

COMP0271: Inteligência Artificial

Aprendizagem por Reforço

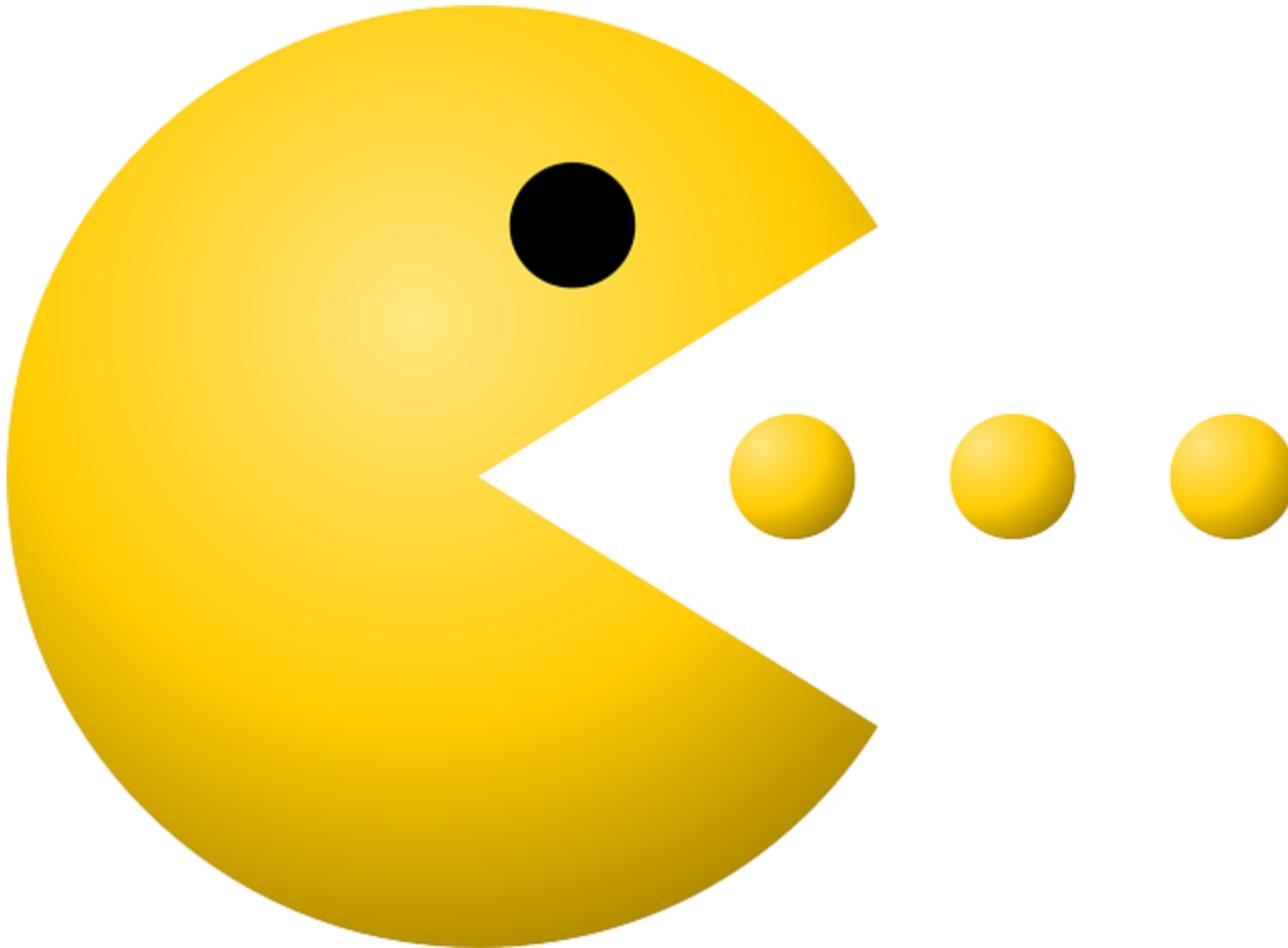


Professor: Hendrik Macedo
Universidade Federal de Sergipe, Brasil

A máquina que aprende a jogar melhor que você!

Hendrik Macedo

05/02/2016



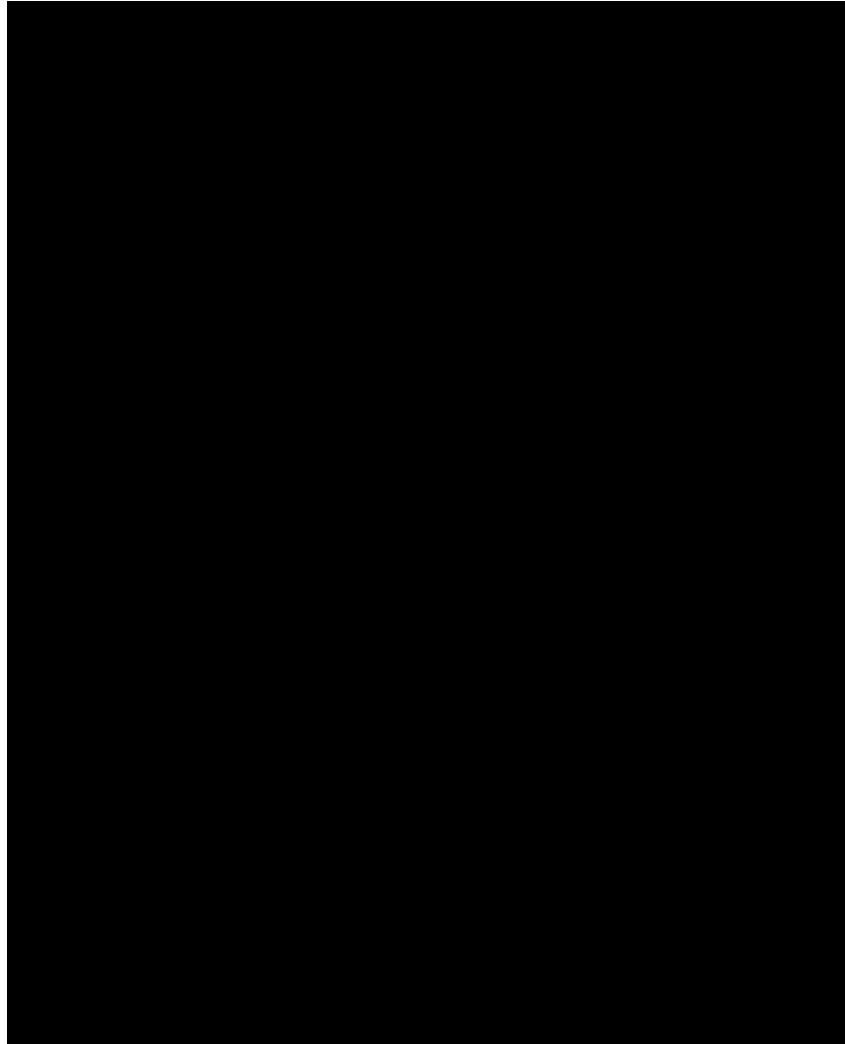
A máquina que aprende a jogar melhor que você!

Hendrik Macedo

05/02/2016

...Google's Deep Mind... o computador foi capaz de, por si só, adquirir expertise sobre uma seleção criteriosa de **49 desses jogos** e em muitos casos **ultrapassar o desempenho dos melhores jogadores humanos** sem sequer observar como estes jogam o jogo ou mesmo **obter qualquer feedback humano** sobre o melhor caminho a seguir. Após explorar cada jogo por inúmeras vezes, o computador era capaz inclusive de descobrir estratégias avançadas que poucos humanos tinham conhecimento...o Q-learning é a versão matemática de um conceito da psicologia chamado **Aprendizagem por Reforço** [3], que é um sistema de recompensa que acredita-se guiar o processo de aprendizagem em humanos e outros animais. No caso do DQN, a **recompensa vem na forma de pontos no jogo**: à medida que o computador tenta diferentes movimentos no jogo, ele **guarda o registro de que combinações levaram às maiores pontuações....**o DQN enxerga e interage com o jogo exatamente como os humanos o fazem: realizando movimentos e **observando os pixels do jogo modificarem....**

Breakout (by Google's Deep Mind)

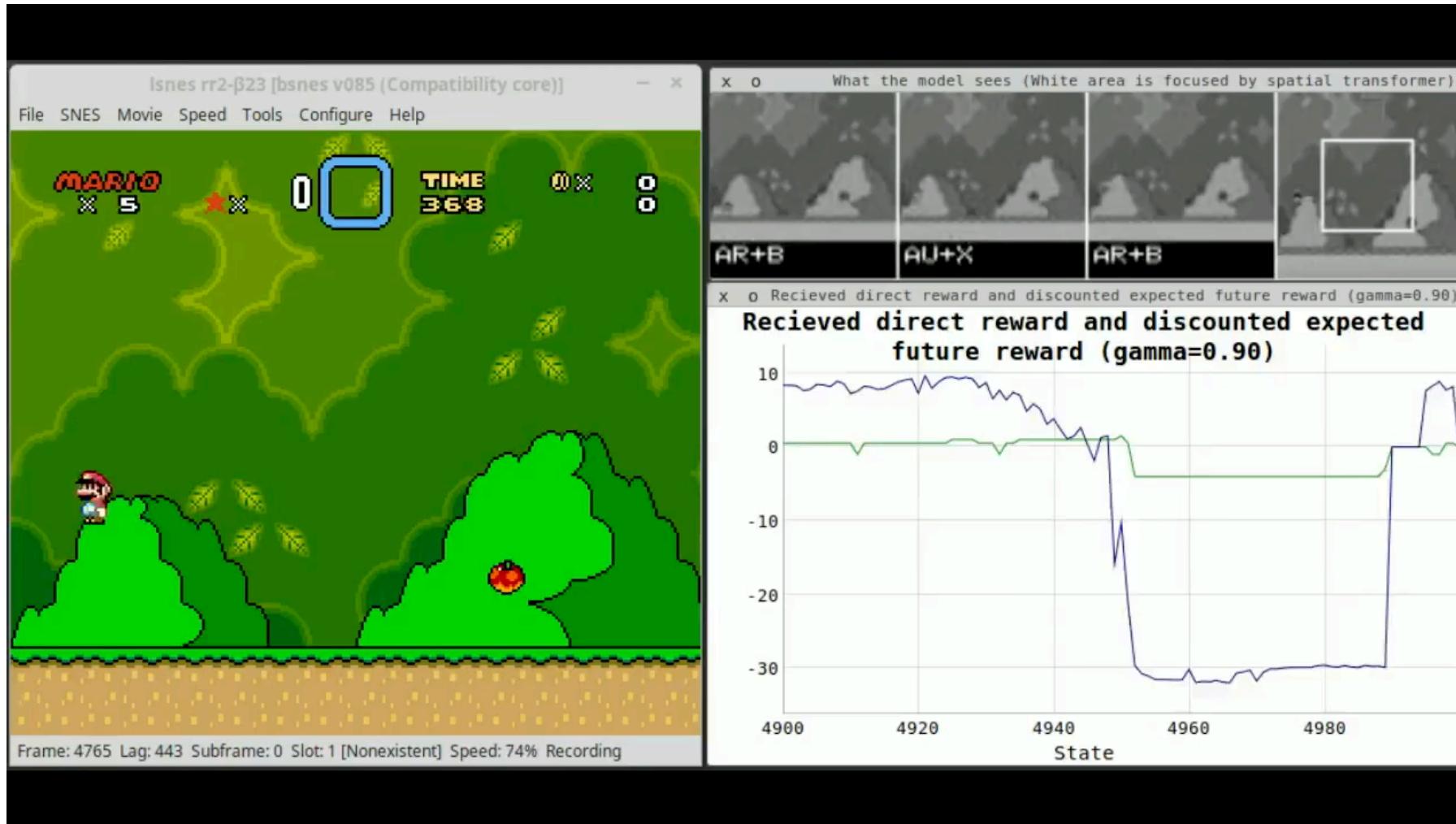


Flappy Bird

- Espaço de estados
 - Distância vertical (discretizada) ao cano inferior
 - Distância horizontal (discretizada) ao próximo par de canos
 - Morto ou Vivo
- Ações
 - Clicar...
 - ... ou não
- Recompensas
 - +1 se Flappy Bird permencer vivo
 - -1000 se Flappy Bird morrer
- 6-7 horas de treinamento



Super Mario



Aibo (by Sony)

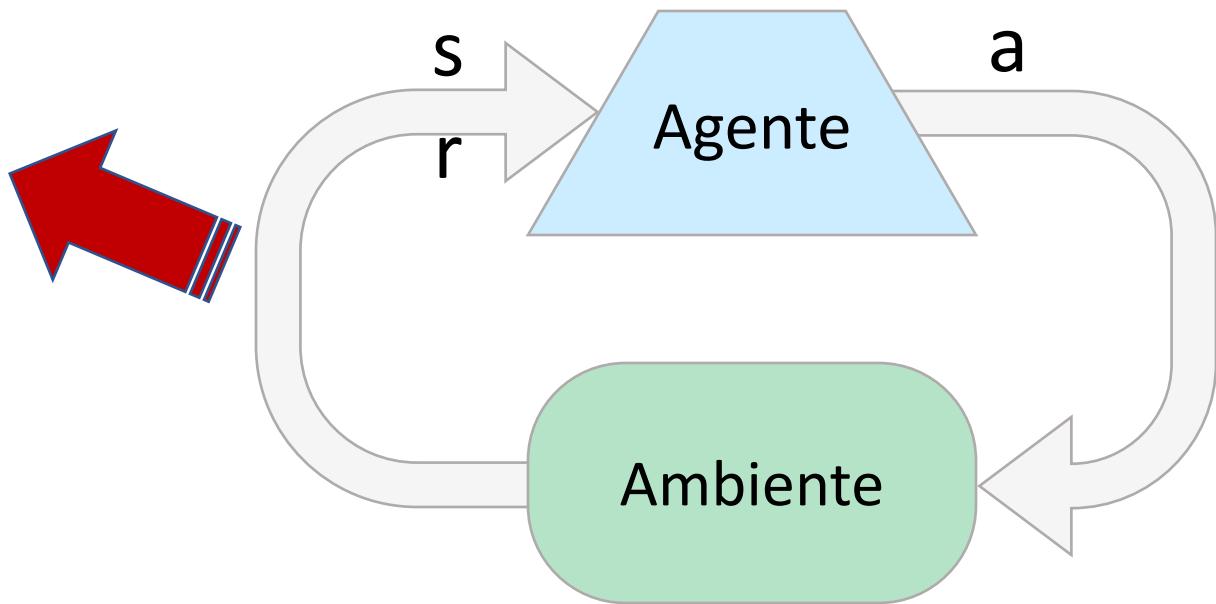


Reinforcement Learning

MDP: $s \in S$, $a \in A$, $T(s,a,s')$, $R(s,a,s')$

Objetivo: encontrar $\pi(s)$

Peculiaridade: $T ??$ $R ??$

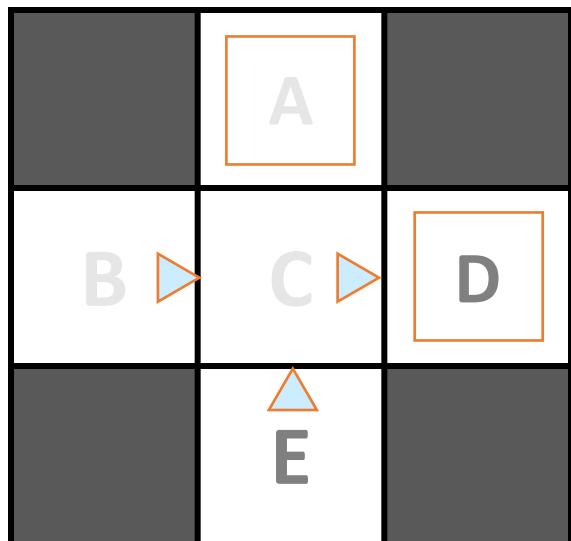


Aprendizagem baseada em modelo

- Aprende um modelo aproximado baseado em experiências
- Passo 1: Aprende modelo MDP empiricamente
 - Conta saídas s' para cada s, a
 - Normaliza para dar uma estimativa de $\hat{T}(s, a, s')$
 - Descobre cada $\hat{R}(s, a, s')$ para (s, a, s')
- Passo 2: resolve o MDP aprendido via *Value Iteration*, por exemplo.

baseada em modelo : Exemplo

Política inicial π



Assumindo: $\gamma = 1$

Observações (treinamento)

Obs 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Obs 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Obs 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Obs 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Modelo aprendido

$$\hat{T}(s, a, s')$$

$T(B, \text{east}, C) = 1.00$
 $T(C, \text{east}, D) = 0.75$
 $T(C, \text{east}, A) = 0.25$
...

$$\hat{R}(s, a, s')$$

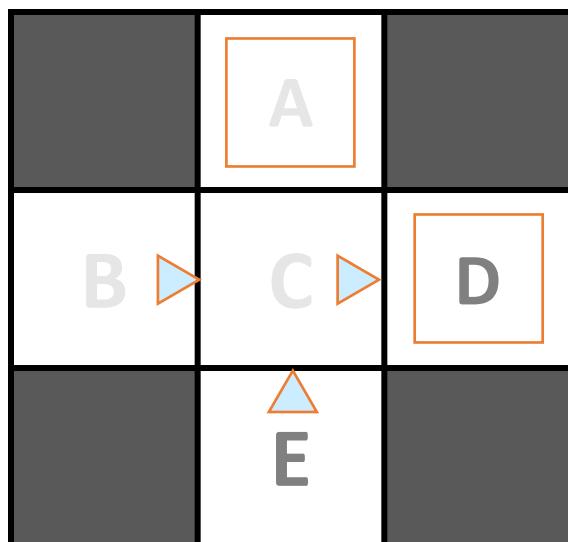
$R(B, \text{east}, C) = -1$
 $R(C, \text{east}, D) = -1$
 $R(D, \text{exit}, x) = +10$
...

Aprendizagem livre de modelo

- Tarefa simplificada: avaliação da Política
 - Entrada: política fixa $\pi(s)$
 - ??? $T(s,a,s')$
 - ??? $R(s,a,s')$
 - Objetivo: aprender os valores dos estados
 - Agente apenas executa a política e aprende a partir da experiência
- **Avaliação direta:** considerar valores de amostra observados
 - Agir de acordo à π
 - Toda vez que se visita um estado, guarda a soma de recompensas descontadas
 - Média dessas amostras

livre de modelo: Exemplo

Política inicial π



Assumindo: $\gamma = 1$

Observações (treinamento)

Obs 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Obs 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

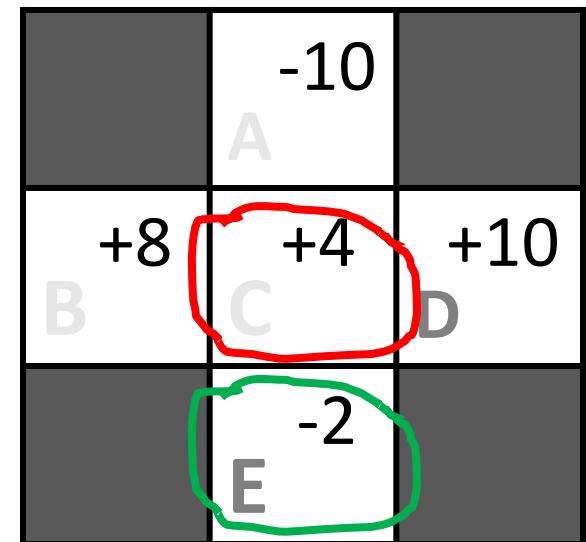
Obs 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Obs 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Valores de saída



livre de modelo

Vantagens vs. Desvantagens

- Vantagens

- Não requer conhecimento sobre T e R
- Computa os valores médios corretos usando apenas amostras

- Desvantagens

- Desperdiça informações sobre conexões entre estados
- Cada estado precisa ser aprendido separadamente
→ longo tempo de aprendizado

Valores de saída

	-10 A	
+8 B	+4 C	+10 D
	-2 E	

livre de modelo : *Alternativa*

Avaliação de Política **baseada em amostragem**

Como não conhecemos T e R na equação abaixo...

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

... observamos amostras de saídas s' e calculamos a média

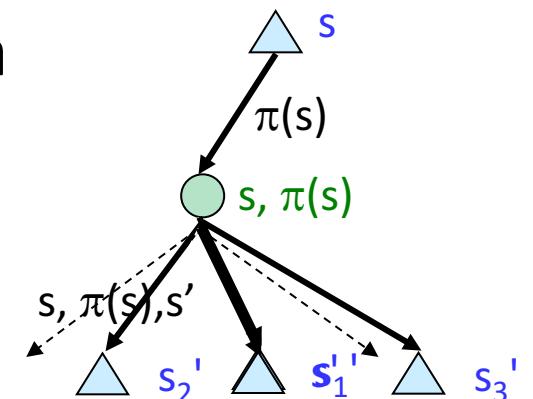
$$\text{sample}_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$\text{sample}_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

...

$$\text{sample}_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i \text{sample}_i$$



Baseado em modelo vs. Livre de modelo

Objetivo: Computar idade esperada de estudantes de uma turma

P(A) conhecido

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \dots$$

Sem P(A), necessidade de amostras $[a_1, a_2, \dots, a_N]$

P(A): “Baseado em modelo”

Funciona porque eventualmente aprende-se o modelo correto.

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

P(A): “Livre de modelo”

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Funcion porque amostras aparecem com as frequências corretas.

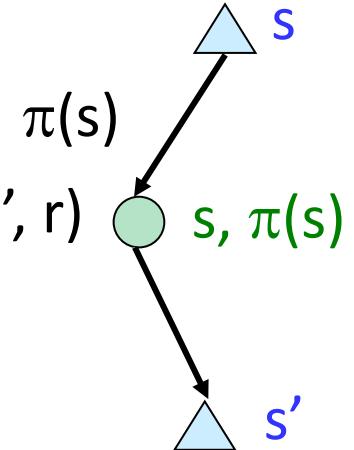
Temporal Difference Learning (TDL)

- Aprender com toda a experiência
 - Atualizar $V(s)$ cada vez que experimentar uma transição (s, a, s', r)
- Método
 - Avaliação de Política fixa!
 - Atualiza $V(s)$ para qualquer estado sucessor

Amostra de $V(s)$: $sample = R(s, \pi(s), s') + \gamma V^\pi(s')$

Atualização de $V(s)$: $V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$

$V^\pi(s) \leftarrow V^\pi(s) + \alpha(sample - V^\pi(s))$



Taxa de aprendizado

Obs: Média exponencial

- Amostras mais recentes possuem mais peso:

$$\bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

$$\bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \dots}{1 + (1 - \alpha) + (1 - \alpha)^2 + \dots}$$

- Diminuir taxa de aprendizagem com o tempo auxilia na convergência

TDL : exemplo

Estados

	A	
B	C	D
	E	

Transições observadas

B, east, C, -2

	0	
0	0	8
	0	

C, east, D, -2

	0	
-1	0	8
	0	

	0	
-1	3	8
	0	

Assumindo:

$$\gamma = 1, \alpha = 1/2$$

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + \alpha [R(s, \pi(s), s') + \gamma V^\pi(s')]$$

$$\begin{aligned} V(B) &= 0.5 * 0 & + 0.5 * [-2 + 1 * V(C) = 0] &= -1 \\ V(C) &= 0.5 * 0 & + 0.5 * [-2 + 1 * V(D) = 8] &= 3 \end{aligned}$$

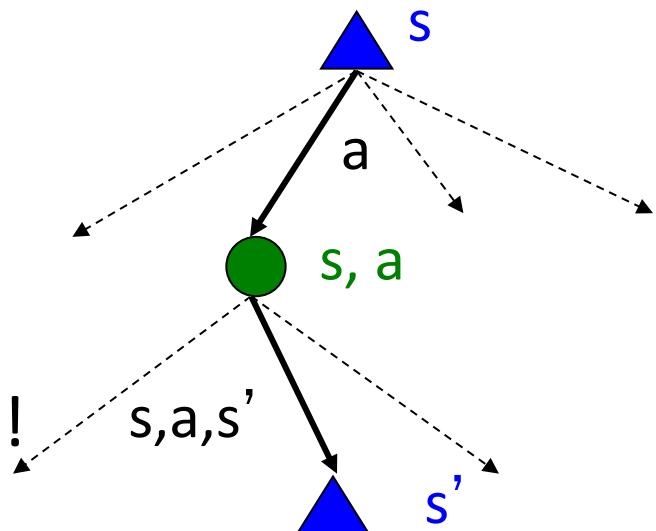
Problemas com TDL

- Serve para avaliação de políticas mas não para definir nova política:

$$\pi(s) = \arg \max_a Q(s, a)$$

$$Q(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V(s')]$$

- Idéia: aprender Q-values, não V
- Torna seleção de ações livre de modelo também!



Q-Learning

- Q-value iteration baseado em amostras

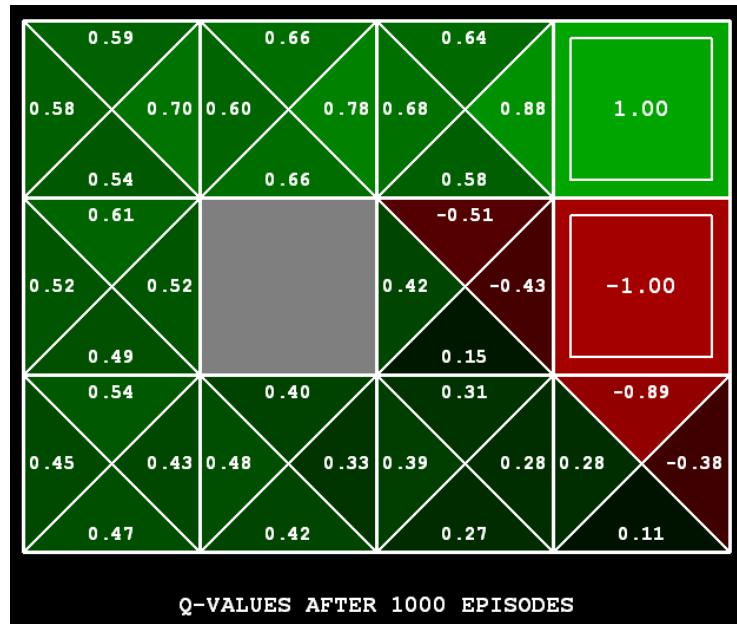
$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

- Aprende valores $Q(s, a)$ à medida que explora o ambiente

- Pega amostra (s, a, s', r)
- Considera estimativa antiga: $Q(s, a)$
- Considera nova estimativa:

$$\text{sample} = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

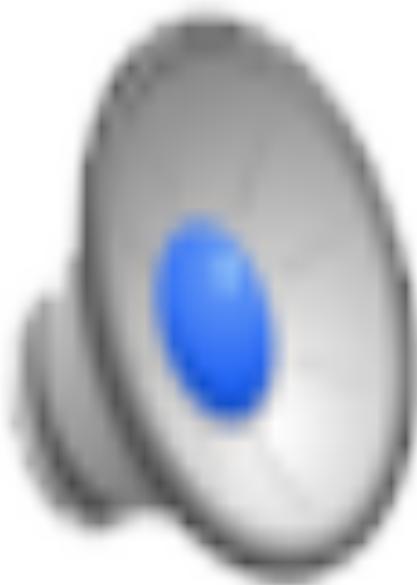
- Incorpora nova estimativa:
- $$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) [\text{sample}]$$



Grid com penhasco



Réptil



Propriedades do Q-Learning

- Converge para a política ótima mesmo agindo de forma sub-ótima!
- Ressalvas:
 - Deve-se explorar suficientemente
 - Deve-se fazer a taxa de aprendizado pequena o suficiente mas não decrescer muito rapidamente

Q-Learning

Incialize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Repita:

execute ação aleatória a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma \max_{a'} Q[s', a'])$$
$$s \leftarrow s'$$

Q-Learning

Incialize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

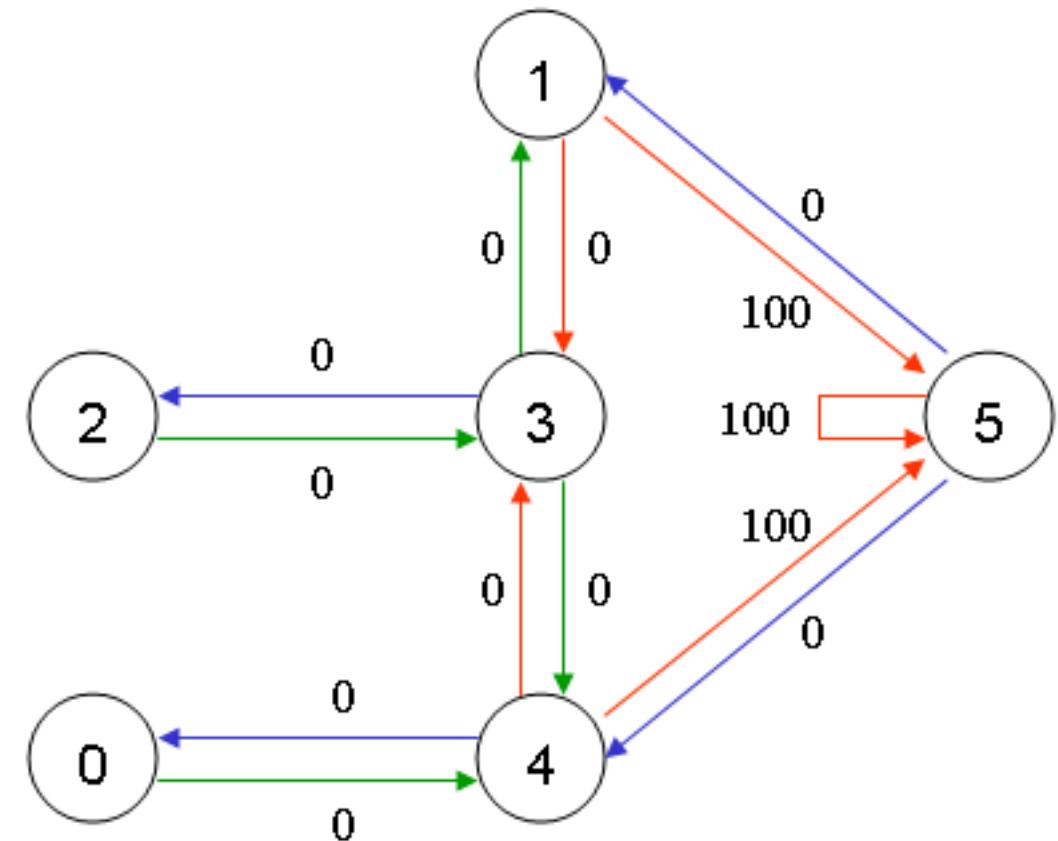
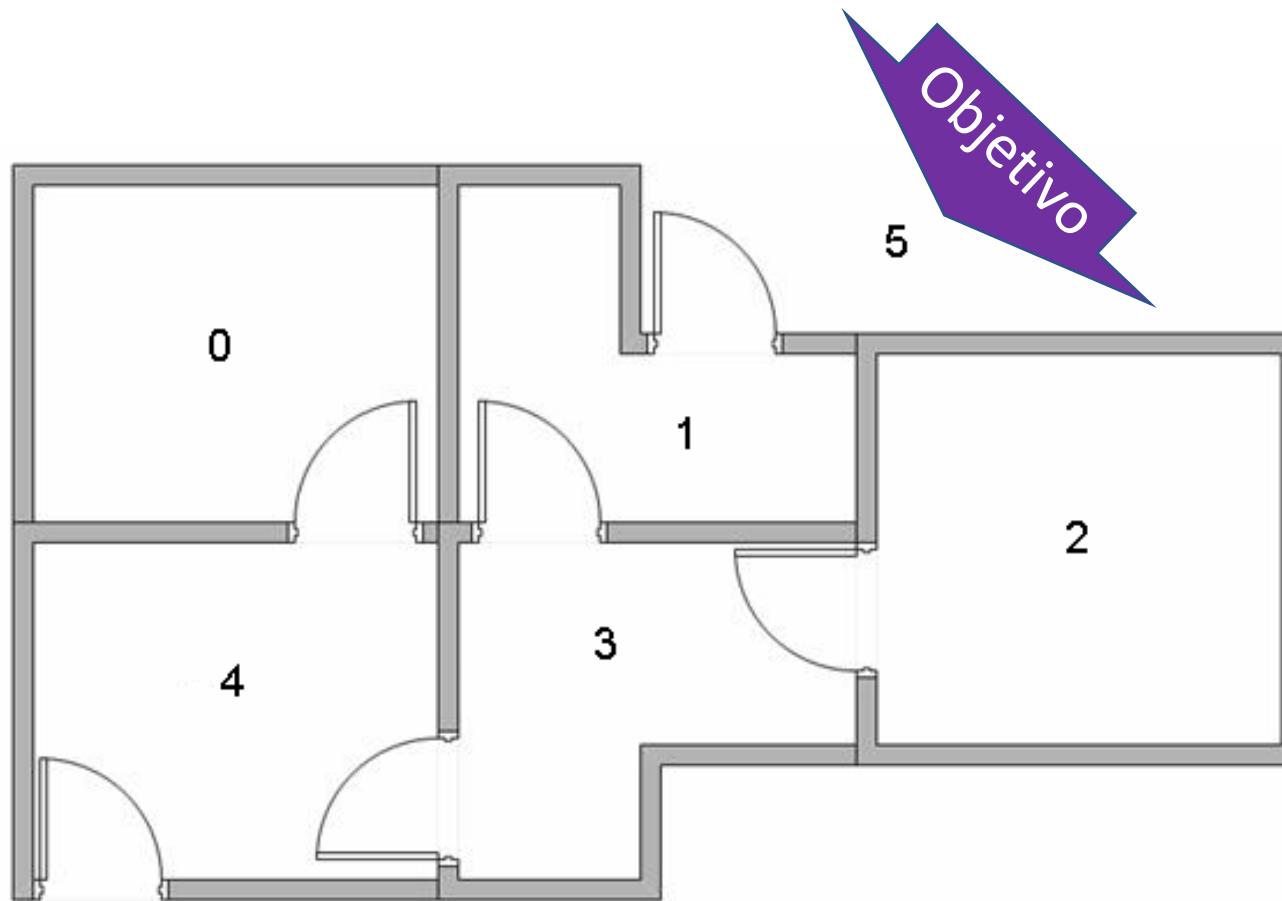
Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma \max_a Q[s', a])$$
$$s \leftarrow s'$$



Q-Learning

Inicialize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

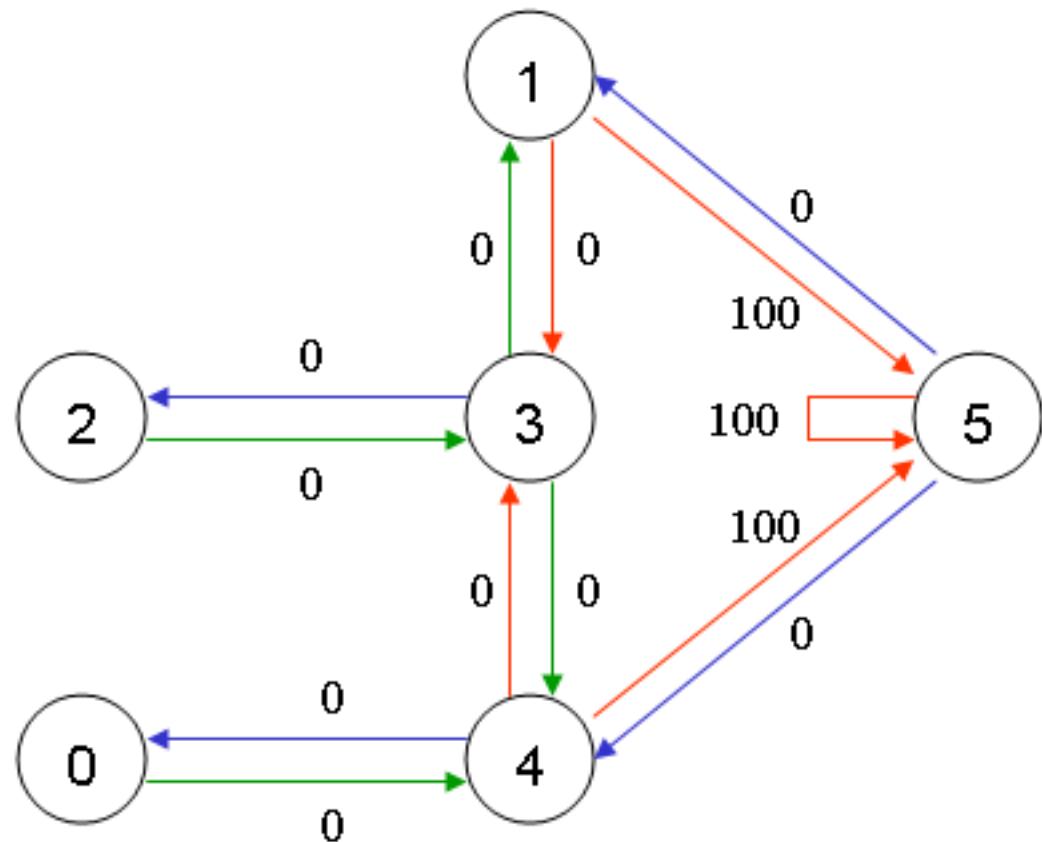
Para cada episódio:

Selecionar um estado s aleatoriamente

Repetir:

execute ação a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma \max_a Q[s', a])$$
$$s \leftarrow s'$$



$$R = \begin{matrix} & \text{Ação} \\ \text{Estado} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} \emptyset & \emptyset & \emptyset & \emptyset & 0 & \emptyset \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & \emptyset \\ \emptyset & 0 & 0 & \emptyset & 0 & \emptyset \\ 0 & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & 0 & \emptyset & \emptyset & 0 & 100 \end{bmatrix} \end{matrix}$$

Q-Learning

$$\gamma = 1, \alpha = 0.8$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

$$s = 1$$

$$\begin{aligned} Q[1,5] &= 0.2 * Q[1,5] + 0.8 * (R[1,5,5] + 1 * \max \{ Q[5,1], Q[5,4], Q[5,5] \}) \\ &= 0 + 0.8 * (100 + 1 * 0) = 80 \end{aligned}$$

$s = 5$ ← Estado objetivo → encerra episódio!

Inic平ize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s,a,s') + \gamma \max_a Q[s', a])$$

$$s \leftarrow s'$$

Estado	0	1	2	3	4	5
0	\emptyset	\emptyset	\emptyset	\emptyset	0	\emptyset
1	\emptyset	\emptyset	\emptyset	0	\emptyset	100
2	\emptyset	\emptyset	\emptyset	0	\emptyset	\emptyset
3	\emptyset	0	\emptyset	\emptyset	0	\emptyset
4	0	\emptyset	\emptyset	\emptyset	\emptyset	100
5	\emptyset	0	\emptyset	\emptyset	0	100

Q-Learning

$$\gamma = 1, \alpha = 0.8$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

$$s = 3$$

$$\begin{aligned} Q[3,1] &= 0.2 * Q[3,1] + 0.8 * (R[3,1,1] + 1 * \max \{ Q[1,3], Q[1,5] \}) \\ &= 0 + 0.8 * (0 + 1 * 80) = 64 \end{aligned}$$

$$s = 1$$

Inic平ize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s,a,s') + \gamma \max_a Q[s', a])$$

$$s \leftarrow s'$$

Estado	0	1	2	3	4	5
0	\emptyset	\emptyset	\emptyset	\emptyset	0	\emptyset
1	\emptyset	\emptyset	\emptyset	0	\emptyset	100
2	\emptyset	\emptyset	\emptyset	0	\emptyset	\emptyset
3	\emptyset	\emptyset	\emptyset	0	\emptyset	\emptyset
4	0	\emptyset	\emptyset	\emptyset	\emptyset	100
5	\emptyset	\emptyset	\emptyset	\emptyset	0	100

Q-Learning

$$\gamma = 1, \alpha = 0.8$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 64 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

$$s = 3$$

$$\begin{aligned} Q[1,3] &= 0.2 * Q[1,3] + 0.8 * (R[1,3,3] + 1 * \max \{ Q[3,1], Q[3,2], Q[3,4] \}) \\ &= 0.2 * 0 + 0.8 * (0 + 1 * 64) = 51.2 \end{aligned}$$

$$s = 3$$

Inic平ize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a , observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s,a,s') + \gamma \max_a Q[s', a])$$

$$s \leftarrow s'$$

Estado	0	1	2	3	4	5
0	\emptyset	\emptyset	\emptyset	\emptyset	0	\emptyset
1	\emptyset	\emptyset	\emptyset	0	\emptyset	100
2	\emptyset	\emptyset	\emptyset	0	\emptyset	\emptyset
3	\emptyset	0	0	\emptyset	\emptyset	\emptyset
4	0	\emptyset	\emptyset	\emptyset	\emptyset	100
5	\emptyset	0	\emptyset	\emptyset	0	100

Q-Learning

$$\gamma = 1, \alpha = 0.8$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 51 & 0 & 80 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

$$s = 3$$

$$\begin{aligned} Q[3,2] &= 0.2 * Q[3,2] + 0.8 * (R[3,2,2] + 1 * \max \{ Q[2,3] \}) \\ &= 0.2 * 0 + 0.8 * (0 + 1 * 0) = 0 \end{aligned}$$

$$s = 2$$

Inic平ize Q[S, A] de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a, observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha (R(s, a, s') + \gamma \max_a Q[s', a])$$

$$s \leftarrow s'$$

Ação

$$R = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} \emptyset & \emptyset & \emptyset & \emptyset & 0 & \emptyset \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & \emptyset \\ \emptyset & 0 & \emptyset & 0 & \emptyset & 0 \\ 0 & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & 0 & \emptyset & \emptyset & 0 & 100 \end{matrix} \right] \end{matrix}$$

Q-Learning

$$\gamma = 1, \alpha = 0.8$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \textcolor{violet}{51} & \textcolor{violet}{80} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \textcolor{violet}{640} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

$$s = 3$$

$$Q[2, \dots] = \dots$$

$$s = \dots$$

Inic平ize Q[S, A] de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Reita:

execute ação a, observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma \max_a Q[s', a])$$
$$s \leftarrow s'$$

Ação

$$R = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} \emptyset & \emptyset & \emptyset & \emptyset & 0 & \emptyset \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & \emptyset & \emptyset & 0 & \emptyset & \emptyset \\ \emptyset & 0 & 0 & \emptyset & 0 & \emptyset \\ 0 & \emptyset & \emptyset & 0 & \emptyset & 100 \\ \emptyset & 0 & \emptyset & \emptyset & 0 & 100 \end{matrix} \right] \end{matrix}$$

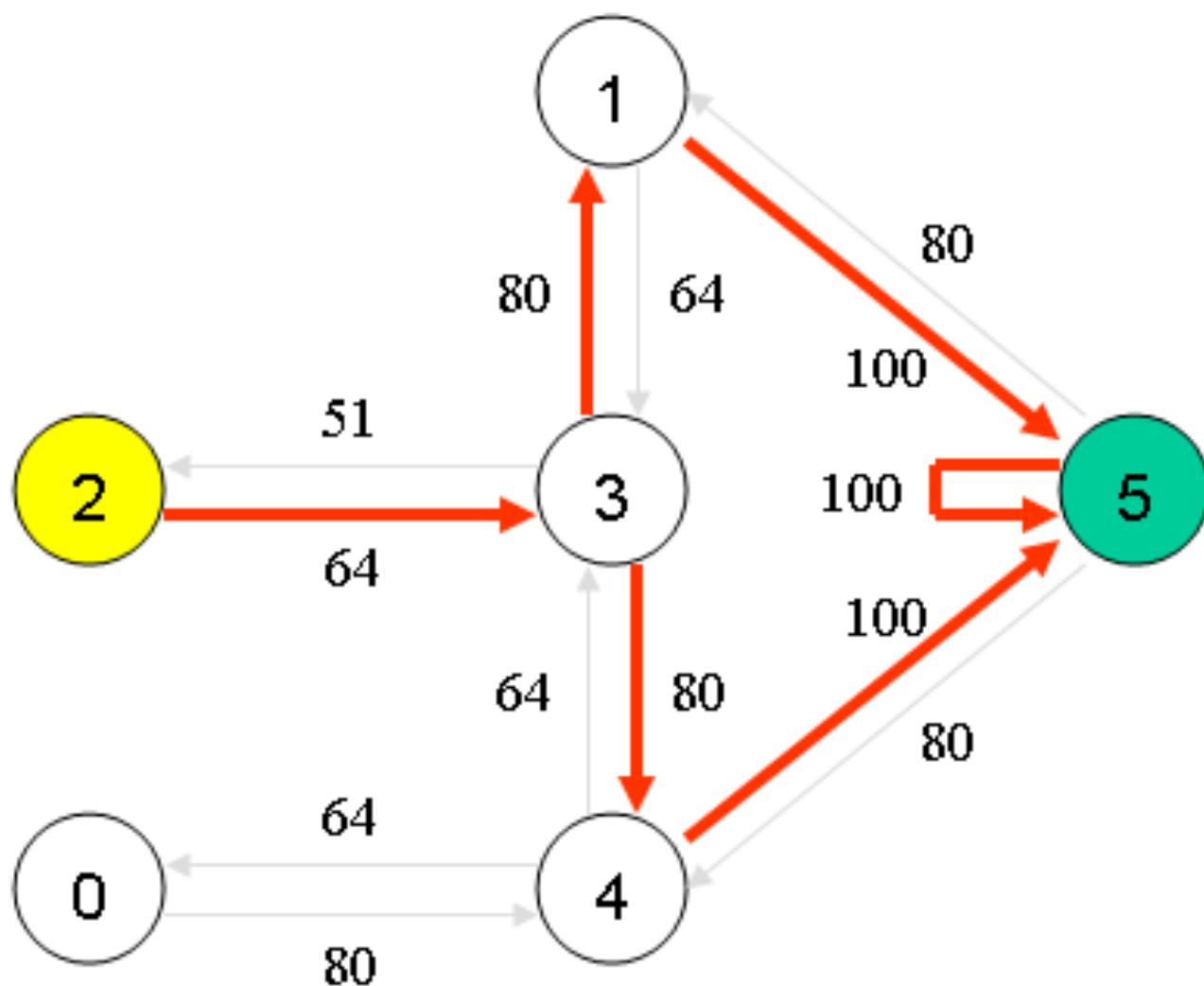
Q-Learning

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{matrix} \right] \end{matrix}$$

Normalizada....

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{matrix} \right] \end{matrix}$$

Política aprendida



Exploration vs. Exploitation

- O **Q-Learning** faz o agente aprender uma função que pode ser usada para determinar uma ação ótima. Mas o algoritmo não indica que ação de fato tomar
- O agente pode decidir de duas formas:
 1. *exploit*: estando em um estado s , selecionar uma ação que maximiza $Q[s, a]$; OU
 2. *explore*: selecionar outra ação qualquer. Na estratégia **ϵ -greedy**, por exemplo, escolhe-se uma ação aleatória com probabilidade ϵ ou a melhor ação do momento com probabilidade $1 - \epsilon$. ϵ pode ser diminuido progressivamente.

Q-Learning ϵ -greedy

Inicialize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado s aleatoriamente

Repita:

execute ação aleatória a

com probabilidade ϵ

OU execute melhor ação a

observe recompensa r e novo estado s' (amostra)

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma \max_{a'} Q[s', a'])$$

$$s \leftarrow s'$$

SARSA

Initialize $Q[S, A]$ de forma arbitrária (ex: \emptyset)

Para cada episódio:

Seleciona um estado S aleatoriamente

Usa amostra mais completa
 $\langle s, a, r, s', a' \rangle$

Seleciona ação a com política ϵ -greedy

Repita:

execute ação a , observe recompensa r , novo estado S' e

seleciona ação a' com política ϵ -greedy (amostra) 

$$Q[s, a] \leftarrow (1-\alpha)Q[s, a] + \alpha(R(s, a, s') + \gamma Q[s', a'])$$

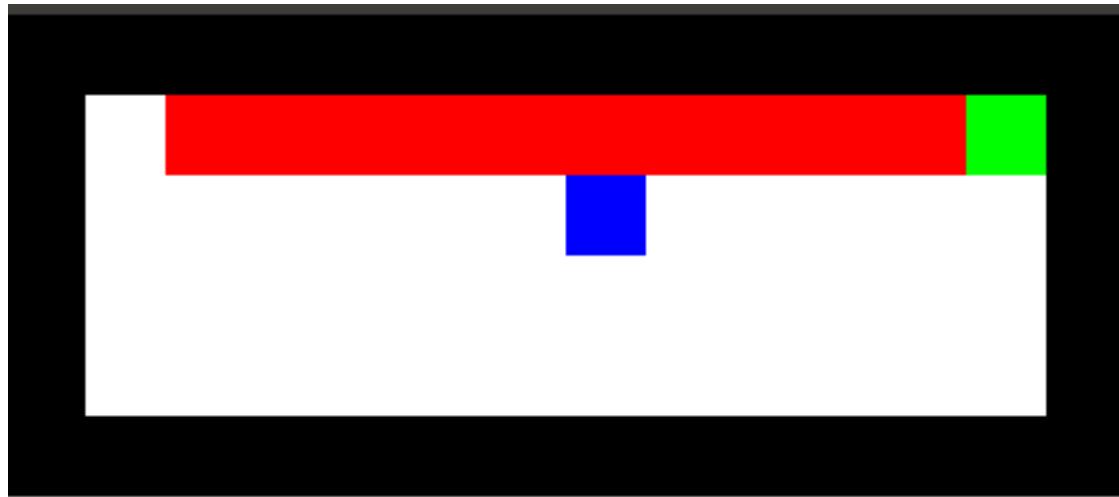
$$s \leftarrow s'$$

$$a \leftarrow a' \quad \text{Novo} \quad \text{Novo}$$

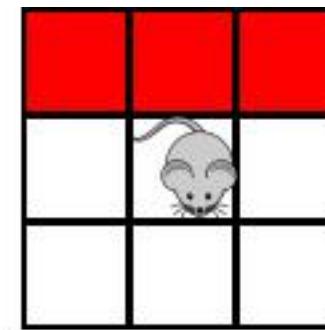
Q-Learning vs. SARSA

- Diferença está na forma como a recompensa futura é encontrada
 - Q-learning: melhor ação possível que pode ser tomada a partir do estado 2 e utiliza isso para atualizar o Q-value do estado 1.
 - SARSA: valor da ação real que já foi tomada no estado 2 é utilizada para atualizar Q-value do estado 1.
- Ou seja,
 - SARSA leva em consideração a política de controle pela qual o agente está se movendo e a incorpora na atualização dos valores de ação (ex: e-greedy ou "sempre pra esquerda", etc.)
 - Q-Learning simplesmente assume que uma política ótima está sendo seguida
- SARSA é útil quando queremos otimizar o valor de um agente que está explorando (*exploring*). Se desejamos fazer *offline learning* e então usar essa política em um agente que não explora, Q-learning é mais apropriado.

Exemplo: rato, abismo e queijo



State:



Action values:

W	NW	N	NE	E	SE	S	SW
	-50	-50	-50	+50			

Exemplo: rato, abismo e queijo

Q-Learning



SARSA



Q: Q-Learning

1. Considere a seguinte tabela Q[S,A]:

	State 1	State 2
Action 1	1.5	2.5
Action 2	4	3

<s,a,r,s'>

Assuma $\alpha=0.1$ e $\gamma=0.5$. Após amostra $<1,1,5,2>$, qual(is) valor(es) da tabela será(ão) atualizado(s) e qual(is) o(s) novo(s) valor(es)?

2. Qual é um problema potencial com um agente Q-Learning que sempre escolhe a ação que maximiza o Q-value?
3. Descreva duas formas de forçar que um agente Q-Learning execute exploration

Q: Q-Learning

1. Considere a seguinte tabela Q[S,A]:

	State 1	State 2
Action 1	1.5	2.5
Action 2	4	3

Assuma $\alpha=0.1$ e $\gamma=0.5$. Após amostra $<1,1,5,2>$, qual(is) valor(es) da tabela será(ão) atualizado(s) e qual(is) o(s) novo(s) valor(es)?

Resp:

$$\begin{aligned} Q[1,1] &= (1-\alpha)Q[1,1] + \alpha(R[1,1,2] + \gamma \max_{a'} Q[s',a']) \\ &= 0.9 * 1.5 + 0.1 * (5 + 0.5 * 3) \\ &= 2 \end{aligned}$$

Q: Q-Learning

1. ...
2. Qual é um problema potencial com um agente Q-Learning que sempre escolhe a ação que maximiza o Q-value?

Resp:

O agente pode ficar preso executando ações não-ótimas. Explorar (exploration) ações que não possuem o maior Q-value pode permitir encontrar uma melhor política.

Q: Q-Learning

1. ...
2. ...
3. Descreva duas formas de forçar que um agente Q-Learning execute *exploration*

Resp:

- (i) Usar estratégia ϵ -greedy (com $0 \leq \epsilon \leq 1$)
 - (ii) Inicializar os Q-values com valores altos. Isso encoraja a *exploration* porque estados não explorados terão valores altos e, assim, a busca greedy tenderá a explorá-los.
- "Otimismo diante da incerteza"

Nos localizando no curso...

- Unidade I: Busca monoagente
 - *Busca global*
 - *Busca local (otimização)*
- Unidade II: Busca multiagente e Planejamento
 - *Jogos com adversários e ambiente determinístico*
 - *Jogos com adversários e ambiente estocástico*
 - *Planejamento de ações com ambiente conhecido (MDP)*
 - *Planejamento de ações com ambiente desconhecido (Aprendizagem por reforço)*
- Unidade III: Conhecimento (preciso e incerto) e raciocínio
- Unidade IV: Aprendizado de máquina (*Machine Learning*)



Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>