Exercício RDL

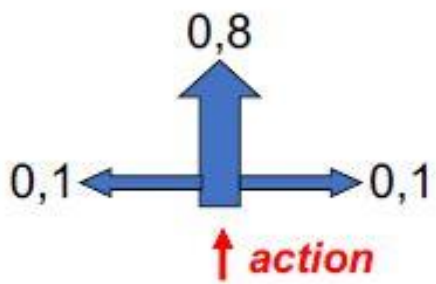
Filipe Assis Mourão

8988914

1): How to reach the goal after 12 steps (avoiding (2,4))? If the agent hits a wall, it stays in the same grid cell.

-0,04	-0,04	-0,04	+1 *goal*
-0,04		-0,04	−1
-0,04 *start*	-0,04	-0,04	-0,04

Transition model:



Provide:

(a) Sequence of actions;

(b) History h;

(c) Probability of h (show the calculation);

(d) Utility of h (show the calculation).

a) É possível chegar ao destino desejado com apenas 5 passos, o que maximiza à utilidade. Como nos foi pedido 12 passos, será escolhido uma sequência que engloba os 5 passos desejados:

[R,R,U,U,L,L,R,L,R,R,U,R]

b) Dado à sequência de ações, podemos agora escrever à história:

(1,1),R, (1,2),R, (1,3),U, (2,3),U, (3,3),L, (3,2),L, (3,1),R, (3,2),L, (3,1),R, (3,2),R, (3,3),U, (3,3),R, (3,4)

c) Para o cálculo da probabilidade dessa historia, será considerado que o robô seguirá todas as instruções estabelecidas com probabilidade 0.8:

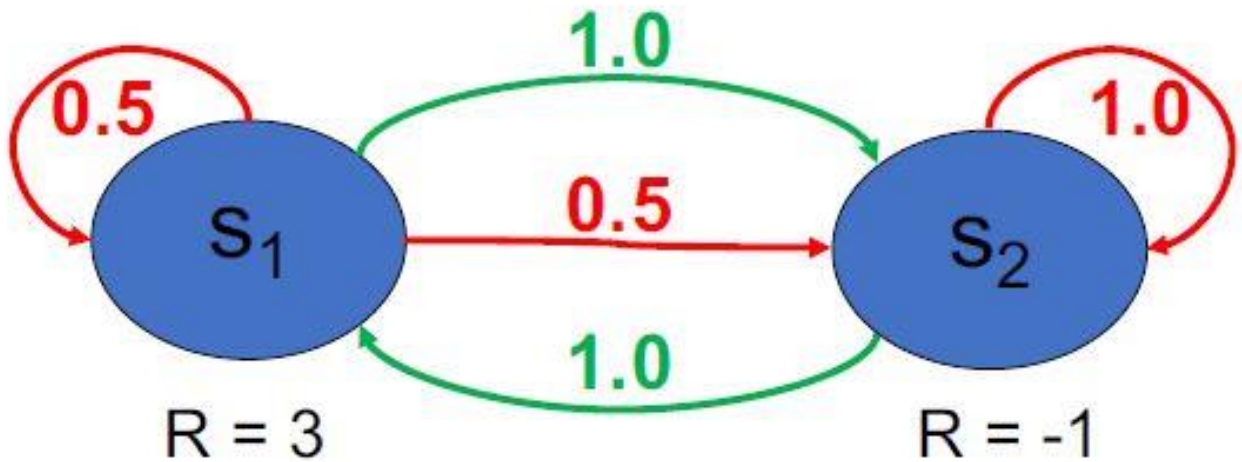
$$P = (0.8)^{12} = 0.0687 \cong 6{,}9\%$$

d) Para o cálculo da utilidade sabemos que o robô terá penalizações nos 12 movimentos que irá executar, além da recompensa ao chegar no destino:

$$U(h) = 12 * -0.04 + 1 = 0.52$$

2): Consider the Value Iteration procedure (Implement and / or show the code used and describe how it differs from that basic given in class). For the values and entries given below, provide the values V(s) at each iteration of VI.

- $\gamma = 0.5$, A={red, green}, S={s_1, s_2}



Devemos escolher um critério de parada, o critério escolhido foi:

$$\frac{(V^{t+1} * V^{t+1}) - (V^{t} * V^{t})}{(V^{t} * V^{t})} < 0.001$$

Código python utilizado:

```python
import numpy as np
```

```python
# Actions and respectivy probabilities
a11_p11 = 0.5; a11_p12 = 0.5; a12_p11 = 0.5;a12_p12 = 0.5
a21_p21 = 1.; a21_p22 = 0; a22_p21 = 0;a22_p22 = 1.
# Rewards
r1 = 3.; r2 = -1.
# Gamma
gamma = 0.5
```

```python
# check stop criteria
def  checkStopCriteria(v1,v0):
    stopCriteria = np.abs((np.inner(v1,v1) - np.inner(v0,v0) )/(np.inner(v0,v0)))
    if   stopCriteria < 0.0001 :
        return True
    else:
        return False
```

```python
# Initial states
v1 = np.array([0.,0.])
v0 = np.array([-1.,-1.])
iteration = 0
while not checkStopCriteria(v1,v0):
    # Calculate new state
    print('------------------------------------------------')
    iteration += 1
    v0 = np.copy(v1)
    v1[0] = r1 + gamma*np.maximum( a12_p11* (v0[0]) + a12_p12*(v0[1]), a11_p11*(v0[0]) + a11_p12*(v0[1]) )
    v1[1] = r2 + gamma*np.maximum( a21_p21* (v0[0]) +a21_p22*(v0[1]), a22_p21*(v0[1]) + a22_p22*(v0[1]))
    print(f'Iteration {iteration}\n state 1 = {v1[0]}\n state 2 = {v1[1]} ')
```

Iterações do algoritmo

	Estado 1	Estado 2
Iteração 1	3.0	-1.0
Iteração 2	3.5	0.5
Iteração 3	4.0	0.75
Iteração 4	4.1875	1.0
Iteração 5	4.296	1.093
Iteração 8	4.386	1.187
Iteração 12	4.4	1.2