayesNetwork类, 主要的属性有:

adj\_matrix：矩阵，贝叶斯网络结构的邻接矩阵表示。

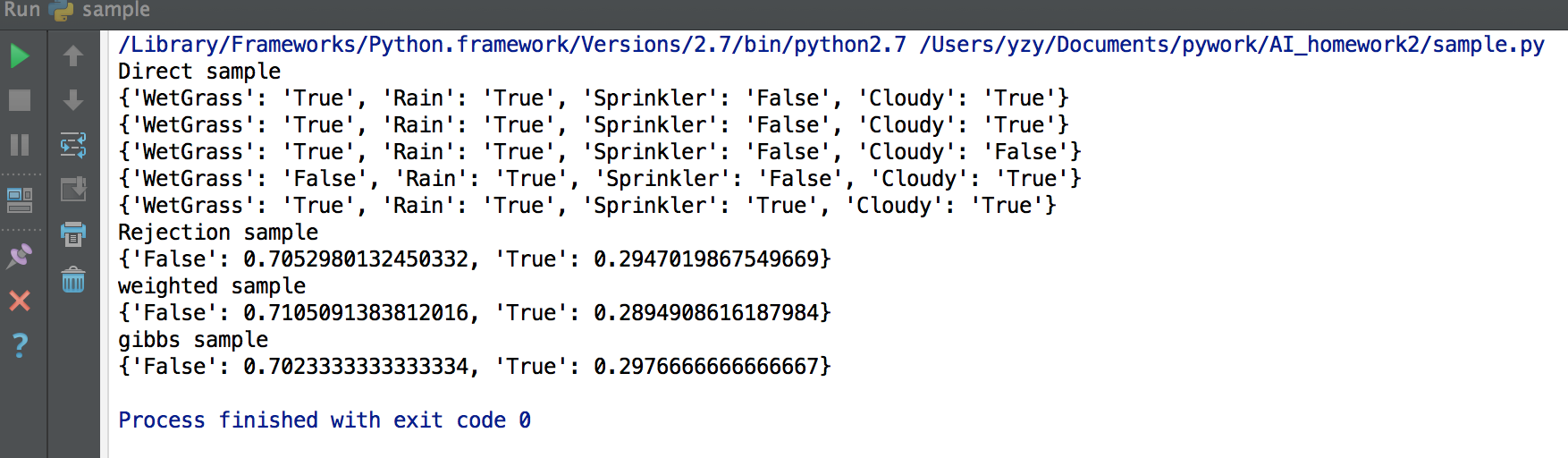
CPT：哈希表，表示贝叶斯网络中每个变量的条件概率表。键为变量名，值为该变量对应的条件概率表。每个变量的条件概率表同样也是一个哈希表，键为条件，值为该条件下的概率。

BayesNetwork类有load\_from\_file方法，接受一个文件名，从该文件名中读取贝叶斯网络的内容。本次实验所采用的贝叶斯网络的文件见附录II。

**(3)实验结果:**

可以使用直接采样得到N(5)组数据，并使用三种采样算法计算

**(4)系统中间及最终结果**



**参考文献**

[1] Stuart J.Russell , Peter Norvig《人工智能-一种现代的方法(第三版)》

[3]关于贝叶斯网络构造部分代码参考自网络和师兄解答

**附录一 代码及其注释**

**BayesNetwork.py**

#!/usr/bin/env python

# coding:utf8

from collections import defaultdict

import copy

# 规范化处理

def normalize(C):

s = float(sum(C.values()))

return {k:C[k]/s for k in C}

class BayesNetwork:

def \_\_init\_\_(self):

self.variable\_matrix = []

def read\_vn\_from\_file(self, fp):

line = fp.readline()

return line.split()

def read\_am\_from\_file(self, fp, n):

adj\_matrix = []

for i in range(n):

line = fp.readline()

row = map(int, line.strip().split())

adj\_matrix.append(row)

return adj\_matrix

def parse\_cpt(self, line):

line = ''.join(line.split())

parts = line.split('|')

condition = {}

if len(parts) > 1:

# 解析条件

for item in parts[0].split(','):

rv, val = item.split('=')

condition[rv] = val

# 解析随机变量及其分布

variable, dist\_str = parts[-1].split('=')

dist = defaultdict(float)

for item in dist\_str[1:-1].split(','):

val, prob = item.split(':')

dist[val] = float(prob)

return condition, variable, dist

def read\_cpt\_from\_file(self, fp):

cpt = defaultdict(dict)

while True:

line = fp.readline()

line = line.strip()

if not line:

break

condition, variable, dist = self.parse\_cpt(line)

cond\_tuple = tuple([(rv, condition[rv])

for rv in sorted(condition.keys())])

cpt[variable][cond\_tuple] = dist

return cpt

# 从配置文件中读取BN

def load\_from\_file(self, fname):

fp = open(fname, 'r')

while True:

line = fp.readline()

if not line:

break

if not line.strip():

continue

seg = line.strip()[:-1]

if seg == 'Variable Name':

self.variable\_list = self.read\_vn\_from\_file(fp)

elif seg == 'Adjacendy Matrix':

self.adj\_matrix = self.read\_am\_from\_file(fp, len(self.variable\_list))

elif seg == 'CPT':

self.CPT = self.read\_cpt\_from\_file(fp)

# 查询指定随机变量的条件概率分布

def query\_cond\_dist(self, variable, condition):

cond\_tuple = tuple([(rv, condition[rv])

for rv in sorted(condition.keys())])

return copy.copy(self.CPT[variable][cond\_tuple])

# 获取网络中所有随机变量的一个拓扑序列

def get\_topo\_order(self):

topo\_order = []

n = len(self.variable\_list)

am = copy.deepcopy(self.adj\_matrix)

for k in range(n):

# 寻找起始节点

for i in range(n):

if i in topo\_order or any(am[j][i] for j in range(n)):

continue

else:

break

topo\_order.append(i)

# 更新邻接矩阵

for i in range(n):

for j in range(n):

am[i][j] = 0

am[j][i] = 0

return [self.variable\_list[i] for i in topo\_order]

# 查询指定随机变量的父结点列表

def query\_parents(self, variable):

n = len(self.variable\_list)

try:

idx = self.variable\_list.index(variable)

except ValueError:

return []

return [self.variable\_list[j] for j in range(n)

if self.adj\_matrix[j][idx]]

# 查询指定随机变量子结点列表

def query\_children(self, variable):

n = len(self.variable\_list)

try:

idx = self.variable\_list.index(variable)

except ValueError:

return []

return [self.variable\_list[j] for j in range(n)

if self.adj\_matrix[idx][j]]

# 给定其它所有随机变量的取值,计算指定随机变量的分布

def query\_whole\_cond\_dist(self, variable, evidence):

parents = self.query\_parents(variable)

condition = {k:evidence[k] for k in parents}

cond\_dist = self.query\_cond\_dist(variable, condition)

children = self.query\_children(variable)

for val in cond\_dist:

for c in children:

parents = self.query\_parents(c)

condition = {k:evidence[k] for k in parents}

condition[variable] = val

c\_cond\_dist = self.query\_cond\_dist(c, condition)

cond\_dist[val] \*= c\_cond\_dist[evidence[c]]

return normalize(cond\_dist)

if \_\_name\_\_ == '\_\_main\_\_':

BN = BayesNetwork()

BN.load\_from\_file('wet\_glass.txt')

print BN.get\_topo\_order()

print BN.adj\_matrix

print BN.query\_parents('Rain')

print BN.query\_cond\_dist('Cloudy', {})

**sample.py**

#!/usr/bin/env python

# coding:utf8

import random

from collections import defaultdict

from BayesNetwork import BayesNetwork

# 根据某个随机变量的分布进行采样

def sample(dist):

rand = random.random()

sd = 0

for val in sorted(dist):

if sd <= rand < sd + dist[val]:

return val

sd += dist[val]

# 直接采样

def direct\_sample(bn):

vl = bn.get\_topo\_order()

event = {}

for rv in vl:

parents = bn.query\_parents(rv)

condition = {k:event[k] for k in parents}

cond\_dist = bn.query\_cond\_dist(rv, condition)

event[rv] = sample(cond\_dist)

return event

# 判断事件event与证据evidence是否一致

def is\_consistent(event, evidence):

return all(event[rv] == evidence[rv] for rv in evidence)

# 规范化处理

def normalize(C):

s = float(sum(C.values()))

return {k:C[k]/s for k in C}

# 拒绝采样

def rejection\_sample(variable, evidence, bn, N):

C = defaultdict(int)

for i in xrange(N):

event = direct\_sample(bn)

if is\_consistent(event, evidence):

val = event[variable]

C[val] += 1

return normalize(C)

# 加权采样,返回一个样本及其对应的权值

def weighted\_sample(bn, evidence):

vl = bn.get\_topo\_order()

event = {}

w = 1.0

for rv in vl:

parents = bn.query\_parents(rv)

condition = {k:event[k] for k in parents}

cond\_dist = bn.query\_cond\_dist(rv, condition)

if evidence.has\_key(rv):

val = evidence[rv]

event[rv] = val

w \*= cond\_dist[val]

else:

event[rv] = sample(cond\_dist)

return event, w

# 使用似然加权方法计算分布

def likelihood\_weighting(variable, evidence, bn, N):

W = defaultdict(float)

for i in xrange(N):

event, w = weighted\_sample(bn, evidence)

val = event[variable]

W[val] += w

return normalize(W)

def gibbs\_sample(variable, evidence, bn, N):

C = defaultdict(int)

# 初始化

event = direct\_sample(bn)

Z = [] # 非证据变量

for rv in event:

if rv in evidence:

event[rv] = evidence[rv]

else:

Z.append(rv)

# 迭代

for i in range(N):

for z in Z:

cond\_dist = bn.query\_whole\_cond\_dist(z, event)

event[z] = sample(cond\_dist)

val = event[variable]

C[val] += 1

return normalize(C)

if \_\_name\_\_ == '\_\_main\_\_':

bn = BayesNetwork()

bn.load\_from\_file('wet\_glass.txt')

# 直接采样示例

print 'Direct sample'

for i in range(5):

print direct\_sample(bn)

# 拒绝采样示例

print 'Rejection sample'

print rejection\_sample('Rain', {'Sprinkler':'True'}, bn, 1000)

# 加权采样示例

print 'weighted sample'

print likelihood\_weighting('Rain', {'Sprinkler':'True'}, bn, 1000)

# 吉布斯采样示例

print 'gibbs sample'

print gibbs\_sample('Rain', {'Sprinkler':'True'}, bn, 1000)

**附录二：贝叶斯网络配置文件**

wet\_glass.txt

Variable Name:

Cloudy Sprinkler Rain WetGrass

Adjacendy Matrix:

0 1 1 0

0 0 0 1

0 0 0 1

0 0 0 0

CPT:

Cloudy={True:0.5, False:0.5}

Cloudy=True | Sprinkler={True:0.1, False:0.9}

Cloudy=False | Sprinkler={True:0.5, False:0.5}

Cloudy=True | Rain={True:0.8, False:0.2}

Cloudy=False | Rain={True:0.2, False:0.8}

Sprinkler=True, Rain=True | WetGrass={True:0.99, False:0.01}

Sprinkler=True, Rain=False | WetGrass={True:0.9, False:0.1}

Sprinkler=False, Rain=True | WetGrass={True:0.9, False:0.1}

Sprinkler=False, Rain=False | WetGrass={True:0, False:1}