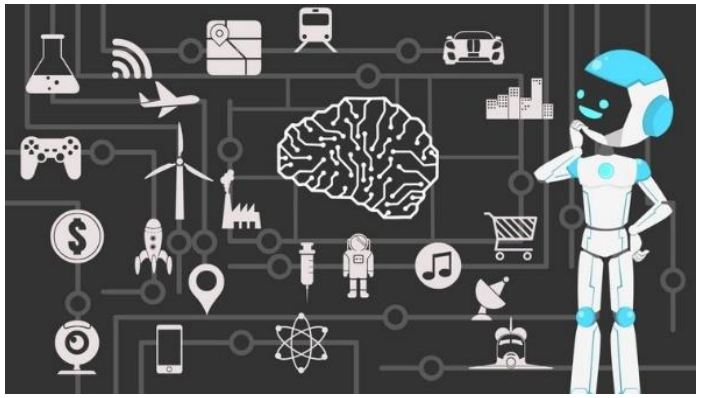


FORMAÇÃO INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING

REINFORCEMENT LEARNING

INTRODUÇÃO

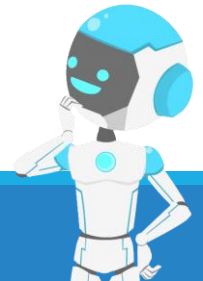
Aprender com Ambiente

- Como deve agir um agente de IA, em um ambiente, a fim de solucionar um problema?

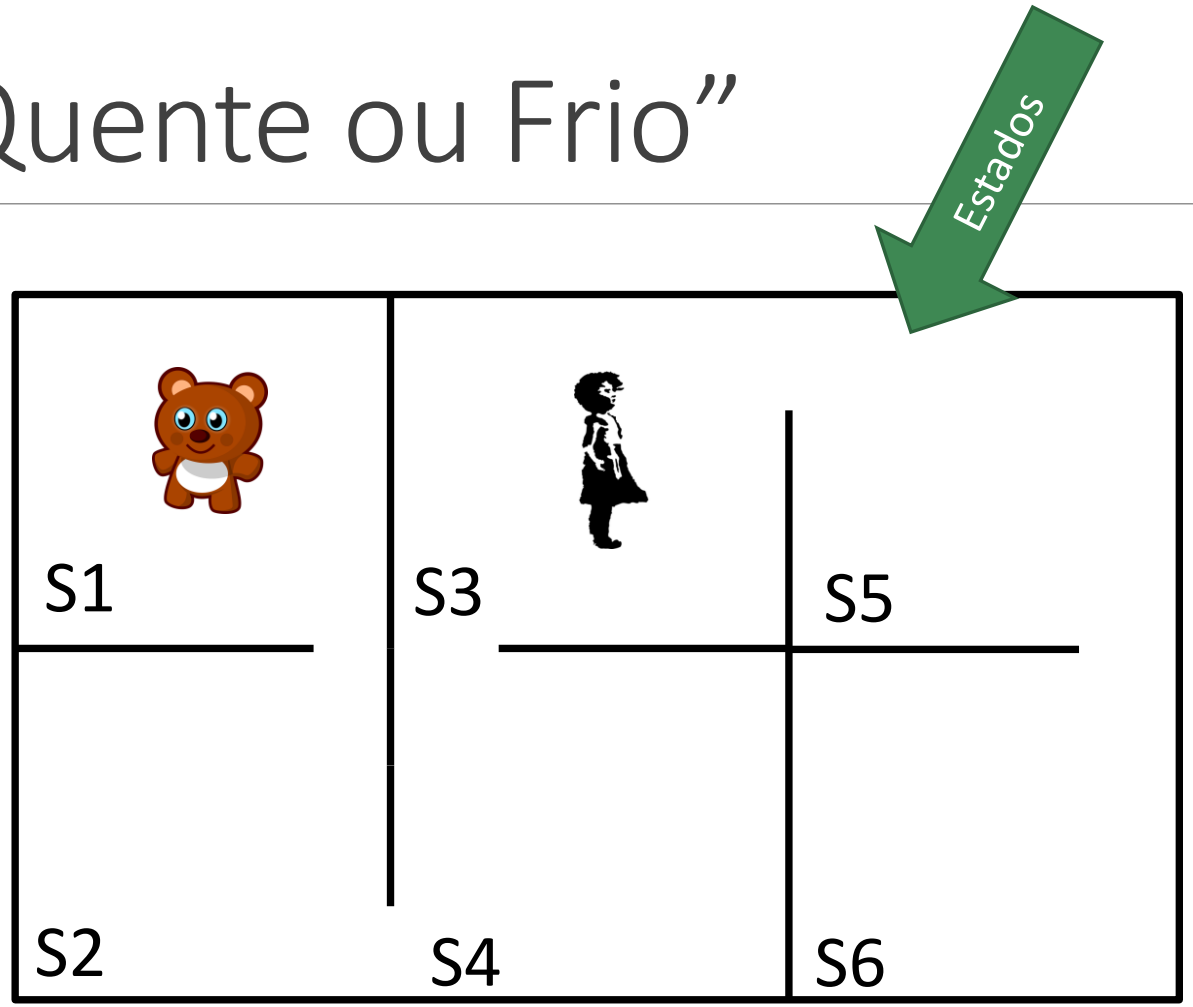
- Proposta:
  - Método de tentativas com acertos e erros
  - Recompensas por acertos

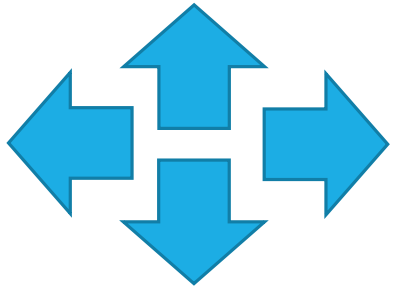
- Markov decision process
  - A cada etapa um agente em um estado S_1, escolhe uma ação A
  - O agente muda de estado S_2 e recebe uma recompensa R

- Q learning
  - Decide qual a melhor ação com base em uma função e a avaliação da interação com o ambiente

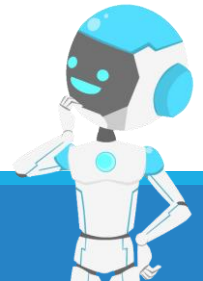


Jogo “Quente ou Frio”

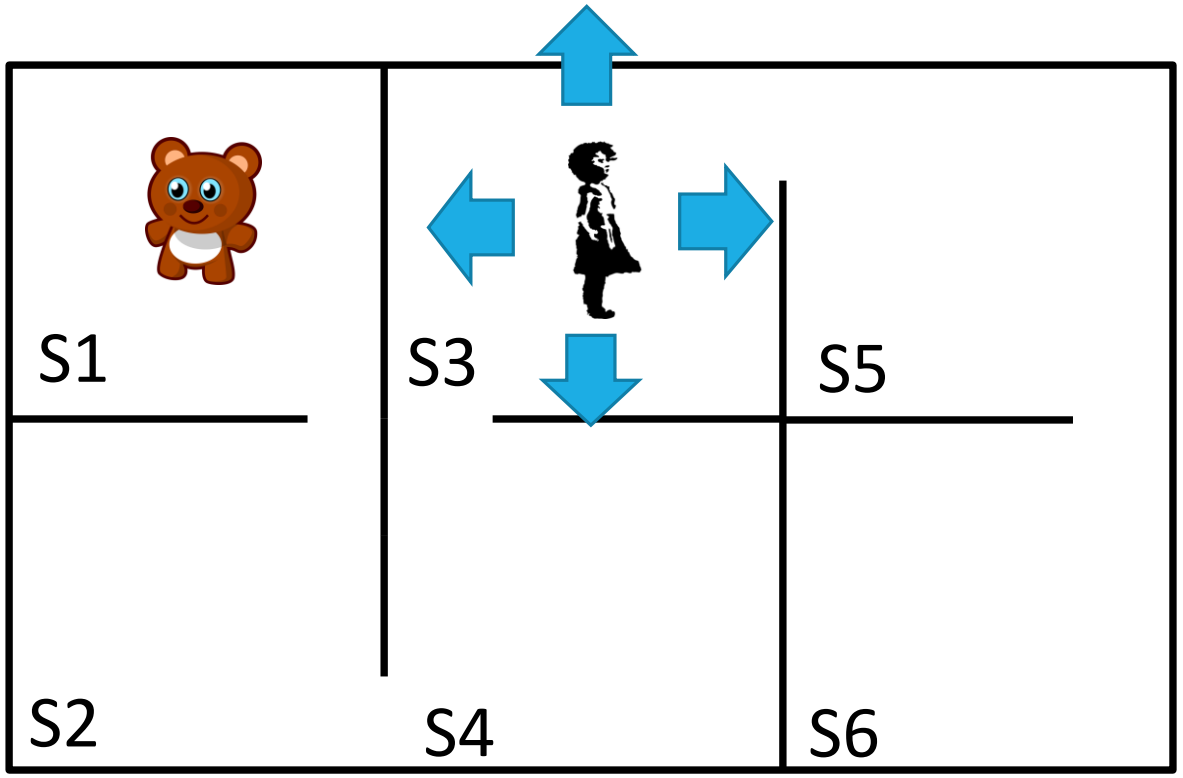




S_1 - Estado Atual
S_2- Novo Estado
A – Ação: Mudança de Estado
R - Recompensa

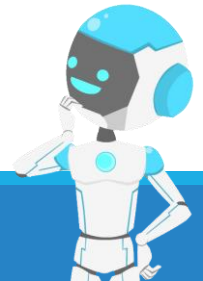


Jogo “Quente ou Frio”

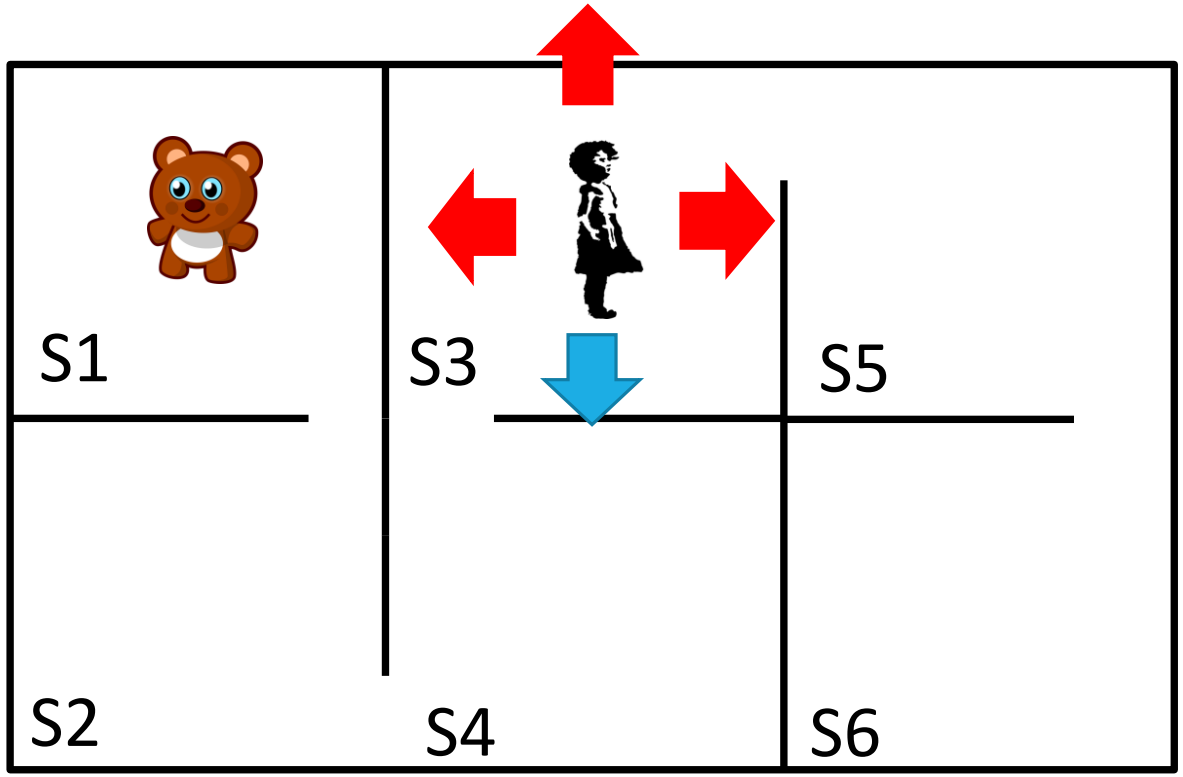


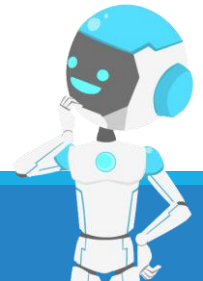
S_1	A	S_2	R
S3	Para Direita	S5	Frio
S5	Para Baixo	S6	Frio
...			
S3	Para Direita	S3	Frio
S3	Para Baixo	S4	Quente
S4	Para Esquerda	S2	Quente

S_1 - Estado Atual
S_2- Novo Estado
A - Ação
R - Recompensa

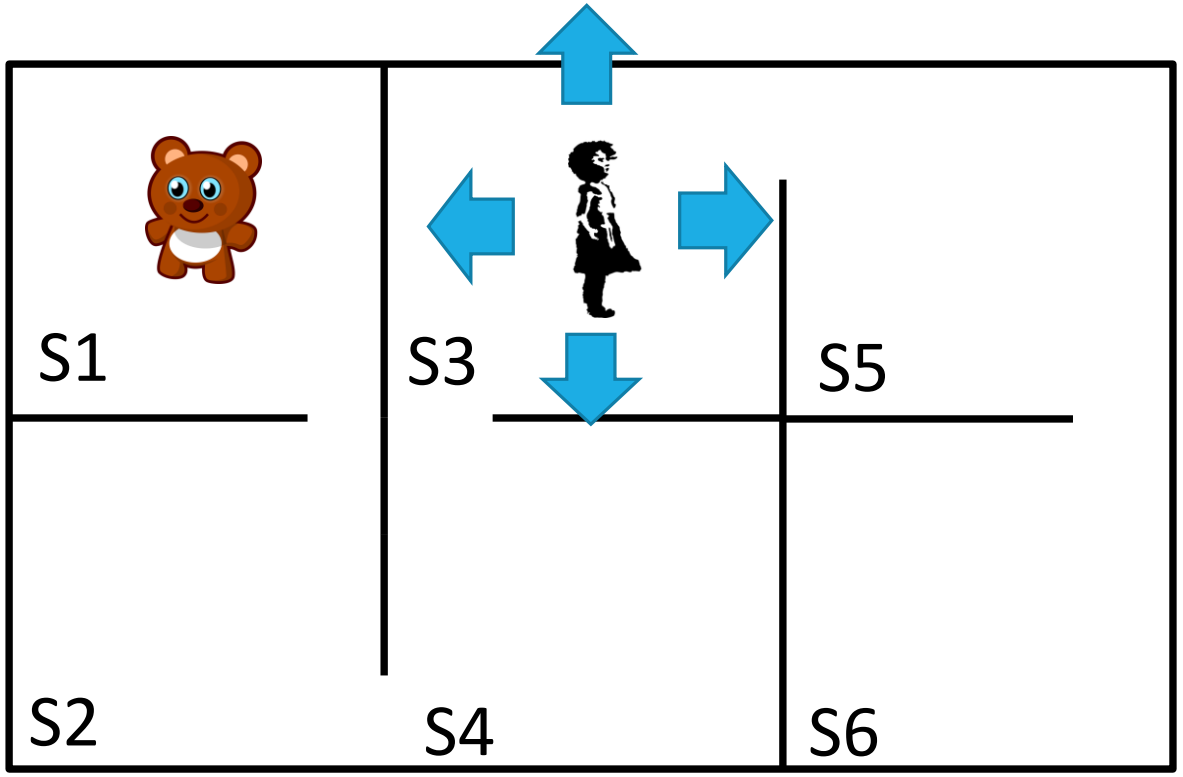


Objetivo do Reinforcement Learning

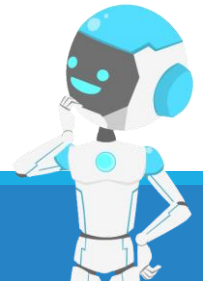




Linguagem Computacional

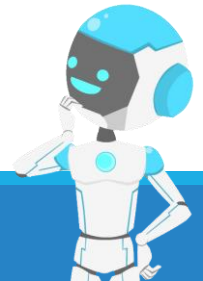


S_1	A	S_2	R
S2	C	S1	10
S2	D	S4	-1
S4	C	S3	-1
S4	E	S2	-1
S3	B	S4	-1
S3	D	S5	-1
S5	E	S3	-1
S5	B	S6	-1
S6	C	S5	-1
S_N	C,B,E,D	S_N	-1

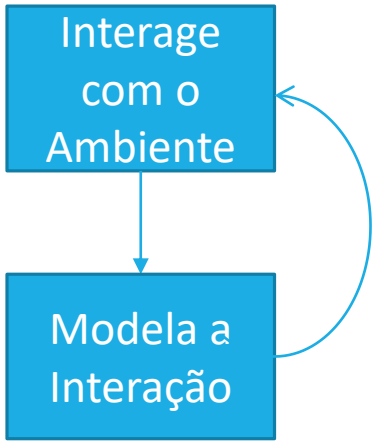


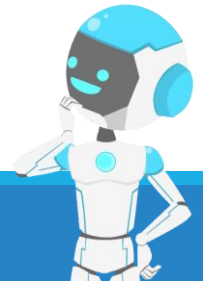
Diferença da Classificação

- Não há dados de treino com uma classe
- O aprendizado se da na tentativa e erro do agente e pelo sistema de recompensas



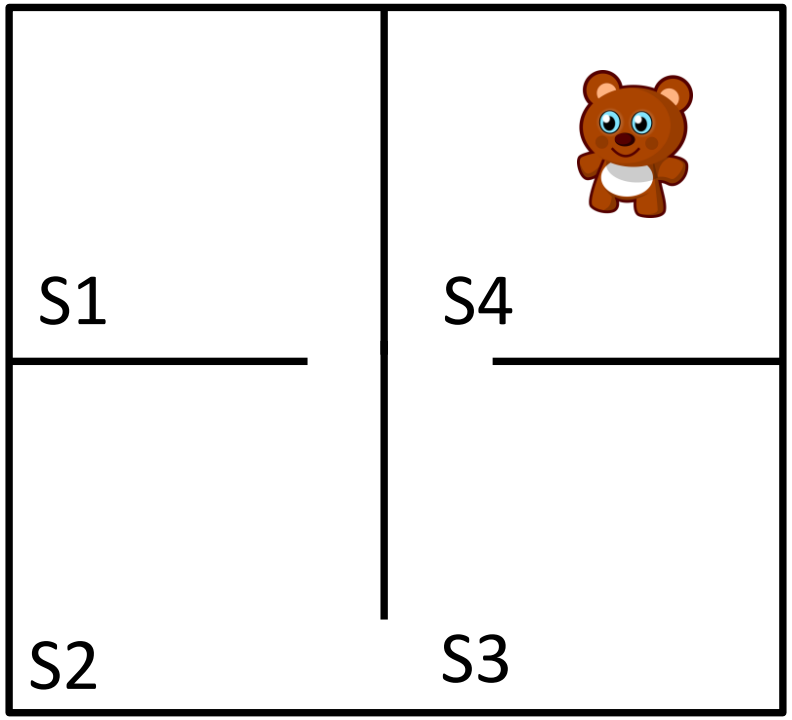
Etapas para Aprendizado



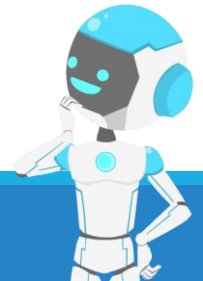


ReinforcementLearning: Grid World

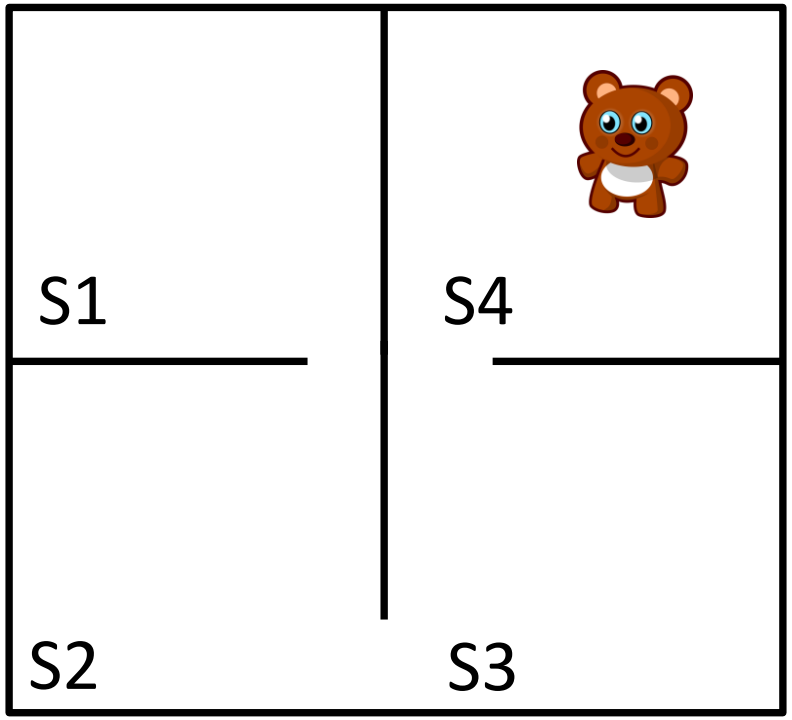
- Pacote do MIT



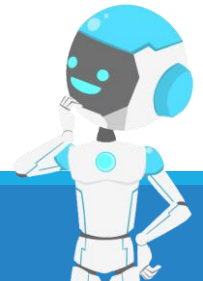

```
function (state, action)
{
   next_state <- state
   if (state == state("s1") && action == "down")
      next_state <- state("s2")
   if (state == state("s2") && action == "up")
      next_state <- state("s1")
   if (state == state("s2") && action == "right")
      next_state <- state("s3")
   if (state == state("s3") && action == "left")
      next_state <- state("s2")
   if (state == state("s3") && action == "up")
      next_state <- state("s4")
   if (next_state == state("s4") && state != state("s4")) {
      reward <- 10
   }
   else {
      reward <- -1
   }
   out <- list(NextState = next_state, Reward = reward)
   return(out)
}
```



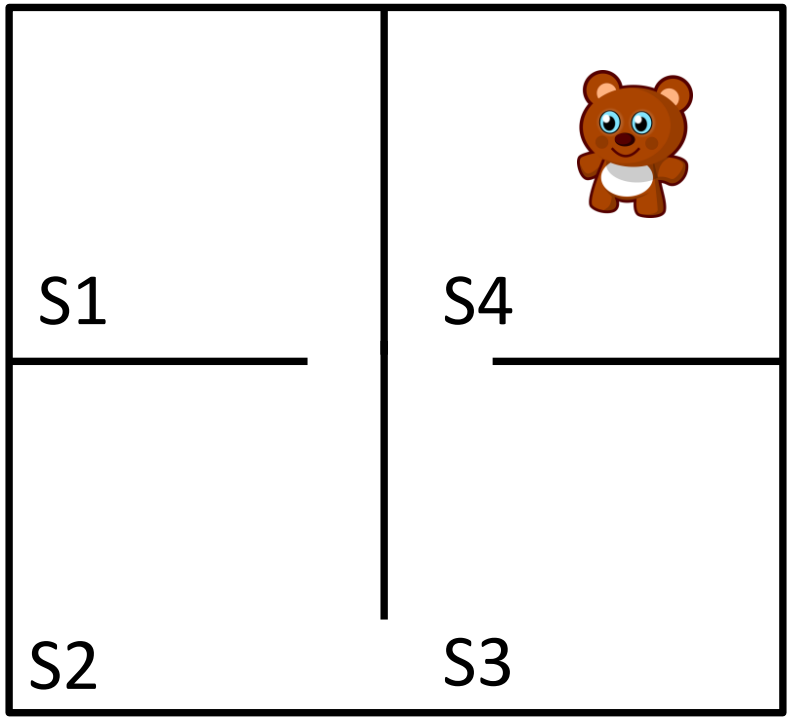
sampleExperience



	State	Action	Reward	NextState
1	s3	up	10	s4
2	s1	up	-1	s1
3	s1	down	-1	s2
4	s1	up	-1	s1
5	s4	left	-1	s4
6	s4	left	-1	s4
7	s1	down	-1	s2
8	s1	right	-1	s1
9	s3	left	-1	s2
10	s4	left	-1	s4
11	s4	left	-1	s4
12	s4	up	-1	s4
13	s1	up	-1	s1
14	s2	right	-1	s3
15	s1	up	-1	s1
16	s1	up	-1	s1
17	s1	up	-1	s1
18	s2	up	-1	s1



Aprender através do “reforço”




```
State-Action function Q
        right          up        down        left
s1 -0.6978541 -0.6564206   0.7572454 -0.7049049
s2  3.5781631 -0.6999091   0.7383973  0.7646410
s3  3.5910362  9.1588341   3.5826810  0.7490790
s4 -1.8577368 -1.8663151  -1.8468374 -1.8485593

Policy
     s1      s2      s3      s4
 "down" "right"    "up"  "down"

Reward (last iteration)
[1] -274
```

Polices