Algoritmo Q-Learning

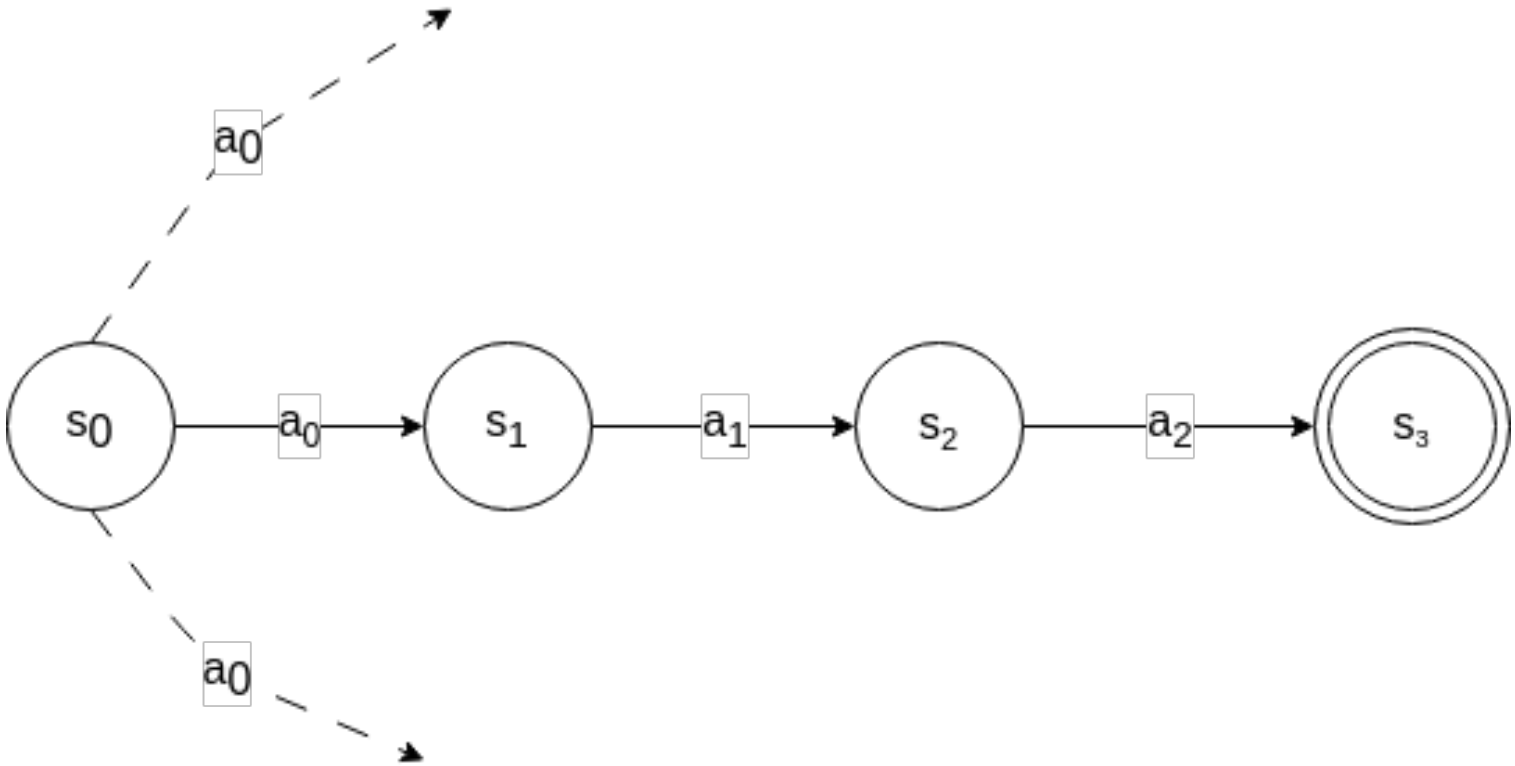
Fabrício Barth

Insper Instituto de Ensino e Pesquisa

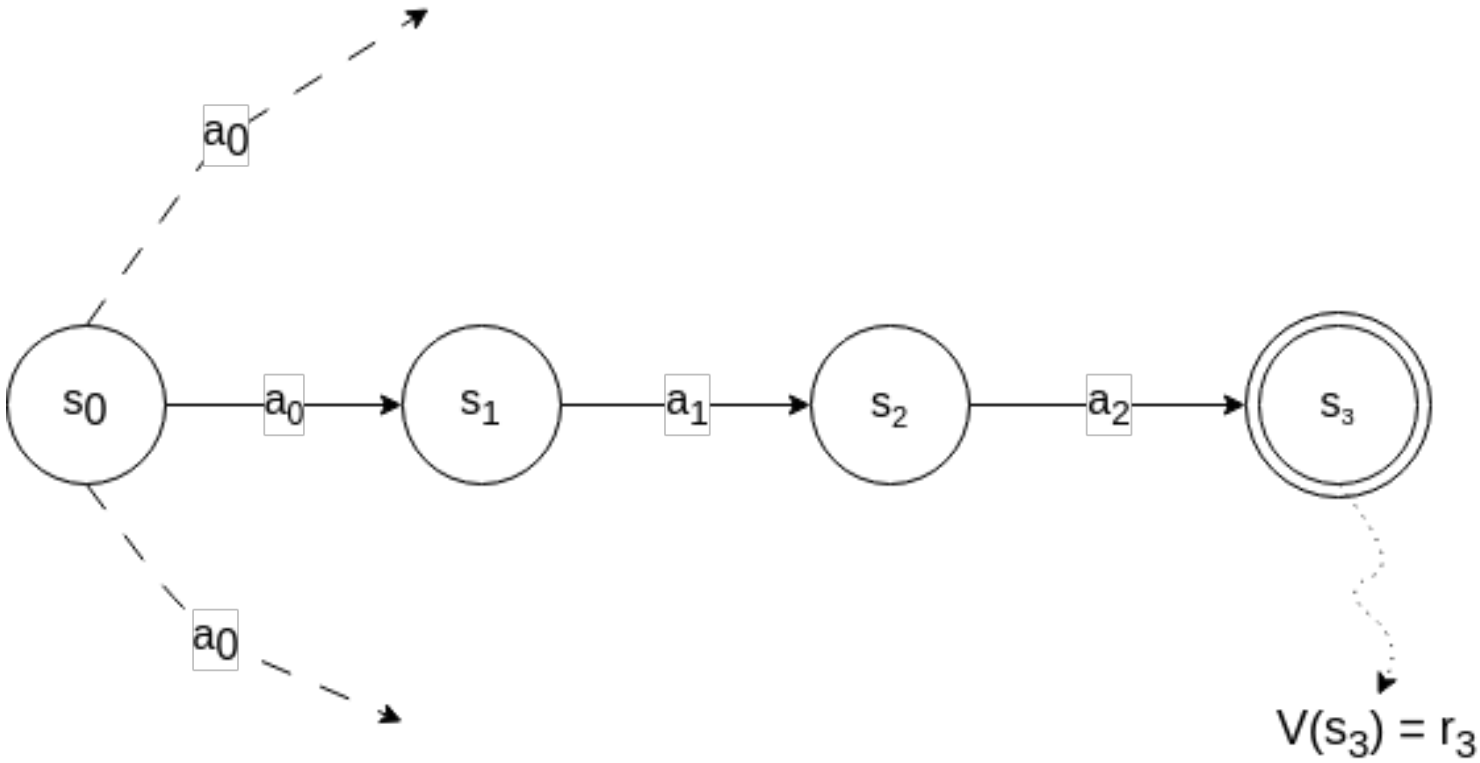
Fevereiro de 2025

Política de Controle

- A política de controle desejada é aquela que **maximiza** os reforços (*reward*) acumulados ao longo do tempo pelo agente.
- Em tese, é a política que faz o agente percorrer o **melhor caminho**.

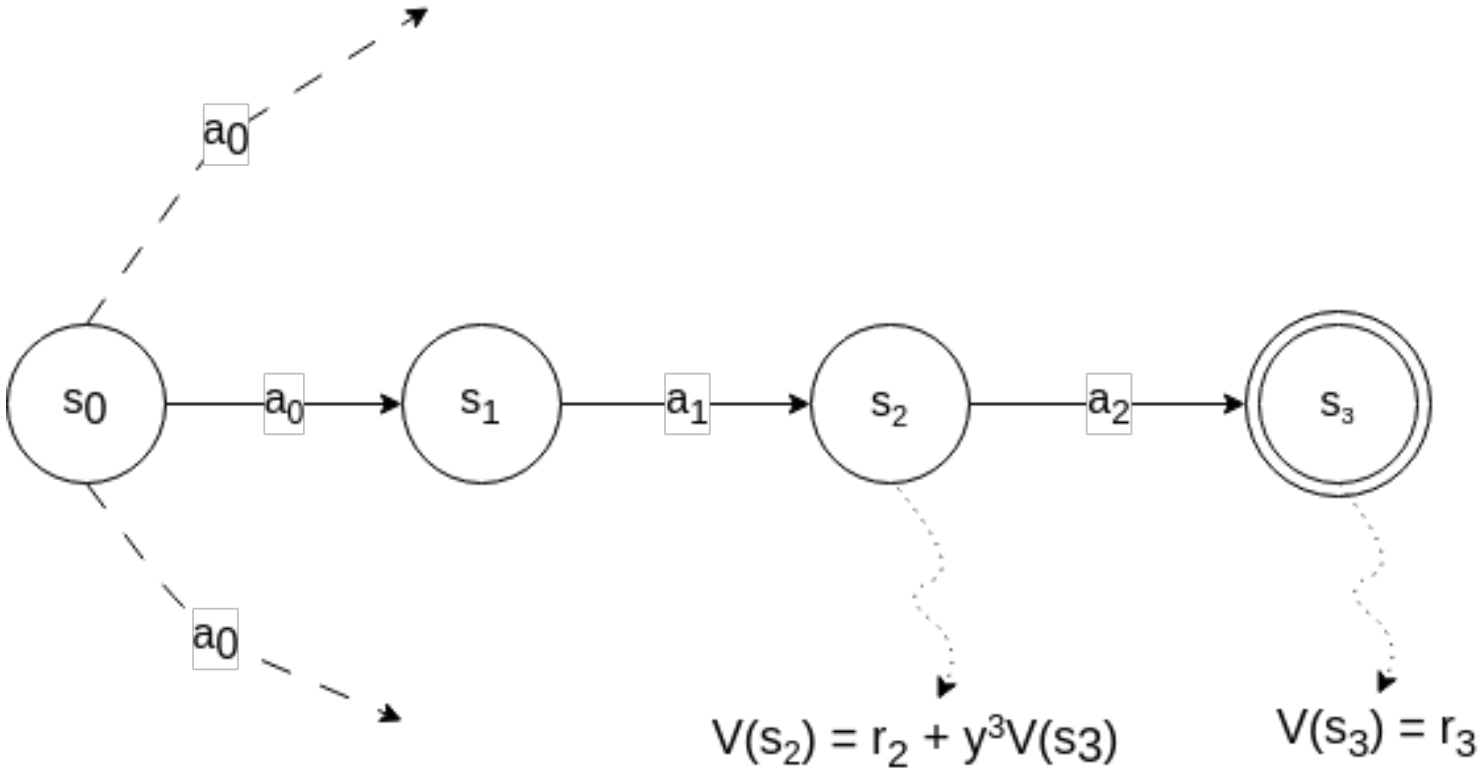


Reward acumulado (1/4)



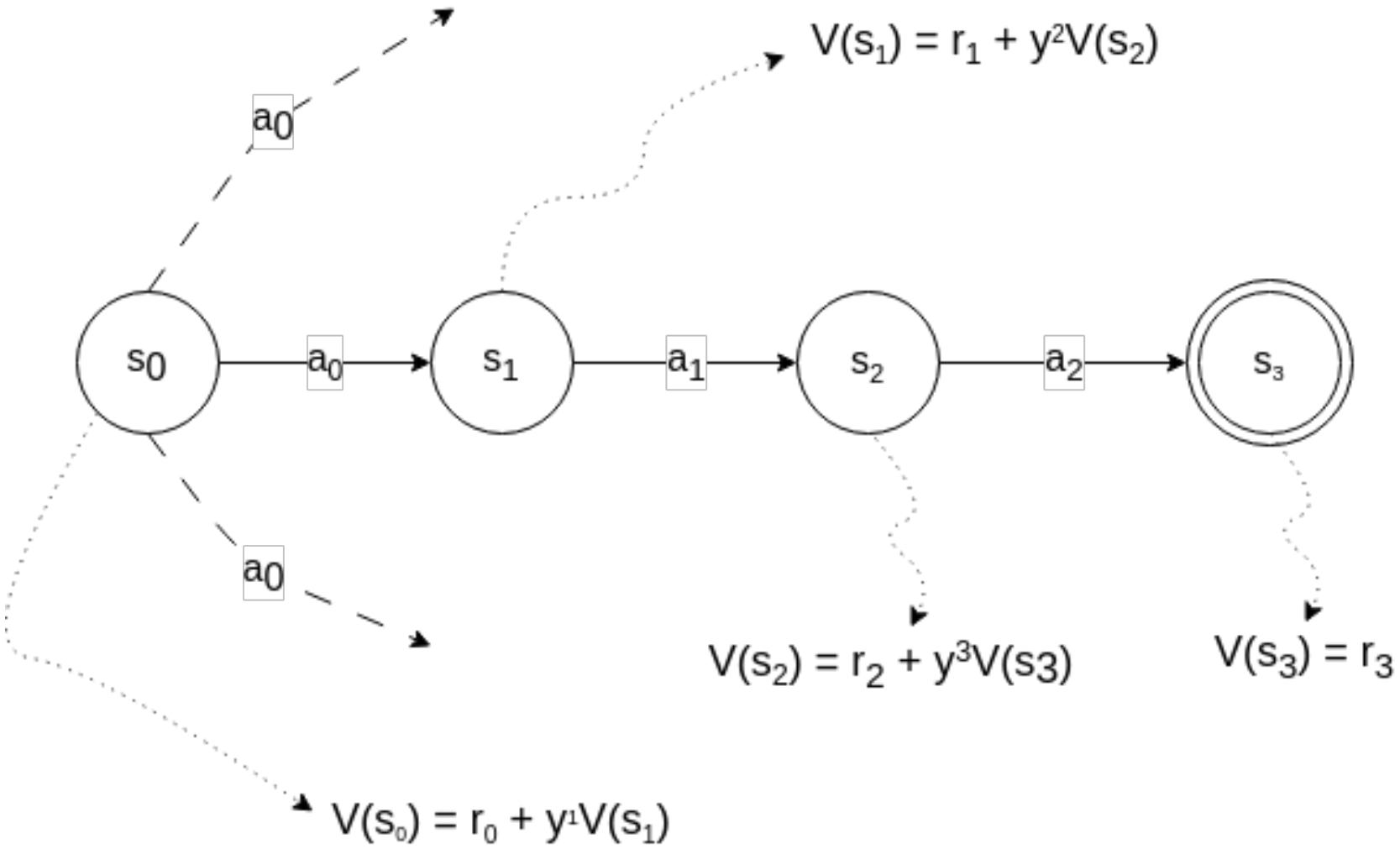
- O valor de um estado final leva-se em consideração apenas o reforço: $V(s_n) = r_n$.

Reward acumulado (2/4)



- O $V(s_2)$ será a soma de r_2 com o $V(s_3)$.
- Considerando o fator de desconto γ, temos: $V(s_2) = r_2 + \gamma^3 V(s_3)$.
- O fator de desconto: $0 \leq \gamma < 1$

Reward acumulado (3/4)



Reward acumulado (4/4)

Desta forma, temos:

$$
\begin{aligned}
V(s_0) &= r_0 + \gamma V(s_1) \\
V(s_0) &= r_0 + \gamma r_1 + \gamma^2 V(s_2) \\
V(s_0) &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V(s_3) \\
V(s_0) &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3
\end{aligned}
$$

Ou melhor:

$$V(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 \cdots + \gamma^n r_n$$

Fator de desconto γ

- O fator de desconto (γ) é um hiperparâmetro que consiste em um número entre 0 e 1 que define a importância das recompensas futuras em relação a atual ($0 \leq \gamma < 1$).
- Valores mais próximos ao 0 dão mais importância a recompensas imediatas enquanto os mais próximos de 1 tentarão manter a importância de recompensas futuras.

Algoritmo Q-Learning

- O algoritmo Q-Learning é um algoritmo do tipo *value-based* que estimam a expectativa de retorno de uma ação a sendo executada em um estado s de acordo com uma política π: $Q^{\pi}(s, a)$.
- Para que agente possa identificar uma política de controle ótima este agente precisa criar um **mapeamento** entre **estados** (S) e **ações** (A).
- Desta forma o agente consegue identificar qual é a ação a com maior retorno em um determinado estado s.

Algoritmo Q-Learning

- Este mapeamento é representado por uma função $Q(S, A)$ onde S são todos os estados possíveis $(s_1, s_2, \cdots)$ e onde A são todas as ações possíveis $(a_1, a_2, \cdots)$

Q-table	a_1	a_2	a_3	a_4
s_1				
s_2				
$\cdots$				
s_n				

Algoritmo Q-Learning

Como é que o agente pode saber quais são as melhores ações em cada estado?

Algoritmo Q-Learning

Como é que o agente pode saber quais são as melhores ações em cada estado?

- A ideia é fazer com que o agente aprenda a função de mapeamento $Q(S, A)$. Ou seja, que seja capaz de identificar qual é a melhor ação para cada estado através das suas **experiências**.
- *Testando* **infinitas** vezes o ambiente. Ou seja, *testando* **muitas** vezes as combinações entre **estados** (S) e **ações** (A).

Algoritmo Q-Learning

function *Q-Learning(env, γ, α, episódios)*
inicializar os valores de $Q(s, a)$ arbitrariamente
for todos os episódios **do**
  inicializar s a partir de *env*
  repeat
    escolher uma ação a para um estado s
    executar a ação a
    observar a recompensa r e o novo estado s'
    $Q(s, a) \leftarrow$ atualizando a partir das experiências
    $s \leftarrow s'$
  until s ser um estado final
end for
return $Q(s, a)$

Poderíamos simplesmente: $Q(s, a) \leftarrow r$. Será que funciona?

Algoritmo Q-Learning

function *Q-Learning(env, γ, α, episódios)*
inicializar os valores de $Q(s, a)$ arbitrariamente
for todos os episódios **do**
  inicializar s a partir de *env*
  repeat
    escolher uma ação a para um estado s
    executar a ação a
    observar a recompensa r e o novo estado s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{A'} Q(s', A') - Q(s, a)]$
    $s \leftarrow s'$
  until s ser um estado final
end for
return $Q(s, a)$

O valor de $Q(s, a)$ não é simplesmente o valor imediato do r. Ele deve levar em consideração toda a trajetória (**Equação de Bellman**).

Algoritmo Q-Learning: hiperparâmetro α

- α é a taxa de aprendizado ($0 < \alpha \leq 1$), quanto maior, mais valor dá ao novo aprendizado.

Que ação escolher?

function Q-Learning(env, α, γ, episódios)
inicializar os valores de $Q(s, a)$ arbitrariamente
for todos os episódios **do**
  inicializar s a partir de *env*
  repeat
    escolher uma ação a para um estado s
    executar a ação a
    observar a recompensa r e o novo estado s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{A'} Q(s', A') - Q(s, a)]$
    $s \leftarrow s'$
  until s ser um estado final
end for
return $Q(s, a)$

Exploration vs *Exploitation*

- A política que o agente utiliza para escolher uma ação a para um estado s não interfere no aprendizado da *Q-table*.

Exploration vs *Exploitation*

- A política que o agente utiliza para escolher uma ação a para um estado s não interfere no aprendizado da *Q-table*.
- No entanto, para que o algoritmo *Q-learning* possa convergir para um determinado problema é necessário que o algoritmo visite pares de ação-estado muitas (infinitas) vezes.

Exploration vs *Exploitation*

- A política que o agente utiliza para escolher uma ação *a* para um estado *s* não interfere no aprendizado da *Q-table*.
- No entanto, para que o algoritmo *Q-learning* possa convergir para um determinado problema é necessário que o algoritmo visite pares de ação-estado muitas (infinitas) vezes.
- Por isso, que a escolha de determinada *ação* em um *estado* poderia ser feita de forma **aleatória**.

Exploration vs *Exploitation*

- A política que o agente utiliza para escolher uma ação a para um estado s não interfere no aprendizado da *Q-table*.
- No entanto, para que o algoritmo *Q-learning* possa convergir para um determinado problema é necessário que o algoritmo visite pares de ação-estado muitas (infinitas) vezes.
- Por isso, que a escolha de determinada *ação* em um *estado* poderia ser feita de forma **aleatória**.
- Porém, normalmente se utiliza uma política que inicialmente escolhe aleatoriamente as ações, e, à medida que vai aprendendo, passa a utilizar cada vez mais as decisões determinadas pela política derivada de Q.

Exploration vs *Exploitation*

- A política que o agente utiliza para escolher uma ação a para um estado s não interfere no aprendizado da *Q-table*.
- No entanto, para que o algoritmo *Q-learning* possa convergir para um determinado problema é necessário que o algoritmo visite pares de ação-estado muitas (infinitas) vezes.
- Por isso, que a escolha de determinada *ação* em um *estado* poderia ser feita de forma **aleatória**.
- Porém, normalmente se utiliza uma política que inicialmente escolhe aleatoriamente as ações, e, à medida que vai aprendendo, passa a utilizar cada vez mais as decisões determinadas pela política derivada de Q.
- Esta estratégia inicia **explorando** (tentar uma ação mesmo que ela não tenha o maior valor de Q) e termina escolhendo a ação que tem o maior valor de Q (*exploitation*).

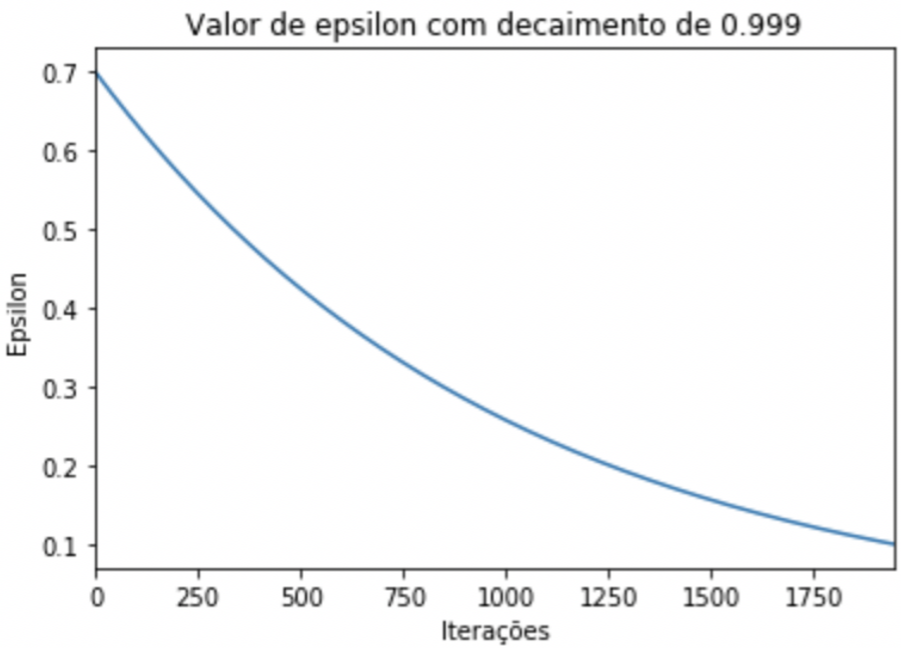
Exemplo de função para escolha de ações

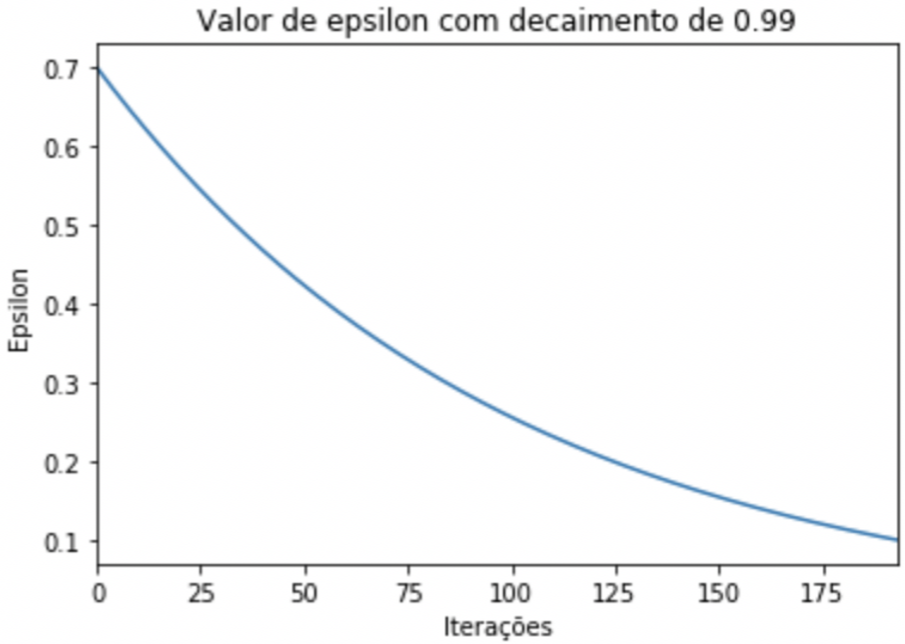
A escolha de uma ação para um estado é dada pela função:

function escolha(s, ϵ): a
rv = random $(0 < rv \leq 1)$
if $rv < \epsilon$ **then**
  return uma ação α aleatória em A
end if
return $\max_a Q(s, a)$

O fator de exploração ϵ $(0 \leq \epsilon \leq 1)$ inicia com um valor alto (0.7, por exemplo) e, conforme a simulação avança, diminiu: $\epsilon \leftarrow \epsilon \times \epsilon_{dec}$, onde $\epsilon_{dec} = 0.99$

Epsilon


Valor de epsilon com decaimento de 0.999
Epsilon
Iterações
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
250
500
750
1000
1250
1500
1750


Valor de epsilon com decaimento de 0.99
Epsilon
Iterações
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
25
50
75
100
125
150
175

Algoritmo Q-Learning

function *Q-Learning(env,* α*,* γ*,* ϵ*,* ϵ_{min}*,* ϵ_{dec}*, episódios)*
inicializar os valores de $Q(s, a)$ arbitrariamente
for todos os episódios **do**
  inicializar s a partir de *env*
  repeat
    $a \leftarrow escolha(s, \epsilon)$
    $s', r \leftarrow$ executar a ação a no *env*
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{A'} Q(s', A') - Q(s, a)]$
    $s \leftarrow s'$
  until s ser um estado final
  if $\epsilon > \epsilon_{min}$ **then** $\epsilon \leftarrow \epsilon \times \epsilon_{dec}$
end for
return Q

Atividade de implementação

Implementando o algoritmo Q-Learning

O objetivo desta atividade é implementar uma versão do algoritmo Q-Learning

Atividades

Siga o roteiro descrito em https://insper.github.io/rl/classes/05_q_learning/ ▸ Link

Atividade de implementação

Hiperparâmetros e seleção das ações

O objetivo desta atividade é compreender o funcionamento e impacto dos hiperparâmetros de α, γ e dos conceitos de *exploration* e *exploitation*.

Atividades

Siga o roteiro descrito em
https://insper.github.io/rl/classes/05_x_hyperparameters/ ▸ Link

Material de **consulta**

- Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA. **Capítulo 6.5**
- Watkins, C.J.C.H., Dayan, P. Q-Learning. Machine Learning 8, 279–292 (1992). ▸ Link
- Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, 2019.