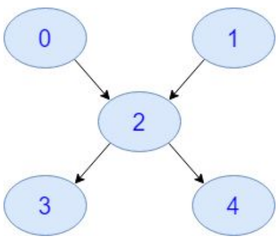
Grupo: 035
Bernardo Esteves 87633
Francisco Barata 87656

Redes Bayesianas

Utilizamos um grafo para definir a Rede Bayesiana, da seguinte forma:

```
gra = [[],[],[0,1],[2],[2]]
```



Com uma lista em que cada elemento (nó) representa os pais de cada uma das variáveis.

As probabilidades do nó em relação aos pais é dado por um array com um número de dimensões equivalente ao número de pais, aonde cada dimensão corresponde ao valor do pai. No caso de não conter pai, o array é de rank 1 só com 1 valor.

Descrição métodos implementados:

- O método **computeProb** calcula a probabilidade do nó ser *false* e de ser *true*, tendo em conta a evidência dos nós pai.

- O método **computeJointProb** que dado uma evidência (sem valores desconhecidos) permite determinar a probabilidade de um acontecimento. A evidência representa o valor de cada variável aleatória (**0** - false; **1** - true).

A Joint Probability é calculada através da fórmula:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$

ou seja, a Joint Probability resulta do produtório das probabilidades condicionais de cada variável consoante os seus pais.

Uma característica da Joint Probability é a soma das probabilidades de todos os acontecimentos possíveis num determinado cenário para todas combinações existentes de uma evidência ser igual a 1.

- O método **computePostProb** calcula a probabilidade a-posteriori de uma variável dada uma evidência.

Utilizando como notação na evidência (**0** - false; **1** - true; **[ ]** para indicar desconhecido; **-1** para indicar a variável que se quer calcular a-posteriori).

As variáveis desconhecidas (**[ ]**) podem assumir tanto o valor *true* ou *false*, sendo por isso necessário considerar diferentes evidências para todas as combinações de valores dessas variáveis.

A Post Probability é um somatório de Joint Probabilities, em que cada Joint Probability é calculada para uma das várias evidências.

Este método permite induzir a probabilidade de um determinado acontecimento através da soma dos produtos das probabilidades condicionais. Uma vantagem deste método é conseguir determinar uma probabilidade à custa de outras probabilidades presentes na Rede Bayesiana.

Discussão da complexidade computacional e possíveis métodos alternativos:

Existem outros métodos que permitem que o cálculo das probabilidades seja mais eficiente. Poderíamos utilizar o **método da eliminação das variáveis**, eliminando as variáveis irrelevantes.

Complexidade **computeProb**:

$O(\textit{número de pais por nó})$

Complexidade **computePostProb**:

$O((\textit{número nós}) * (\textit{número pais por nó}) * 2^{\textit{variáveis desconhecidas}})$

Complexidade **computeJointProb:**

$O((\textit{número de nós}) * (\textit{número de pais por nó}))$

Descrição crítica dos resultados pedido

Os resultados obtidos estão conforme o esperado, como podemos verificar pela soma dos joint probabilities dar 1 e o cálculo das

Grupo: 035
Bernardo Esteves 87633
Francisco Barata 87656

probabilidades com variáveis desconhecidas estarem dentro dos parâmetros.

Aprendizagem por Reforço

Implementação do algoritmo Q-learning dado uma lista de trajetórias. Este algoritmo aprende os valores de Q através da equação:

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

Funções implementadas:

Traces2Q

Função que obtém a matriz de valores Q através do algoritmo de Q-Learning. Recebe uma lista de steps do agente com o estado atual, ação realizada, o estado alcançado e a recompensa pela ação.

Policy

Função que retorna uma ação conforme um dado estado e matriz de valores Q. Pode ser executado em modo exploration que corresponde a devolver uma ação aleatória, ou em modo exploitation que corresponde a executar a melhor ação conforme o valor Q do estado.
Recebe um estado, uma lista de parâmetros que corresponde a uma matriz de valores Q e o tipo de policy.

Implementação

No nosso projeto escolhemos o **learning rate** de 0.7 e o **número de samples** de 100000, estes foram os valores que nos parecem convergir mais rápido e corretamente para uma solução.

Política Óptima

Na presença de uma matriz de valores Q optima para um dado problema (Q*), podemos obter uma politica óptima (π*) ao selecionar a ação que proporciona o maior valor de Q conforme um dado estado.

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad \forall s \in \mathbb{S}$$

Função de recompensa

A função de recompensa R, dado um estado inicial **s** e uma ação **a**, devolve o valor da recompensa.

Ambiente 1:

R(s,a) -> $1 \; se \; s = 6 \; \vee s = 0$

$0 \; caso \; contrário$

Ambiente 2:

R(s,a) -> $0 \; se \; s = 7$

$-1 \; caso \; contrário$

Movimento do Agente:

O agente pode explorar o mundo ao executar ações aleatórias. Após a aprendizagem, o agente toma uma ação/move-se através de uma política e uma matriz de estados Q aprendida.

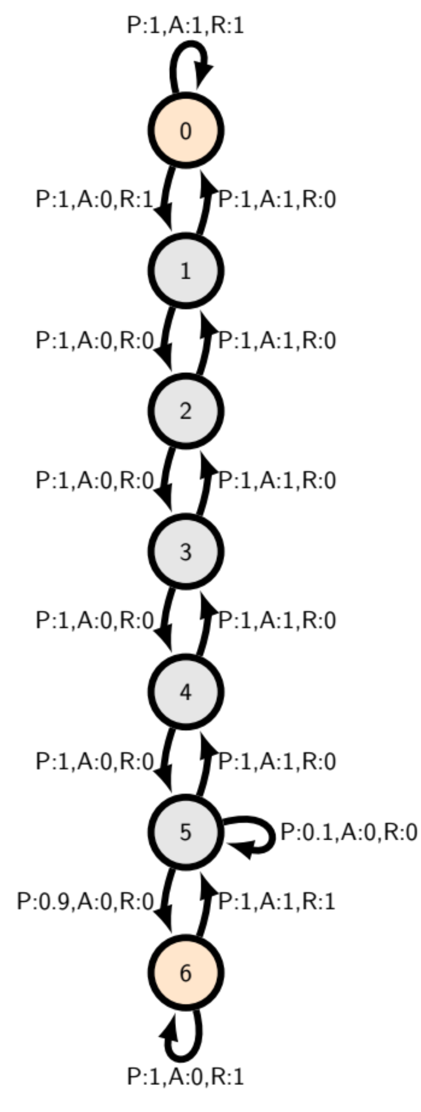
Podemos saber a probabilidade da transição para outro estado conforme uma ação aplicada a um estado através da representação gráfico do ambiente.

Representação gráfica dos ambientes:

O estado final esta representado a laranja.

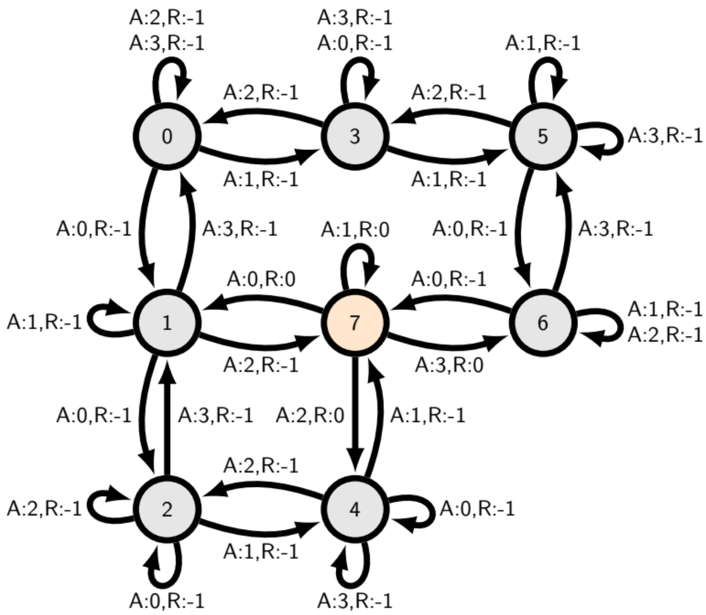
Ambiente 1:

(à direita)



Grupo: 035
Bernardo Esteves 87633
Francisco Barata 87656

Ambiente 2:



(Todos as transições representadas neste ambiente têm probabilidade de 1)

Vantagens

O agente aprende com um algoritmo relativamente fácil de implementar.

Desvantagens/Limitações

Para execução do Q-Learning é necessário saber o número de **estados** e de **ações** possíveis para cada estado, e a habilidade de descobrir as **funções de transição** entre estados e as suas **recompensas** conforme as ações. Tal como ainda é necessário definir um **valor de desconto** para as rewards (gamma) e um **learning rate** (alpha) para o algoritmo convergir corretamente. Do mesmo modo deve ser possível realizar um grande número de ações no ambiente para o agente poder generalizar.

Descrição crítica dos resultados pedidos

Os resultados obtidos estão conforme o esperado.

Os valores obtidos foram:

Ambiente 1:

Matriz Q:

```
[ 9.1        , 10.        ]
[ 7.29       ,  9.        ]
[ 6.561      ,  8.1       ]
[ 7.25808552,  7.29       ]
[ 8.08746193,  6.561      ]
[ 8.9984691 ,  7.22248731]
[10.        ,  9.0785674 ]
```

policy:

```
[1, 1, 1, 1, 0, 0, 0]
```

Ambiente 2:

Matriz Q:

```
[-1.9       , -3.41954785, -2.70999996, -2.71      ]
[-2.71      , -1.9       , -1.        , -2.71      ]
[-2.71      , -1.9       , -2.71      , -1.9       ]
[-3.43275505, -2.71      , -2.70999997, -3.43255233]
[-1.9       , -1.        , -2.71      , -1.9       ]
[-1.9       , -2.71      , -3.43898569, -2.71      ]
[-1.        , -1.9       , -1.9       , -2.71      ]
[-0.9       ,  0.        , -0.9       , -0.9       ]
```

policy: `[0, 2, 1, 2, 1, 0, 0, 1]`

O que se podia melhorar

Uma das coisas que se podia melhorar é fazer exploração inicial mais eficiente. Algumas implementações que podiam ser consideradas são o *Boltzmann exploration* e o ϵ-greedy.