School of Data and Computer Science, Sun Yat-sen University

E09 Variable Elimination

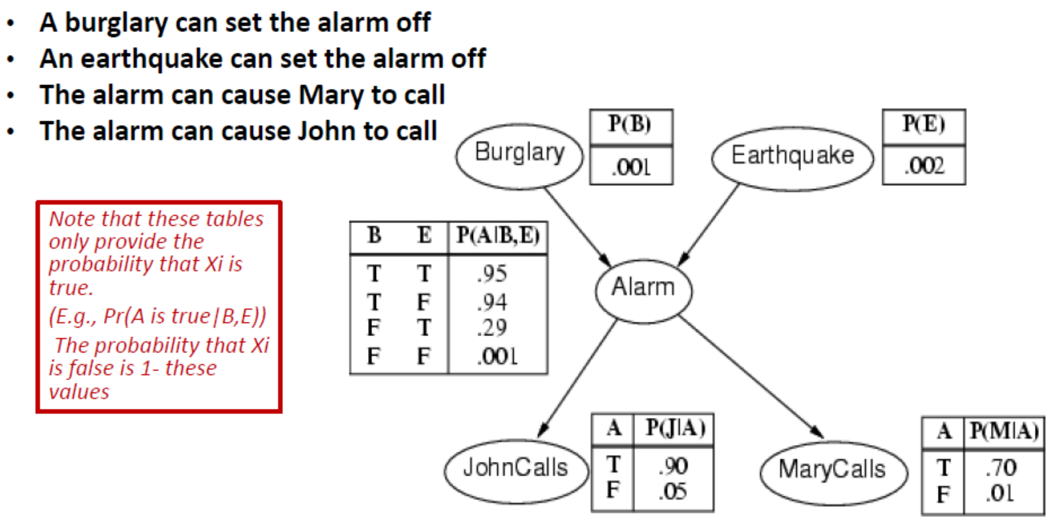
18340052 何泽

2020 年 11 月 13 日

目录

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 Xi is true.

(E.g., Pr(A is true|B,E))

The probability that Xi is false is 1- these values

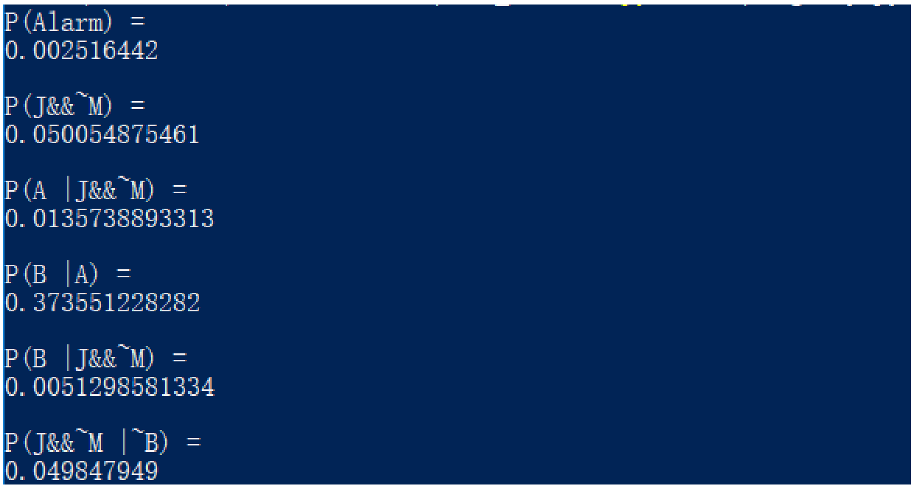
P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01



```
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:

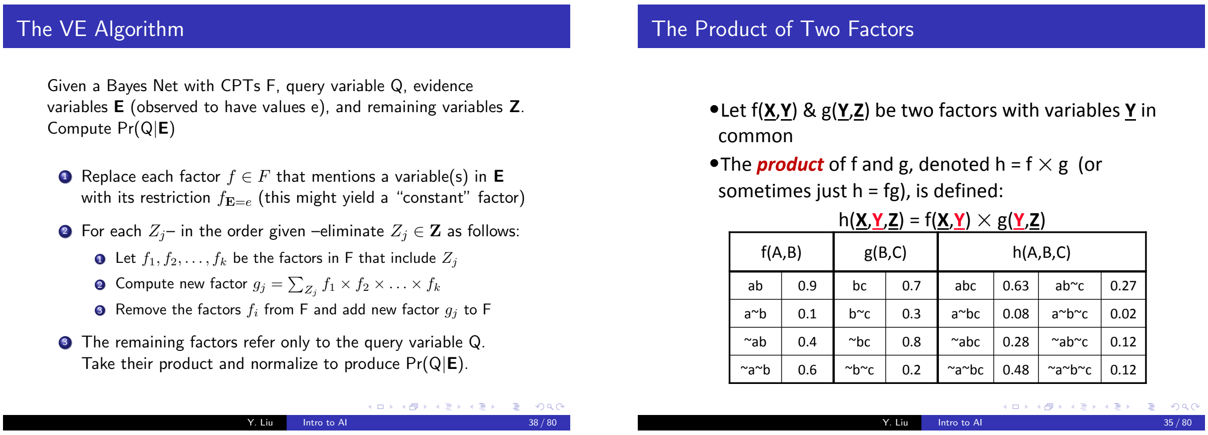
```
class VariableElimination:
    @staticmethod
    def inference(factorList, queryVariables,
    orderedListOfHiddenVariables, evidenceList):
        for ev in evidenceList:
            #Your code here
        for var in orderedListOfHiddenVariables:
            #Your code here
        print "RESULT:"
        res = factorList[0]
        for factor in factorList[1:]:
            res = res.multiply(factor)
        total = sum(res.cpt.values())
        res.cpt = {k: v/total for k, v in res.cpt.items()}
        res.printInf()
    @staticmethod
    def printFactors(factorList):
        for factor in factorList:
            factor.printInf()
class Util:
    @staticmethod
    def to_binary(num, len):
        return format(num, '0' + str(len) + 'b')
class Node:
    def __init__(self, name, var_list):
        self.name = name
        self.varList = var_list
        self.cpt = {}
    def setCpt(self, cpt):
        self.cpt = cpt
    def printInf(self):
        print "Name␣=␣" + self.name
        print "␣vars␣" + str(self.varList)
        for key in self.cpt:
            print "␣␣␣key:␣" + key + "␣val␣:␣" + str(self.cpt[key])
        print ""
    def multiply(self, factor):
        """function that multiplies with another factor"""
        #Your code here
        new_node = Node("f" + str(newList), newList)
        new_node.setCpt(new_cpt)
        return new_node
    def sumout(self, variable):
        """function that sums out a variable given a factor"""
        #Your code here
        new_node = Node("f" + str(new_var_list), new_var_list)
        new_node.setCpt(new_cpt)
        return new_node
    def restrict(self, variable, value):
        ”””function that restricts a variable to some value
        in a given factor”””
        #Your code here
        new_node = Node(”f” + str(new_var_list), new_var_list)
        new_node.setCpt(new_cpt)
        return new_node
# create nodes for Bayes Net
B = Node(”B”, [”B”])
E = Node(”E”, [”E”])
A = Node(”A”, [”A”, ”B”,”E”])
J = Node(”J”, [”J”, ”A”])
M = Node(”M”, [”M”, ”A”])

# Generate cpt for each node
B.setCpt({’0’: 0.999, ’1’: 0.001})
E.setCpt({’0’: 0.998, ’1’: 0.002})
A.setCpt({’111’: 0.95, ’011’: 0.05, ’110’:0.94,’010’:0.06,
’101’:0.29,’001’:0.71,’100’:0.001,’000’:0.999})
J.setCpt({’11’: 0.9, ’01’: 0.1, ’10’: 0.05, ’00’: 0.95})
M.setCpt({’11’: 0.7, ’01’: 0.3, ’10’: 0.01, ’00’: 0.99})

print ”P(A)␣**********************”
VariableElimination.inference([B,E,A,J,M], [’A’], [’B’, ’E’, ’J’,’M’], {})

print ”P(B␣|␣J~M)␣**********************”
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

3 Codes and Results

Code:



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**)

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

Y. Liu Intro to AI 38 / 80

The Product of Two Factors

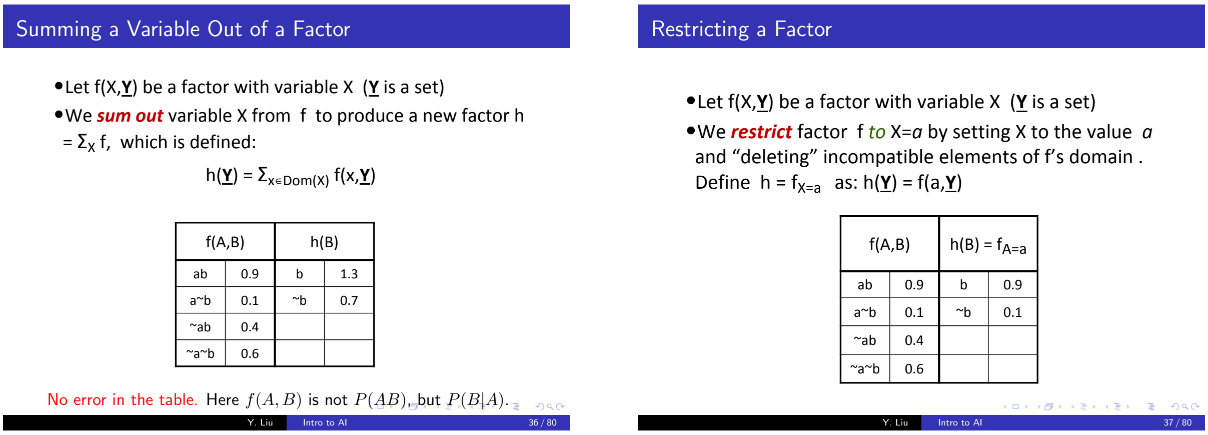
- Let f($\underline{\mathbf{X}}$,$\underline{\mathbf{Y}}$) & g($\underline{\mathbf{Y}}$,$\underline{\mathbf{Z}}$) be two factors with variables $\underline{\mathbf{Y}}$ in common
- The ***product*** of f and g, denoted h = f × g (or sometimes just h = fg), is defined:

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

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

Y. Liu Intro to AI 35 / 80

图 1: VE and Product



Summing a Variable Out of a Factor

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

$$h(\underline{Y}) = \Sigma_{x\in Dom(X)} f(x,\underline{Y})$$

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

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

Y. Liu Intro to AI 36 / 80

Restricting a Factor

- Let f(X,$\underline{\mathrm{Y}}$) be a factor with variable X ($\underline{\mathrm{Y}}$ 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($\underline{\mathrm{Y}}$) = f(a,$\underline{\mathrm{Y}}$)

f(A,B)		h(B) = $f_{A=a}$	
ab	0.9	b	0.9
a~b	0.1	~b	0.1
~ab	0.4		
~a~b	0.6		

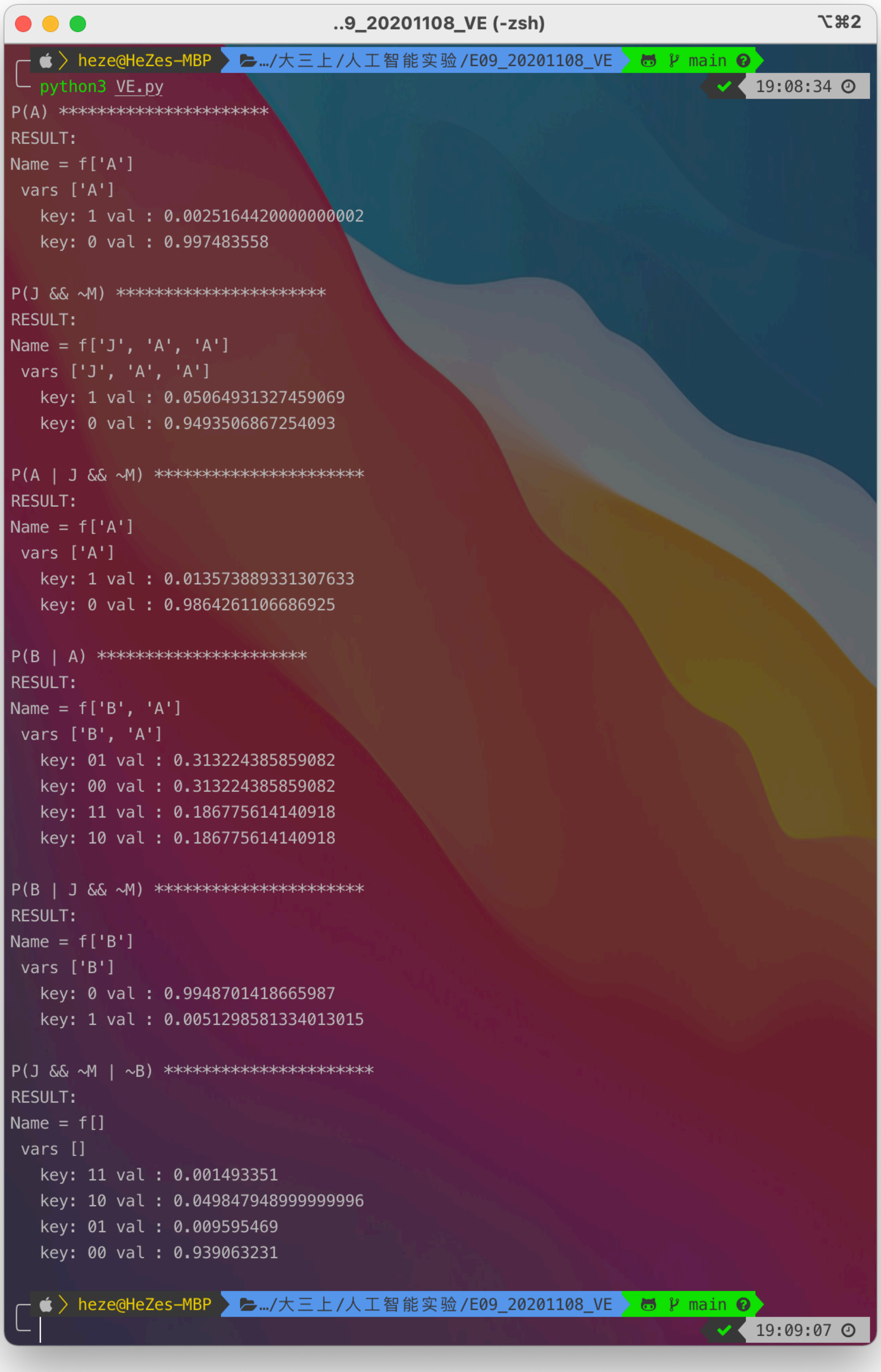
Y. Liu Intro to AI 37 / 80

图 2: Sumout and Restrict

```python
class VariableElimination:
    @staticmethod
    def inference(factorList, queryVariables,
                        orderdListOfHiddenVariables, evidenceList):
        for ev in evidenceList:
            for factor in factorList:
                if ev in factor.varList:
                    factorList.append(factor.restrict(ev, evidenceList[ev]))
                    factorList.remove(factor)

        for var in orderdListOfHiddenVariables:
            eliminationList = list(filter(lambda factor: var in factor.varList,factorList))
            new_var = eliminationList[0]
            for e in eliminationList:
                for i in factorList:
                    if i.name == e.name:
                        factorList.remove(i)
                if not e == eliminationList[0]:
                    new_var = new_var.multiply(e)
            new_var = new_var.sumout(var)
            factorList.append(new_var)
        print("RESULT: ")
        res = factorList[0]
        for factor in factorList[1:]:
            res = res.multiply(factor)
        total = sum(res.cpt.values())
        res.cpt = {k: v/total for k, v in res.cpt.items()}
        res.printInf()

    @staticmethod
    def printFactors(factorList):
        for factor in factorList:
            factor.printInf()


class Util:
    @staticmethod
    def to_binary(num, len):
        return format(num, '0' + str(len) + 'b')


class Node:
    def __init__(self, name, var_list):
        self.name = name
        self.varList = var_list
        self.cpt = {}

    def setCpt(self, cpt):
        self.cpt = cpt

    def printInf(self):
        print("Name = " + self.name)
        print(" vars  " + str(self.varList))
        for key in self.cpt:
            print("   key:  " + key + " val :  " + str(self.cpt[key]))
        print()

    def multiply(self, factor):
        newList = [var for var in self.varList]
        new_cpt = {}

        idx1 = []
        idx2 = []
        for var1 in self.varList:
            for var2 in factor.varList:
                if var1 == var2:
                    idx1.append(self.varList.index(var1))
                    idx2.append(factor.varList.index(var2))
                else:
                    newList.append(var2)

        for k1, v1 in self.cpt.items():
            for k2, v2 in factor.cpt.items():
                flag = True
                for i in range(len(idx1)):
                    if k1[idx1[i]] != k2[idx2[i]]:
                        flag = False
                        break
                if flag:
                    new_key = k1
                    for i in range(len(k2)):
                        if i in idx2: continue
                        new_key += k2[i]
                    new_cpt[new_key] = v1 * v2
        new_node = Node("f" + str(newList), newList)
        new_node.setCpt(new_cpt)
        return new_node

    def sumout(self, variable):
        new_var_list = [var for var in self.varList]
        new_var_list.remove(variable)
        new_cpt = {}
        idx = self.varList.index(variable)
        for k, v in self.cpt.items():
            if k[:idx] + k[idx+1:] not in new_cpt.keys():
                new_cpt[k[:idx] + k[idx+1:]] = v
            else: new_cpt[k[:idx] + k[idx+1:]] += v
        new_node = Node("f" + str(new_var_list), new_var_list)
        new_node.setCpt(new_cpt)
        return new_node

    def restrict(self, variable, value):
        new_var_list = [i for i in self.varList]
        new_var_list.remove(variable)
        new_cpt = {}
        idx = self.varList.index(variable)
        for k, v in self.cpt.items():
            if k[idx] == str(value):
                new_cpt[k[:idx] + k[idx+1:]] = v
        new_node = Node("f" + str(new_var_list), new_var_list)
        new_node.setCpt(new_cpt)
        return new_node

B = Node("B", ["B"])
E = Node("E", ["E"])
A = Node("A", ["A", "B", "E"])
J = Node("J", ["J", "A"])
M = Node("M", ["M", "A"])

B.setCpt({'0': 0.999, '1': 0.001})
E.setCpt({'0': 0.998, '1': 0.002})
A.setCpt({'111': 0.95, '011': 0.05,'110': 0.94, '010': 0.06,
          '101': 0.29, '001': 0.71,'100': 0.001, '000': 0.999})
J.setCpt({'11': 0.9, '01': 0.1,'10': 0.05, '00': 0.95})
M.setCpt({'11': 0.7, '01': 0.3,'10': 0.01, '00': 0.99})

```

```
print(”P(A) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’A’],
          [’B’, ’E’, ’J’, ’M’],{})

print(”P(J && ~M) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’J’],
          [’B’, ’E’, ’A’],{’M’: 0})

print(”P(A | J && ~M) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’A’],
          [’B’, ’E’],{’J’: 1, ’M’: 0})

print(”P(B | A) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’B’],
          [’E’, ’J’, ’M’],{’A’: 1})

print(”P(B | J && ~M) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’B’],
          [’E’, ’A’],{’J’: 1, ’M’: 0})

print(”P(J && ~M | ~B) **********************”)
VariableElimination.inference(
          [B, E, A, J, M], [’J’, ’M’],
          [’E’, ’A’],{’B’: 0})
```

Result:



```
 > heze@HeZes-MBP  …/大三上/人工智能实验/E09_20201108_VE   main ?
└ python3 VE.py                                               ✓  19:08:34
P(A) **********************
RESULT:
Name = f['A']
 vars ['A']
   key: 1 val : 0.0025164420000000002
   key: 0 val : 0.997483558

P(J && ~M) **********************
RESULT:
Name = f['J', 'A', 'A']
 vars ['J', 'A', 'A']
   key: 1 val : 0.05064931327459069
   key: 0 val : 0.9493506867254093

P(A | J && ~M) **********************
RESULT:
Name = f['A']
 vars ['A']
   key: 1 val : 0.013573889331307633
   key: 0 val : 0.9864261106686925

P(B | A) **********************
RESULT:
Name = f['B', 'A']
 vars ['B', 'A']
   key: 01 val : 0.313224385859082
   key: 00 val : 0.313224385859082
   key: 11 val : 0.186775614140918
   key: 10 val : 0.186775614140918

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

P(J && ~M | ~B) **********************
RESULT:
Name = f[]
 vars []
   key: 11 val : 0.001493351
   key: 10 val : 0.049847948999999996
   key: 01 val : 0.009595469
   key: 00 val : 0.939063231

 > heze@HeZes-MBP  …/大三上/人工智能实验/E09_20201108_VE   main ?
└ |                                                           ✓  19:09:07
```