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

86411 - Filipe dos Santos Oliveira Marques

December 7, 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

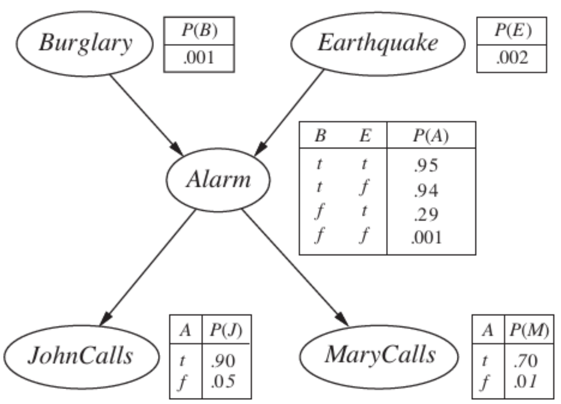

Figure 1: Bayesian Network

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.

- **Desvantagens desta abordagem:**
  - Tamanho das tabelas de probabilidade conjunta é exponencial $O(2^n)$.

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 de 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*.

- **Vantagens desta abordagem:**
  - Permite reduzir o custo de calcular as probabilidades fase à abordagem da probabilidade conjunta.
- **Limitações:**
  - Esta abordagem não é ótima visto alguns valores serem calculados várias vezes ao longo da computação da probabilidade posterior.

1.3 Complexidade Computacional

1.3.1 Probabilidade Conjunta

Um nó X_i com k nós pais vai ter 2^k linhas na sua tabela de probabilidade condicional.

Cada linha vai guardar um valor p para $X_i = t$. Assim para uma rede onde cada nó não tem mais de k nós pais, vamos precisar de $O(n2^k)$ números. Ao seja, cresce ***linearmente*** com n (o número de nós na rede).

1.3.2 Probabilidade Posterior

Sabendo que a rede em estudo é ***singly connected***[1], *a complexidade espacial e temporal de inferência exata é linear com o tamanho da rede.*[2]

Uma possível alternativa a este método seria: *Variable Elimination* que permitiria fazer os cálculos uma vez e guardar para usar mais tarde quando forem necessários, melhorando assim substancialmente o algoritmo de enumeração.

[1] Há no máximo uma ligação entre dois nós na rede.
[2] AIMA, pag.528, cap.14.4.3.

2 Parte 2

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

2.1 Ambiente do Agente

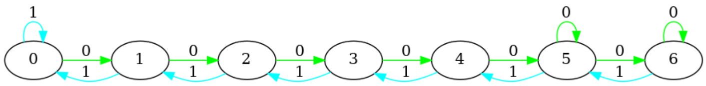

Figure 2: Ambiente 1

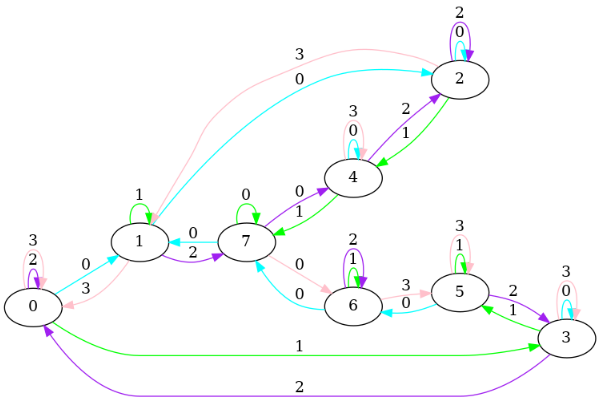

Figure 3: Ambiente 2

2.2 Função de Recompensa

2.2.1 Ambiente 1

Para os estados $\{0, 6\}$ observamos que a recompensa toma sempre o valor "1", para qualquer que seja a ação [0, 1]. Os restantes estados $\{1, 2, 3, 4, 5\}$, para qualquer ação [0, 1], o agente recebe sempre o valor "0".

2.2.2 Ambiente 2

Para o estado $\{7\}$ observamos que a recompensa toma sempre o valor "0", para qualquer que seja a ação [0, 1, 2, 3]. Os restantes estados $\{0, 1, 2, 3, 4, 5, 6\}$, para qualquer ação [0, 1, 2, 3], o agente recebe sempre o valor "-1".

2.3 Política Ótima

Após a função Q ter convergido a política ótima pode ser obtida pela formula:

$$\pi^*(s) = max_{a'}Q^*(s, a')$$

Ou seja, num estado s é preferível escolher a ação a' para qual o valor da função Q é maior, a este método chamamos de *Exploitation*.

2.4 Movimento do Agente

2.4.1 Ambiente 1

Neste Ambiente o movimento do agente é simples:

- **Estado 0**:
  - **Ação 0**: Avançar para o estado 1.
  - **Ação 1**: Ficar no mesmo estado.
- **Estado** $\{1, 2, 3, 4\}$:
  - **Ação 0**: Avançar para o próximo estado.
  - **Ação 1**: Recuar para o estado anterior.
- **Estado 5**:
  - **Ação 0**: Ficar no mesmo estado ***OU*** avançar para o estado 6. (Escolha não determinista).
  - **Ação 1**: Recuar para o estado anterior.
- **Estado 6**:
  - **Ação 0**: Ficar no mesmo estado.
  - **Ação 1**: Recuar para o estado anterior

2.4.2 Ambiente 2

Sendo o movimento do deste ambiente mais complexo passo a descrever o movimento mais comum que pode ser executado em cada nó.

- Para cada estado o agente possui **4 ações** possíveis:
  - Por norma o agente possui sempre duas ações em que pode ficar no mesmo estado(Sendo o estado 1 a exceção a esta regra).
  - Em cada estado o agente possui uma ação para avançar e outra para recuar para um estado anterior.
    * Sendo exceção o estado "0" que tem duas ações para avançar e o estado "7" que tem três ações para recuar.

2.5 Análise dos Resultados

O algoritmo implementado consegui passar aos testes disponibilizados com uma taxa de aprendizagem $\alpha = 0.1$.

2.6 Implementação

Nesta secção vamos discutir a implementação das duas funções pedidas.

2.6.1 traces2Q

Para esta função foi feito um ciclo que corria até a função Q convergir. Servindo-nos da formula da função Q:

$$Q[s, a] = \\ Q[s, a] + \alpha(R + \gamma * argmax_{a'}Q[s, a'] - Q[s, a])$$

Cada entrada da função *Q[s, a]* foi preenchida e ao mesmo tempo calculada a normal da diferença entre a matrix da função *Q* atual e da anterior, caso a diferença fosse menor que 0.00001 a função *traces2Q* terminava pois considerámos que nessa altura a função *Q* já convergiu.

2.6.2 policy

Esta função foi simples de implementar visto apenas termos de escolher uma ação a para o estado atual x. A escolha da ação depende também da política escolhida:

- ***Exploitation***:
  - Escolhemos a ação a para qual *Q[x, a]* é maximizado.
- ***Exploration***:
  - Escolhemos uma ação *random* possível no estado x.