Projeto Inteligência Artificial(3º Ano , 1º Semestre 2018/2019)

86411 - Filipe dos Santos Oliveira Marques

December 4, 2018

1 Parte 1

Nesta secção vamos analisar a solução produzida para a primeira parte do projeto - Inferência Exata em Redes Bayesianas.

1.1 Análise dos Resultados

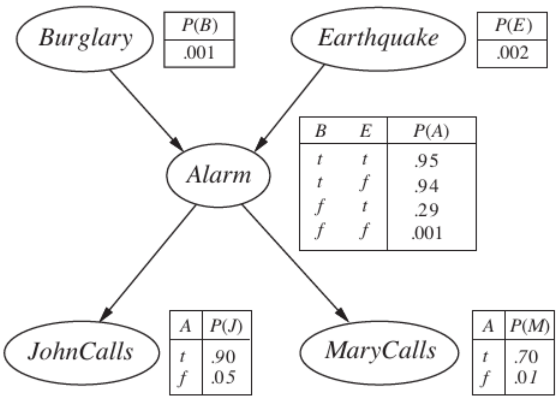

Figure 1: Bayesian Network

A Figura 1 mostra a rede bayesiana utilizada para testar o algoritmos produzidos.
Os valores pedidos no enunciado foram todos calculados com sucesso e os resultados das *queries* são:

- $P(B|j = t, m = t) = 0.2842$
- $P(E|j = t, m = t) = 0.176$
- $P(J|a = t, e = f) = 0.900$

1.2 Implementação

1.2.1 Probabilidade Conjunta

Para calcular a probabilidade conjunta temos de ter em conta algumas asserções em redes Bayesianas, nomeadamente que:

- Trata-se de uma rede acíclica;
- Cada nó é independente dos seus nós não descentes dado os seus predecessores imediatos(*parents*);

Sabendo isto, podemos definir a probabilidade conjunta:

$$P(y_1, ..., y_n) = \prod_{i=1}^{n} P(y_i|Parents(y_i))$$

Onde $Y = \{y_1, ..., y_n\}$ representa o conjunto de variáveis na rede Bayesiana.
Em Python teriamos:

```
# class BN
def computeJointProb(self, evid):
  p = 1
  for i in range(len(self.prob)):
    p*=self.prob[i].computeProb(evid)[evid[i]]
  return p
```

1.2.2 Probabilidade Posterior

Para calcular a probabilidade posterior podes usar a probabilidade condicional. Por exemplo, para calcular a probabilidade de haver um *Burglar* sabendo que *JohnCalls* e *MaryCalls* temos:

$$P(B|j = t, m = t) = \frac{P(B, j, m)}{P(j, m)} = \alpha P(B, j, m)$$

Para calcular a probabilidade $P(B, j, m)$ temos somar as probabilidades conjuntas para todos os valores de 'e' e 'a' onde $j = t$ e $m = t$.
Assim:

$$P(B, j, m) = \sum_e \sum_a P(B, j, m, e, a)$$

Tendo conhecimento da rede e das condições de independência podemos reescrever a segunda parte da equação como:

$$\sum_e \sum_a P(B)P(j|A)P(m|A)P(e)P(A|B,e)$$

Agrupando os fatores temos que:

$$P(B)\sum_e P(e)\sum_a P(A|B,e)P(m|A)P(j|A)$$

Esta abordagem de calcular a probabilidade posterior é chamada de enumeração. Para calcular a probabilidade posterior usamos o algoritmo *Enumeration-Ask*[pag.525 - AIMA]

```
# Enumeration Ask
def enumerationAsk(X, e, bn):
  Q = [0, 0]
  for xi in [0, 1]:
    e_xi = e.copy()
    Q[xi] = enumerateAll(bn.getVars(),
        extend(e_xi, X, xi), bn)
  return [Q(0)/sum(Q), Q(1)/sum(Q)]

def enumerateAll(vars, e, bn):
  if not vars: return 1
  Y, node = vars[0], bn.getNode(Y)
  rest = vars[1:]
  if isinstance(e[y], int):
    prob = node.computeProb(e)[e[Y]]
    return prob*enumerateAll(rest, e, bn)
  else:
    sumation = 0
    for y in [0, 1]:
      e_y = e.copy()
      prob = node.computeProb(e)[y]
      sumation += prob*enumerateAll(rest,
          extend(e_y, Y, y), bn)
    return sumation
```

1.3 Complexidade Computacional

1.3.1 Probabilidade Conjunta

Na tabela de probabilidade conjunta para um valor X com k *parents* tem 2^k combinações de valores dos nós pais. E para cada probabilidade apenas é necessário armazenar o valor onde $X = true$(pois $X = false$ é apenas $1 - valor\ de\ true$) Assim a complexidade para a tabela completa de probabilidade conjunta é:

$O(2^n)$ Onde n é o numero de nós na rede

1.3.2 Probabilidade Posterior

2 Parte 2

Nesta secção vamos analisar a solução produzida para a segunda parte do projeto - Aprendizagem por Reforço.