**哈尔滨工业大学**

**人工智能实验三**

**报告**

**(2015年度秋季学期)**

**授课教师 赵铁军 郑德权**

**学生姓名 张 琦**

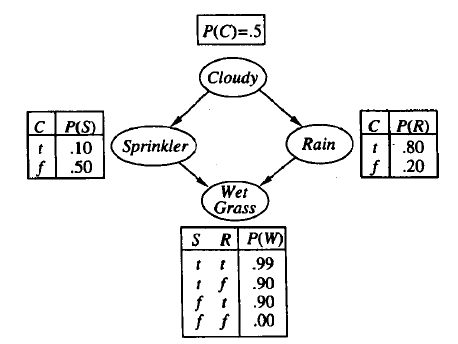
**学 号 15S003055**

**学 院 计算机科学与技术**

**一、问题描述**

**1.1 待解决问题的解释**

“草地浇水”多联通贝叶斯网络问题：



采样估计图中给出的所有概率

**1.2 问题的形式化描述**

有四个事件：多云C、浇水S、下雨R、草湿W可以进行观测，

估计P(C)、P(S|C)、P(S|~C)、P(R|C)、P(R|~C)、P(W|S,R)、P(W|S,~R)、P(W|~S,R)、P(W|~S,~R)

P(X)表示X事件发生的概率

P(X|Y)表示X事件在Y事件已发生时的条件概率

~表示逻辑非

**1.3 直接采样算法如何应用于该问题**

1. 利用给出的概率产生足够的样本数据。先以0.5的概率产生是否多云，然后再根据多云的情况产生是否浇水以及下雨，再根据浇水和下雨的情况决定是否草地湿润，如此得到一组样本，再将此过程重复N次。
2. 用样本数据估计这些给出的概率。统计(C), (S, C), (S, ~C), (R, C), (R, ~C), (S, R), (S, ~R), (~S, R), (~S, ~R), (W, S, R), (W, S, ~R), (W, ~S, R), (W, ~S, ~R)这些事件发生的次数，然后使用以下公式估计概率：(n(X)表示事件X发生的次数)

P(C) = n(C)/N

P(S|C) = n(S, C)/n(C)

P(S|~C) = n(S, ~C)/N – n(C)

P(R|C) = n(R, C)/n(C)

P(R|~C) = n(R, ~C)/n(~C)

P(W|S,R) = n(W, S, R)/n(S, R)

P(W|S,~R) = n(W, S, ~R)/n(S, ~R)

P(W|~S,R) = n(W, ~S, R)/n(~S, R)

P(W|~S,~R) = n(W, ~S, ~R)/n(~S, ~R)

1. 观察直接采样算法在N不同时得到的估计值与真实值之间的关系。

**二、算法介绍**

**2.1直接采样算法一般介绍**

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，最简单种类的随机采样过程是对没有与之关联的证据的网络事件进行采样。其思想是按照拓扑次序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。

在任何采样方法中，都是通过对实际生成的样本进行计数来计算答案的。假设总共有N个样本，令N(x1,…,xn)为特定事件x1,…,xn发生的频率。我们期望这个频率在极限情况下能够收敛到根据采样概率得到的它的期望值：



只要我们使用约等于符号（“≈”），我们要表达的就是这个含义——也就是说，估计概率在大量样本极限下成为精确值。这样的估计被成为一致的（consistent）。例如，可以为不完全指定事件x1,…,xm的概率产生一个一致估计，其中m<=n，如下所示：



也就是说，可以用采样过程中生成的、能与不完全指定事件相匹配的完整事件所占的比例来估计该事件的概率。这就是直接采样算法。

**三、算法实现**

**3.1 实验环境与问题规模**

实验环境：windows10+codeblocks

编程语言：C++

问题规模 20,100,1000,10000

**3.2 数据结构**

float cloudy = 0.5;//多云的概率

float sprinkle[2] = {0.1, 0.5};//sprinkle[0]表示在多云时候浇灌的概率，sprinkle[1]表示在没有多云时浇灌的概率

float rain[2] = {0.8, 0.2};//rain[0]表示在多云的时候下雨的概率，rain[1]表示在没有多云时下雨的概率

float wet[2][2];/\*wet[0][0]表示sprinkle且rain时wetgrass的概率；

wet[0][1]表示sprinkle且未下雨时wetgrass的概率；

wet[1][0]表示未浇灌且rain时wetgrass的概率；

wet[1][1]表示未浇灌且未下雨时wetgrass的概率；\*/

int Cnum = 0;//统计多云的次数

int Snum = 0;//统计浇灌的次数

int Rnum = 0;//统计下雨的次数

int Wnum = 0;//统计草湿的次数

int SCnum = 0;//统计多云且浇灌的次数

int SNCnum = 0;//统计未多云且浇灌的次数

int RCnum = 0;//多云且下雨的次数

int RNCnum = 0;//未多云且下雨的次数

int SRnum = 0;//浇灌且下雨的次数

int SNRnum = 0;//浇灌且未下雨的次数

int NSRnum = 0;//未浇灌且下雨的次数

int NSNRnum = 0;//未浇灌且未下雨的次数

int WSRnum = 0;//草湿且浇灌且下雨的次数

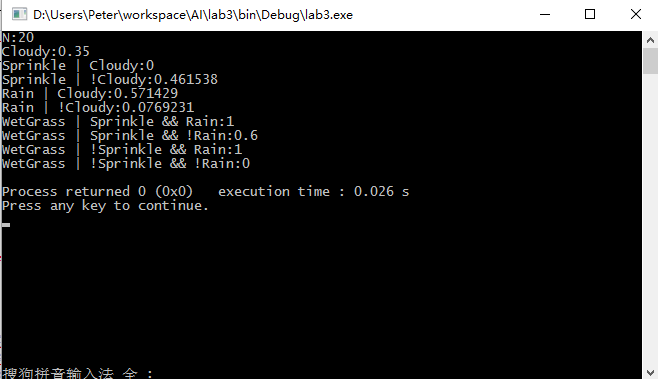
int WNSRnum = 0;//草湿且未浇灌且下雨的次数

int WSNRnum = 0;//草湿且浇灌且未下雨的次数

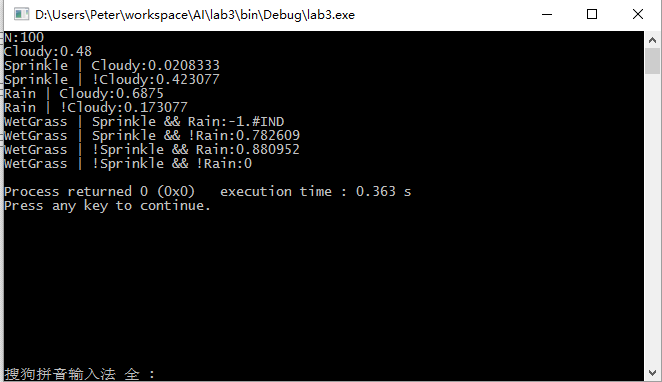
int WNSNRnum = 0;//草湿且未浇灌且未下雨的次数

**3.3 最终输出结果**

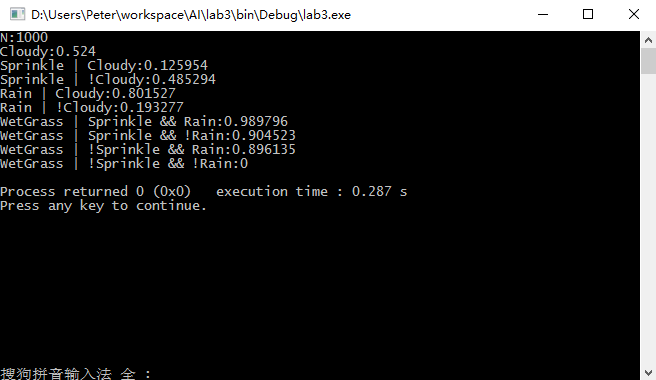
N=20时：



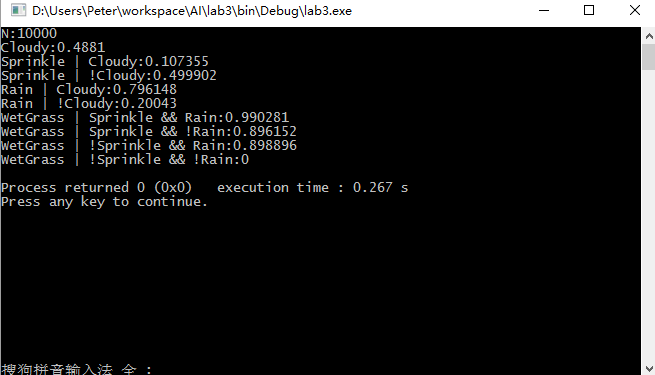
N=100时：



N=1000时：



N=10000时：



**3.4 实验结论**

观察可知，采样次数越大，得到的估计值与真实值之间的误差越小；采样次数过小时，估计值误差较大。

故可得结论：直接采样算法在采样次数比较大的时候可以用来估计概率，因为其频率在样本量趋近无穷时的值可以认为是概率的真实值。

**附录— 源代码及其注释**

#include <iostream>

#include <time.h>

#include <stdlib.h>

using namespace std;

float random01()//0-1之间产生浮点随机数

{

return rand()/float(RAND\_MAX);

}

float cloudy = 0.5;//多云的概率

float sprinkle[2] = {0.1, 0.5};//sprinkle[0]表示在多云时候浇灌的概率，sprinkle[1]表示在没有多云时浇灌的概率

float rain[2] = {0.8, 0.2};//rain[0]表示在多云的时候下雨的概率，rain[1]表示在没有多云时下雨的概率

float wet[2][2];/\*wet[0][0]表示sprinkle且rain时wetgrass的概率；

wet[0][1]表示sprinkle且未下雨时wetgrass的概率；

wet[1][0]表示未浇灌且rain时wetgrass的概率；

wet[1][1]表示未浇灌且未下雨时wetgrass的概率；\*/

void print\_sample(bool \*a)//打印样本数据四元组

{

cout<<a[0]<<' '<<a[1]<<' '<<a[2]<<' '<<a[3]<<endl;

}

bool \*sample(int N)//样本随机产生函数

{

bool \*a = new bool[4];//预设样本数据四元组,a[0]表示是否多云，a[1]表示是否浇灌，a[2]表示是否下雨，a[3]表示是否草湿

if (random01() < cloudy)//如果随机数小于多云的概率则多云，否则不多云

{

a[0] = true;

}

else

{

a[0] = false;

}

if(a[0])//在多云的情况下判断是否浇灌和下雨

{

if (random01() < sprinkle[0])

{

a[1] = true;

}

else

{

a[1] = false;

}

if (random01() < rain[0])

{

a[2] =true;

}

else

{

a[2] = false;

}

}

else//在未多云的情况下判断

{

if (random01() < sprinkle[1])

{

a[1] = true;

}

else

{

a[1] = false;

}

if (random01() < rain[1])

{

a[2] =true;

}

else

{

a[2] = false;

}

}

if (a[1])//在浇灌的情况下

{

if(a[2])//下雨

{

if (random01() < wet[0][0])

{

a[3] = true;

}

else

{

a[3] = false;

}

}

else

{

if (random01() <wet[0][1])

{

a[3] = true;

}

else

{

a[3] = false;

}

}

}

else//在未浇灌的情况下

{

if(a[2])//下雨

{

if (random01() < wet[1][0])

{

a[3] = true;

}

else

{

a[3] = false;

}

}

else

{

if (random01() <wet[1][1])

{

a[3] = true;

}

else

{

a[3] = false;

}

}

}

//print\_sample(a);

return a;

}

int main()

{

srand((unsigned)time(NULL));

random01();

wet[0][0] = 0.99;

wet[0][1] = 0.9;

wet[1][0] = 0.9;

wet[1][1] = 0.0;

int N = 10000;

bool \*b = sample(N);

bool \*\*dataset = new bool \*[N];

int Cnum = 0;//统计多云的次数

int Snum = 0;//统计浇灌的次数

int Rnum = 0;//统计下雨的次数

int Wnum = 0;//统计草湿的次数

int SCnum = 0;//统计多云且浇灌的次数

int SNCnum = 0;//统计未多云且浇灌的次数

int RCnum = 0;//多云且下雨的次数

int RNCnum = 0;//未多云且下雨的次数

int SRnum = 0;//浇灌且下雨的次数

int SNRnum = 0;//浇灌且未下雨的次数

int NSRnum = 0;//未浇灌且下雨的次数

int NSNRnum = 0;//未浇灌且未下雨的次数

int WSRnum = 0;//草湿且浇灌且下雨的次数

int WNSRnum = 0;//草湿且未浇灌且下雨的次数

int WSNRnum = 0;//草湿且浇灌且未下雨的次数

int WNSNRnum = 0;//草湿且未浇灌且未下雨的次数

for (int i = 0;i < N;i++)//生成随机样本，并统计次数

{

dataset[i] = sample(N);

//print\_sample(dataset[i]);

if (dataset[i][0])

{

Cnum ++;

if (dataset[i][1])

SCnum ++;

if (dataset[i][2])

RCnum ++;

}

else

{

if (dataset[i][1])

SNCnum ++;

if (dataset[i][2])

RNCnum ++;

}

if (dataset[i][1])

Snum ++;

if (dataset[i][2])

Rnum ++;

if (dataset[i][3])

Wnum ++;

if (dataset[i][1])

{

if(dataset[i][2])

{

SRnum ++;

if(dataset[i][3])

WSRnum ++;

}

else

{

SNRnum ++;

if(dataset[i][3])

WSNRnum ++;

}

}

else

{

if(dataset[i][2])

{

NSRnum ++;

if(dataset[i][3])

WNSRnum ++;

}

else

{

NSNRnum ++;

if(dataset[i][3])

WNSNRnum ++;

}

}

}

cout<<"N:"<<N<<endl;

cout<<"Cloudy:"<<(float)Cnum/N<<endl;//打印各类概率的估计值

cout<<"Sprinkle | Cloudy:"<<(float)SCnum/Cnum<<endl;

cout<<"Sprinkle | !Cloudy:"<<(float)SNCnum/(N-Cnum)<<endl;

cout<<"Rain | Cloudy:"<<(float)RCnum/Cnum<<endl;

cout<<"Rain | !Cloudy:"<<(float)RNCnum/(N-Cnum)<<endl;

cout<<"WetGrass | Sprinkle && Rain:"<<(float)WSRnum/SRnum<<endl;

cout<<"WetGrass | Sprinkle && !Rain:"<<(float)WSNRnum/SNRnum<<endl;

cout<<"WetGrass | !Sprinkle && Rain:"<<(float)WNSRnum/NSRnum<<endl;

cout<<"WetGrass | !Sprinkle && !Rain:"<<(float)WNSNRnum/NSNRnum<<endl;

return 0;

}