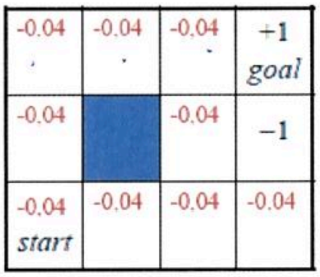
PCS5024 - Aprendizado Estatístico

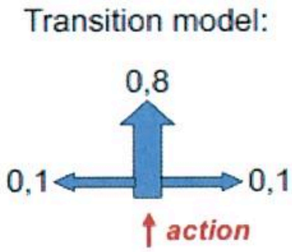
Professora: Anna Helena Reali Costa

Mestrando: Luiz Müller

- **07/05/2019: Markov Decision Process (MDP) and Value Iteration (VI)**

1) How to reach goal after 12 steps (avoiding (2,4))?





1.1) <u>Sequence of actions:</u>

Possíveis ações: A = {Up(U), Down(D), Left(L), Right(R)}

Sequência de 12 passos: [U, U, R, R, L, L, L, L, L, R, R, R]

1.2) <u>History h:</u>

Para a sequência acima definida uma possível história é:

h = [(1,1), (2,1), (3,1), (3,2), (3,3), (3,2), (3,1), (3,1), (3,1), (3,1), (3,2), (3,3), (3,4)]

1.3) <u>Probability of h:</u>

A probabilidade de ocorrência da história do item 1.2, considerando a sequência de ações de 12 passos definida no item 1.1 e o modelo de transição proposto é de:

p(h) = 0,8^6 * (0,8+0,1)*(0,8+0,1)*(0,8+0,1) * 0,8^3

p(h) = 0,0978 = 9,78%

1.4) <u>Utility of h:</u>

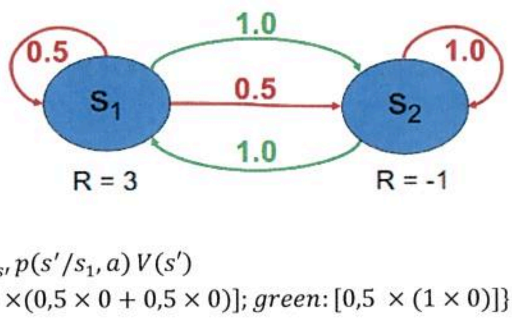
V(h) = 12*(-0,04) + 1

V(h) = 0,52

---

2) Show the values V(s) at each iteration of VI (Value Iteration).

- γ = 0.5, A={red, green}, S={s_1, s_2}



<u>1ª iteração:</u>

V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$

V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 0 + 0{,}5 \times 0)];\ green\colon [0{,}5\ \times (1 \times 0)]\}$

V'(s_1) = $3 + \max_a\{0; 0\}$

V'(s_1) = 3

V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red\colon [0{,}5 \times (1 \times 0)];\, green\colon [0{,}5\ \times (1 \times 0)]\}$
V'(s_2) = $-1 + \max_a\{0; 0\}$
V'(s_2) = -1

2ª iteração:
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 3 + 0{,}5 \times (-1))];\, green\colon [0{,}5\ \times (1 \times (-1))]\}$
V'(s_1) = $3 + \max_a\{0{,}5; -0{,}5\}$
V'(s_1) = 3,5
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red\colon [0{,}5 \times (1 \times (-1))];\, green\colon [0{,}5\ \times (1 \times 3)]\}$
V'(s_2) = $-1 + \max_a\{-0{,}5; 1{,}5\}$
V'(s_2) = 0,5

3ª iteração:
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 3{,}5 + 0{,}5 \times (0{,}5))];\, green\colon [0{,}5\ \times (1 \times (0{,}5))]\}$
V'(s_1) = $3 + \max_a\{1; 0{,}25\}$
V'(s_1) = 4,0
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red\colon [0{,}5 \times (1 \times (0{,}5))];\, green\colon [0{,}5\ \times (1 \times 3{,}5)]\}$
V'(s_2) = $-1 + \max_a\{0{,}25; 1{,}75\}$
V'(s_2) = 0,75

4ª iteração:
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 4 + 0{,}5 \times (0{,}75))];\, green\colon [0{,}5\ \times (1 \times (0{,}75))]\}$
V'(s_1) = $3 + \max_a\{1{,}1875; 0{,}375\}$
V'(s_1) = 4,1875
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red\colon [0{,}5 \times (1 \times (0{,}75))];\, green\colon [0{,}5\ \times (1 \times 4)]\}$
V'(s_2) = $-1 + \max_a\{0{,}375; 2\}$
V'(s_2) = 1

5ª iteração:
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 4{,}1875 + 0{,}5 \times 1)];\, green\colon [0{,}5\ \times (1 \times 1)]\}$
V'(s_1) = $3 + \max_a\{1{,}296875; 0{,}5\}$
V'(s_1) = 4,296875
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red\colon [0{,}5 \times (1 \times 1)];\, green\colon [0{,}5\ \times (1 \times 4{,}1875)]\}$
V'(s_2) = $-1 + \max_a\{0{,}5; 2{,}09375\}$
V'(s_2) = 1,09375

6ª iteração:
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1) = $3 + \max_a\{red\colon [0{,}5 \times (0{,}5 \times 4{,}296875 + 0{,}5 \times 1{,}09375)];\, green\colon [0{,}5\ \times (1 \times 1{,}09375)]\}$
V'(s_1) = $3 + \max_a\{1{,}34765625; 0{,}546875\}$
V'(s_1) $\cong$ 4,348
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$

V'(s_2) = $-1 + \max_a\{red: [0{,}5 \times (1 \times 1{,}09375)]; green: [0{,}5\ \times (1 \times 4{,}296875)]\}$
V'(s_2) = $-1 + \max_a\{0{,}546875; 2{,}1484375\}$
V'(s_2) $\cong$ 1,148

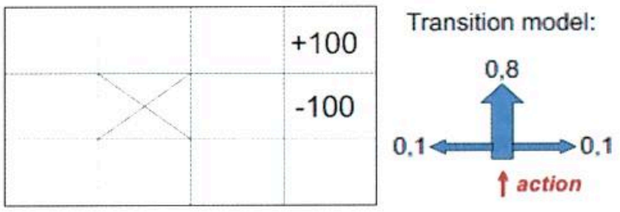
<u>7ª iteração:</u>
V'(s_1) = $r(s_1) + \max_a \gamma \times \sum_{s'} p(s'/s_1, a)\, V(s')$
V'(s_1)= $3 + \max_a\{red: [0{,}5 \times (0{,}5 \times 4{,}34765625 + 0{,}5 \times 1{,}1484375)]; green: [0{,}5 \times (1 \times 1{,}1484375)]\}$
V'(s_1) = $3 + \max_a\{1{,}374023438; 0{,}57421875\}$
$\boxed{\textbf{V'(s}_1\textbf{)} \cong \textbf{4,4}}$
V'(s_2) = $r(s_2) + \max_a \gamma \times \sum_{s'} p(s'/s_2, a)\, V(s')$
V'(s_2) = $-1 + \max_a\{red: [0{,}5 \times (1 \times 1{,}1484375)]; green: [0{,}5\ \times (1 \times 4{,}34765625)]\}$
V'(s_2) = $-1 + \max_a\{0{,}57421875; 2{,}173828125\}$
$\boxed{\textbf{V'(s}_2\textbf{)} \cong \textbf{1,2}}$

---

3) V*(S)? π*(S)?

Each action: r(s,a) = -1 $\gamma = 0.9$

$$V^{t+1}(s) \leftarrow r(s) + \max_a \gamma \sum_{s'} p(s' \mid s, a) V^t(s')$$



CÓDIGO EM PYTHON 3

```
#PCS-5024 - Aprendizado Estatístico
#1º Quadrimestre de 2019
#Profª. Anna Helena Reali Costa
#Mestrando: Luiz Müller
#07/05/2019: Homework Markov Decision Process (MDP) and Value Iteration (VI)

#3) V*(S)? π*(S)?

import numpy as np
import math

def max(x,y):
    """
    -> Calcula o valor dentro do somatório na fórmula do Value Iteration (VI) para cada ação e retorna o máximo
    :param x: posição x na matriz x = 0, 1 ou 2
    :param y: posição y na matriz y = 0, 1, 2 ou 3
    :return maximo: maior valor calculado para o somatório na fórmula do Value Iteration (VI)
    """
    if(x==0 and y==0):
        Upper=(0.9*(V[0,0])+0.1*(V[0,1]))
        Down=(0.8*(V[1,0])+0.1*(V[0,0])+0.1*(V[0,1]))
        Left=(0.9*(V[0,0])+0.1*(V[0,1]))
        Right=(0.8*(V[0,1])+0.1*(V[0,0])+0.1*(V[1,0]))
    if(x==0 and y==1):
        Upper=(0.8*(V[0,1])+0.1*(V[0,0])+0.1*(V[0,2]))
        Down=(0.8*(V[0,1])+0.1*(V[0,0])+0.1*(V[0,2]))
        Left=(0.8*(V[0,0])+0.2*(V[0,1]))
```

```
        Right=(0.8*(V[0,2])+0.2*(V[0,1]))
    if(x==0 and y==2):
        Upper=(0.8*(V[0,2])+0.1*(V[0,1])+0.1*(V[0,3]))
        Down=(0.8*(V[1,2])+0.1*(V[0,1])+0.1*(V[0,3]))
        Left=(0.8*(V[0,1])+0.1*(V[0,2])+0.1*(V[1,2]))
        Right=(0.8*(V[0,3])+0.1*(V[0,2])+0.1*(V[1,2]))
    if(x==1 and y==0):
        Upper=(0.8*(V[0,0])+0.2*(V[1,0]))
        Down=(0.8*(V[2,0])+0.2*(V[1,0]))
        Left=(0.8*(V[1,0])+0.1*(V[0,0])+0.1*(V[2,0]))
        Right=(0.8*(V[1,0])+0.1*(V[0,0])+0.1*(V[2,0]))
    if(x==1 and y==2):
        Upper=(0.8*(V[0,2])+0.1*(V[1,2])+0.1*(V[1,3]))
        Down=(0.8*(V[2,2])+0.1*(V[1,2])+0.1*(V[1,3]))
        Left=(0.8*(V[1,2])+0.1*(V[0,2])+0.1*(V[2,2]))
        Right=(0.8*(V[1,3])+0.1*(V[0,2])+0.1*(V[2,2]))
    if(x==2 and y==0):
        Upper=(0.8*(V[1,0])+0.1*(V[2,0])+0.1*(V[2,1]))
        Down=(0.9*(V[2,0])+0.1*(V[2,1]))
        Left=(0.9*(V[2,0])+0.1*(V[1,0]))
        Right=(0.8*(V[2,1])+0.1*(V[2,0])+0.1*(V[1,0]))
    if(x==2 and y==1):
        Upper=(0.8*(V[2,1])+0.1*(V[2,0])+0.1*(V[2,2]))
        Down=(0.8*(V[2,1])+0.1*(V[2,0])+0.1*(V[2,2]))
        Left=(0.8*(V[2,0])+0.2*(V[2,1]))
        Right=(0.8*(V[2,2])+0.2*(V[2,1]))
    if(x==2 and y==2):
        Upper=(0.8*(V[1,2])+0.1*(V[2,1])+0.1*(V[2,3]))
        Down=(0.8*(V[2,2])+0.1*(V[2,1])+0.1*(V[2,3]))
        Left=(0.8*(V[2,1])+0.1*(V[2,2])+0.1*(V[1,2]))
        Right=(0.8*(V[2,3])+0.1*(V[2,2])+0.1*(V[1,2]))
    if(x==2 and y==3):
        Upper=(0.8*(V[1,3])+0.1*(V[2,3])+0.1*(V[2,2]))
        Down=(0.9*(V[2,3])+0.1*(V[2,2]))
        Left=(0.8*(V[2,2])+0.1*(V[2,3])+0.1*(V[1,3]))
        Right=(0.9*(V[2,3])+0.1*(V[1,3]))

    maximo=Upper
    if Down>maximo:
        maximo=Down
    if Left>maximo:
        maximo=Left
    if Right>maximo:
        maximo=Right
    print(f'({x},{y}): U={Upper:.2f}, D={Down:.2f}, L={Left:.2f}, R={Right:.2f} e max={maximo:.2f}')
    return maximo


#Dados de entrada
R=-1        #Custo de qualquer ação
gama=0.9  #Fator de desconto para penalizar os prêmios mais tardios (mais próximo de zero -> mais imediatista)
r=np.zeros((3,4),dtype=float)   #Inicializando com valor/utilidade zero para todos os demais estados
r[0,3]=100
r[1,3]=-100
V=np.zeros((3,4),dtype=float)   #Inicializando com valor/utilidade zero para todos os demais estados
V[0,3]=100
V[1,3]=-100
V1=np.zeros((3,4),dtype=float) #Inicializando com valor/utilidade zero para todos os demais estados
V1[0,3]=100
V1[1,3]=-100
delta=100*np.ones((3,4),dtype=float) #Para delta.max() ser igual a 100 no início
criterio_convergencia=0.001            #Definindo o critério de convergência
```

```
print(f'V0={V1}')
i=0
while (delta.max()>criterio_convergencia):
  i=i+1
  print(f'\n\033[7;30mApós a iteração {i}:\033[m')
  for y in range (0,4):
    for x in range (0,3):
      if (not (x==1 and y==1) and not (x==0 and y==3) and not (x==1 and y==3)): #(0,3) e (1,3) foram considerados estados absorventes
        V1[x,y]=r[x,y]+R+gama*max(x,y)
  print(f'V={V1}')
  delta=abs(V1-V)
  print(f'delta={delta.max():.3f}')
  V=V1.copy()

print(f'\n3.1) Foram necessárias {i} iterações para delta<{criterio_convergencia}. O valor de V* é:\n')
print(f'V*={V1}')

#Sabendo V* é fácil encontrar a política ótima Pi*: escolhendo a ação que maximiza a utilidade esperada do estado subsequente.
Pi=[['Right','Right','Right',' X '],['Upper',' X ','Upper',' X '],['Upper','Right','Upper','Left ']]
print(f'\n3.2) O valor de Pi* é:\n')
print(f'Pi*={Pi[0]}\n    {Pi[1]}\n    {Pi[2]}')
```

RESPOSTA:

3.1) Foram necessárias 18 iterações para delta<0.001. O valor de V* é:

V*= [[61.10802774 72.06816902 83.46652670 100]
[52.43618493 X 55.05043826 -100]
[44.10415518 37.56700325 44.17381105 23.96153231]]

3.2) O valor de Pi* é:

Pi*=['Right' , 'Right', 'Right' , X]
['Upper', X , 'Upper', X]
['Upper', 'Right', 'Upper', 'Left ']

→	→	→	+100
↑	X	↑	-100
↑	→	↑	←