

## E09 Variable Elimination

---

18340149 孙新梦

2020 年 11 月 11 日

### 目录

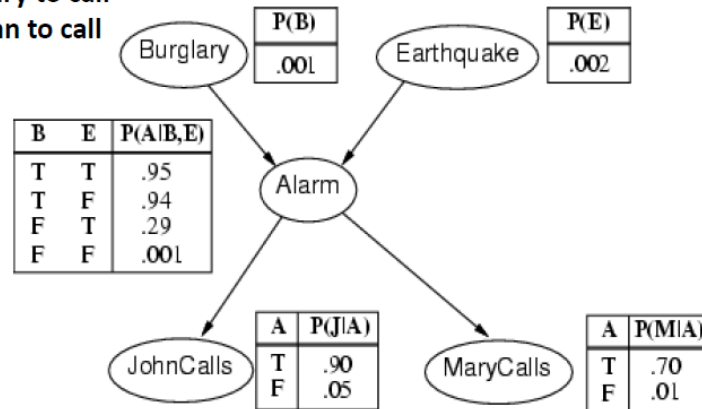
<b>1</b>	<b>VE</b>	<b>2</b>
<b>2</b>	<b>Task</b>	<b>5</b>
<b>3</b>	<b>Codes and Results</b>	<b>6</b>
<b>4</b>	<b>体会感想</b>	<b>12</b>

# 1 VE

The burglary example is described as following:

- A burglary can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Note that these tables only provide the probability that  $X_i$  is true.  
(E.g.,  $Pr(A \text{ is true} | B, E)$ )  
The probability that  $X_i$  is false is 1- these values



```
P(Alarm) =
0.002516442

P(J&&~M) =
0.050054875461

P(A | J&&~M) =
0.0135738893313

P(B | A) =
0.373551228282

P(B | J&&~M) =
0.0051298581334

P(J&&~M | ~B) =
0.049847949
```

Here is a VE template for you to solve the burglary example:

```
1 class VariableElimination:
2     @staticmethod
3     def inference(factorList, queryVariables,
4         orderedListOfHiddenVariables, evidenceList):
5         for ev in evidenceList:
6             #Your code here
7         for var in orderedListOfHiddenVariables:
8             #Your code here
```

```

9         print "RESULT:"
10        res = factorList[0]
11        for factor in factorList[1:]:
12            res = res.multiply(factor)
13        total = sum(res.cpt.values())
14        res.cpt = {k: v/total for k, v in res.cpt.items()}
15        res.printInf()
16
17        @staticmethod
18        def printFactors(factorList):
19            for factor in factorList:
20                factor.printInf()
21
22    class Util:
23        @staticmethod
24        def to_binary(num, len):
25            return format(num, '0' + str(len) + 'b')
26
27    class Node:
28        def __init__(self, name, var_list):
29            self.name = name
30            self.varList = var_list
31            self.cpt = {}
32
33        def setCpt(self, cpt):
34            self.cpt = cpt
35
36        def printInf(self):
37            print "Name_=" + self.name
38            print "_vars_" + str(self.varList)
39            for key in self.cpt:
40                print "___key:_" + key + "_val_:_" + str(self.cpt[key])
41
42            print ""
43
44        def multiply(self, factor):
45            """function that multiplies with another factor"""
46            #Your code here
47            new_node = Node("f" + str(newList), newList)

```

```

41         new_node.setCpt(new_cpt)
42         return new_node
43     def sumout(self, variable):
44         """function that sums out a variable given a factor"""
45         #Your code here
46         new_node = Node("f" + str(new_var_list), new_var_list)
47         new_node.setCpt(new_cpt)
48         return new_node
49     def restrict(self, variable, value):
50         """function that restricts a variable to some value
51         in a given factor"""
52         #Your code here
53         new_node = Node("f" + str(new_var_list), new_var_list)
54         new_node.setCpt(new_cpt)
55         return new_node
56 # create nodes for Bayes Net
57 B = Node("B", ["B"])
58 E = Node("E", ["E"])
59 A = Node("A", ["A", "B", "E"])
60 J = Node("J", ["J", "A"])
61 M = Node("M", ["M", "A"])
62
63 # Generate cpt for each node
64 B.setCpt({'0': 0.999, '1': 0.001})
65 E.setCpt({'0': 0.998, '1': 0.002})
66 A.setCpt({'111': 0.95, '011': 0.05, '110':0.94, '010':0.06,
67 '101':0.29, '001':0.71, '100':0.001, '000':0.999})
68 J.setCpt({'11': 0.9, '01': 0.1, '10': 0.05, '00': 0.95})
69 M.setCpt({'11': 0.7, '01': 0.3, '10': 0.01, '00': 0.99})
70
71 print "P(A) ⊥*****"
72 VariableElimination.inference([B,E,A,J,M], ['A'], ['B', 'E', 'J', '
M'], {})

```

73

74 `print "P(B⊥ | ⊥J⊥M) ⊥*****"`75 `VariableElimination.inference([B,E,A,J,M], ['B'], ['E','A'], {'J':1, 'M':0})`

## 2 Task

- You should implement 4 functions: `inference`, `multiply`, `sumout` and `restrict`. You can turn to Figure 1 and Figure 2 for help.
- Please hand in a file named `E09_YourNumber.pdf`, and send it to `ai_2020@foxmail.com`

### The VE Algorithm

Given a Bayes Net with CPTs  $F$ , query variable  $Q$ , evidence variables  $E$  (observed to have values  $e$ ), and remaining variables  $Z$ . Compute  $\Pr(Q|E)$

- Replace each factor  $f \in F$  that mentions a variable(s) in  $E$  with its restriction  $f_{E=e}$  (this might yield a "constant" factor)
- For each  $Z_j$  in the order given –eliminate  $Z_j \in Z$  as follows:
  - Let  $f_1, f_2, \dots, f_k$  be the factors in  $F$  that include  $Z_j$
  - Compute new factor  $g_j = \sum_{Z_j} f_1 \times f_2 \times \dots \times f_k$
  - Remove the factors  $f_i$  from  $F$  and add new factor  $g_j$  to  $F$
- The remaining factors refer only to the query variable  $Q$ . Take their product and normalize to produce  $\Pr(Q|E)$ .

### The Product of Two Factors

- Let  $f(\underline{X}, \underline{Y})$  &  $g(\underline{Y}, \underline{Z})$  be two factors with variables  $\underline{Y}$  in common
- The **product** of  $f$  and  $g$ , denoted  $h = f \times g$  (or sometimes just  $h = fg$ ), is defined:

$$h(\underline{X}, \underline{Y}, \underline{Z}) = f(\underline{X}, \underline{Y}) \times g(\underline{Y}, \underline{Z})$$

f(A,B)		g(B,C)		h(A,B,C)			
ab	0.9	bc	0.7	abc	0.63	ab <sup>~</sup> c	0.27
a <sup>~</sup> b	0.1	b <sup>~</sup> c	0.3	a <sup>~</sup> bc	0.08	a <sup>~</sup> b <sup>~</sup> c	0.02
<sup>~</sup> ab	0.4	<sup>~</sup> bc	0.8	<sup>~</sup> abc	0.28	<sup>~</sup> ab <sup>~</sup> c	0.12
<sup>~</sup> a <sup>~</sup> b	0.6	<sup>~</sup> b <sup>~</sup> c	0.2	<sup>~</sup> a <sup>~</sup> bc	0.48	<sup>~</sup> a <sup>~</sup> b <sup>~</sup> c	0.12

图 1: VE and Product

### Summing a Variable Out of a Factor

- Let  $f(X, \underline{Y})$  be a factor with variable  $X$  ( $\underline{Y}$  is a set)
- We **sum out** variable  $X$  from  $f$  to produce a new factor  $h = \sum_X f$ , which is defined:

$$h(\underline{Y}) = \sum_{X \in \text{Dom}(X)} f(X, \underline{Y})$$

f(A,B)		h(B)	
ab	0.9	b	1.3
a <sup>~</sup> b	0.1	<sup>~</sup> b	0.7
<sup>~</sup> ab	0.4		
<sup>~</sup> a <sup>~</sup> b	0.6		

No error in the table. Here  $f(A, B)$  is not  $P(A, B)$ , but  $P(B|A)$ .

### Restricting a Factor

- Let  $f(X, \underline{Y})$  be a factor with variable  $X$  ( $\underline{Y}$  is a set)
- We **restrict** factor  $f$  to  $X=a$  by setting  $X$  to the value  $a$  and "deleting" incompatible elements of  $f$ 's domain. Define  $h = f_{X=a}$  as:  $h(\underline{Y}) = f(a, \underline{Y})$

f(A,B)		h(B) = f <sub>A=a</sub>	
ab	0.9	b	0.9
a <sup>~</sup> b	0.1	<sup>~</sup> b	0.1
<sup>~</sup> ab	0.4		
<sup>~</sup> a <sup>~</sup> b	0.6		

图 2: Sumout and Restrict

### 3 Codes and Results

```
1 """ 孙新梦
2
3
4     18340149
5     AI E09
6 """
7
8
9 class VariableElimination:
10     @staticmethod
11     def inference(factorList, queryVariables,
12                  orderedListOfHiddenVariables, evidenceList):
13
14         # step1.restrict. 把每个因子里面含有证据的替换取值, 创建新的表
15         for ev in evidenceList: # 遍历证据
16             for factor in factorList: # 遍历所有因子找出含有证据的相关
17                                     # 因子
18                 if ev in factor.varList:
19                     # 使用函数替换变量的值并创建新的 restrictcpt
20                     factorList.append(factor.restrict(ev,
21                                                         evidenceList[ev]))
22                     # 删除原先的 factor
23                     factorList.remove(factor)
24
25         # step2.elimination 按照给定的顺序进行变量消除算法.
26         for var in orderedListOfHiddenVariables: # 消除顺序遍历变量
27             # 含有目标变量的要加入 eliminationList
28             eliminationList = list(filter(lambda factor: var in
29                                           factor.varList, factorList))
30
31         new_var = eliminationList[0]
32         for eli in eliminationList:
```

```

30         for factor in factorList:
31             if factor.name == eli.name:
32                 factorList.remove(factor)
33             # 第一个之后乘起来
34             if eli != eliminationList[0]:
35                 new_var = new_var.multiply(eli)
36             new_var = new_var.sumout(var)    #加和
37             factorList.append(new_var)
38
39     #计算结果
40     print("RESULT:")
41     res = factorList[0]
42     for factor in factorList[1:]:
43         res = res.multiply(factor)
44     total = sum(res.cpt.values())
45     res.cpt = {k: v / total for k, v in res.cpt.items()}
46     res.printInf()
47
48     @staticmethod
49     def printFactors(factorList):
50         for factor in factorList:
51             factor.printInf()
52
53
54 class Util:
55     @staticmethod
56     def to_binary(num, len):
57         return format(num, '0' + str(len) + 'b')
58
59
60 class Node:
61     def __init__(self, name, var_list):
62         self.name = name

```

```

63         self.varList = var_list
64         self.cpt = {}
65
66     def setCpt(self, cpt):
67         self.cpt = cpt
68
69     def printInf(self):
70         print("Name_=" + self.name)
71         print("_vars_" + str(self.varList))
72         for key in self.cpt:
73             print("_key:" + key + "_val:" + str(self.cpt[key]
74                 ))
75         print("")
76
77     def multiply(self, factor):
78         """function that multiplies with another factor"""
79         # Your code here
80         # 两个节点的表相乘
81         newlist = [var for var in self.varList]
82         new_cpt = {}
83
84         # 计算公共变量的索引
85         index1 = []
86         index2 = []
87         for var1 in self.varList:
88             for var2 in factor.varList:
89                 if var1 == var2:
90                     index1.append(self.varList.index(var1))
91                     index2.append(factor.varList.index(var2))
92                 else:
93                     newlist.append(var2)
94
95         for k1, v1 in self.cpt.items():

```



```

95         for k2, v2 in factor.cpt.items():
96             flag = True # 表示两个项能否做乘积，要求之前记录的公共部
                           分相同都为之类00
97             for i in range(len(index1)):
98                 if k1[index1[i]] != k2[index2[i]]:
99                     flag = False
100                     break
101             if flag:
102                 new_key = k1 # 以为蓝本，添加除了相同的项之外
                              的符号k1k2
103                 for i in range(len(k2)):
104                     if i in index2:
105                         continue
106                     new_key += k2[i]
107
108                 new_cpt[new_key] = v1 * v2
109
110     theList = []
111     for letter in newList:
112         theList.append(letter)
113     #print(theList)
114
115     #new_node = Node("f" + str(newList), newList) # 为两个节点
        的合并去重newlistlist
116     new_node = Node("f"+str(theList), theList)
117     new_node.setCpt(new_cpt) # 新的为两个产生的cptnewcpt
118
119     return new_node
120
121 def sumout(self, variable):
122     """function that sums out a variable given a factor"""
123     # Your code here
124     new_var_List = [var for var in self.varList]
125     new_var_List.remove(variable)

```

```

126         new_cpt = {}
127
128         # index需要被加和的遍历的下标:
129         index = self.varList.index(variable)
130
131         # 遍历字典的键值对, 把目标变量相同的加在一起cpt
132         for k, v in self.cpt.items():
133             # 没有记录过这一项新增上来,
134             if k[:index] + k[index + 1:] not in new_cpt.keys():
135                 new_cpt[k[:index] + k[index + 1:]] = v
136             else: # 有了加和在原来基础上
137                 new_cpt[k[:index] + k[index + 1:]] += v
138
139         theList = []
140         for letter in new_var_List:
141             theList.append(letter)
142         #print(theList)
143
144         new_node = Node("f" + str(theList), theList)
145         new_node.setCpt(new_cpt)
146         return new_node
147
148     def restrict(self, variable, value):
149         """function that restricts a variable to some value
150         in a given factor"""
151         # Your code here
152         new_var_List = [var for var in self.varList]
153         new_var_List.remove(variable)
154         new_cpt = {}
155
156         # index需要被限制的变量下标:
157         index = self.varList.index(variable)
158

```

```

159         # 把和相同的值放进新字典 valuevariable
160         for k, v in self.cpt.items():
161             if k[index] == str(value):
162                 new_cpt[k[:index] + k[index + 1:]] = v
163
164         theList = []
165         for letter in new_var_List:
166             theList.append(letter)
167         #print(theList)
168
169         new_node = Node("f" + str(theList), theList)
170         new_node.setCpt(new_cpt)
171         return new_node
172
173
174 # create nodes for Bayes Net
175 B = Node("B", ["B"])
176 E = Node("E", ["E"])
177 A = Node("A", ["A", "B", "E"])
178 J = Node("J", ["J", "A"])
179 M = Node("M", ["M", "A"])
180
181 # Generate cpt for each node
182 B.setCpt({'0': 0.999, '1': 0.001})
183 E.setCpt({'0': 0.998, '1': 0.002})
184 A.setCpt({'111': 0.95, '011': 0.05, '110': 0.94, '010': 0.06,
185           '101': 0.29, '001': 0.71, '100': 0.001, '000': 0.999})
186 J.setCpt({'11': 0.9, '01': 0.1, '10': 0.05, '00': 0.95})
187 M.setCpt({'11': 0.7, '01': 0.3, '10': 0.01, '00': 0.99})
188
189 print("P(A) ⊥*****")
190 VariableElimination.inference([B, E, A, J, M], ['A'], ['B', 'E', '
    J', 'M'], {})

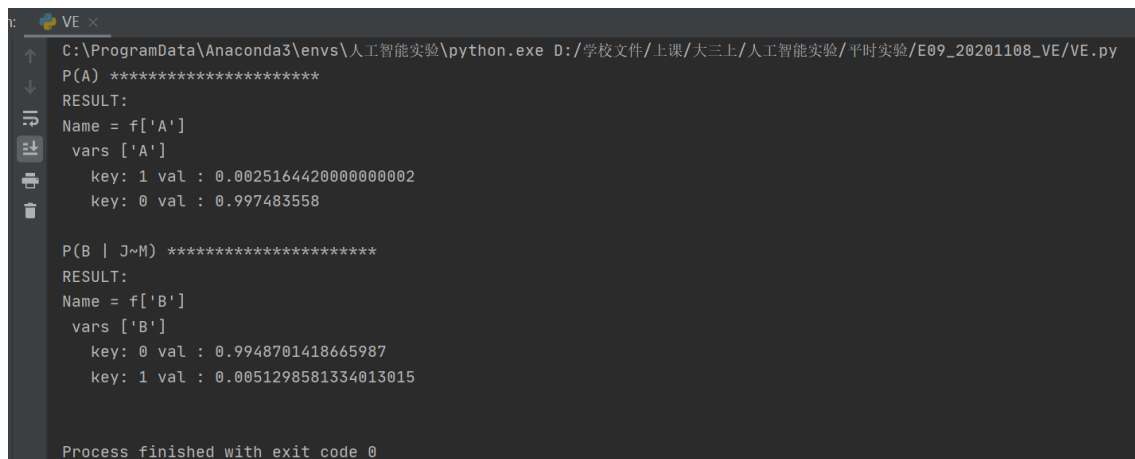
```

```

191
192 print("P(B⊥ |  $\neg$ J $\sim$ M)  $\neg$ *****")
193 VariableElimination.inference([B, E, A, J, M], ['B'], ['E', 'A'],
    {'J': 1, 'M': 0})

```

### 结果截图



```

C:\ProgramData\Anaconda3\envs\人工智能实验\python.exe D:/学校文件/上课/大三上/人工智能实验/平时实验/E09_20201108_VE/VE.py
P(A) *****
RESULT:
Name = f['A']
vars ['A']
key: 1 val : 0.0025164420000000002
key: 0 val : 0.997483558

P(B | J~M) *****
RESULT:
Name = f['B']
vars ['B']
key: 0 val : 0.9948701418665987
key: 1 val : 0.0051298581334013015

Process finished with exit code 0

```

## 4 体会感想

本次实验实现VE算法的过程，是对理论课的内容以及上次的理论作业一个更深刻的体会。VE算法在计算概率的时候非常常用，算法思想是首先把证据的值替换进去，之后按照给出的顺序逐步做乘积加和的消除变量，得到新的factor替换使用的factor，之后剩下的变量归一化就是所求概率。