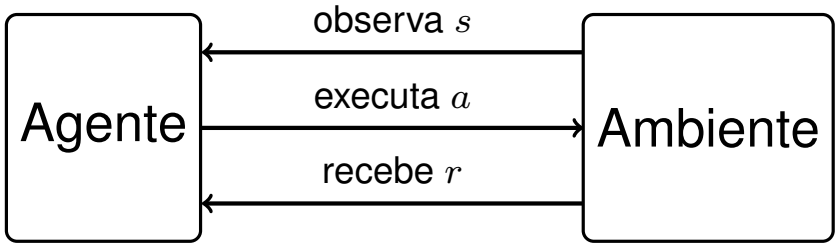
Aprendizado por Reforço
Reinforcement Learning

Valdinei Freire

(EACH - USP)

Problema de RL



Processo: em cada tempo t

- agente observa estado s_t
- agente escolhe ação a_t e atua no ambiente
- agente recebe recompensa r_t
- ambiente transita para um novo estado s_{t+1}
- tanto recompensa como transições são potencialmente estocástica e dependentes de s_t (também s_{t+1} para recompensa) e a_t, ou, em geral, o histórico completo

Problema de RL

Objetivo: buscar recompensas positivas e evitar recompensas negativas

Classe de Algoritmos:

- Model-Based
- Value-Based
- Policy Search

Formalização: objetivo do Agente

Considere o histórico $H = s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T$

1. Agente não-interativo
   - agente recebe um histórico H e deve determinar uma política (quasi-)ótima π^*
   - agente não interfere no ambiente (outra entidade atua arbitrariamente no ambiente)
2. Agente Interativo
   - agente atua no ambiente e interfere no histórico H
   - *on-line learning*: agente deve maximizar $\sum_{t=0}^{T} r_t$
   - *active learning*: agente deve obter política (quasi-)ótima no menor tempo T

Formalização: episódios

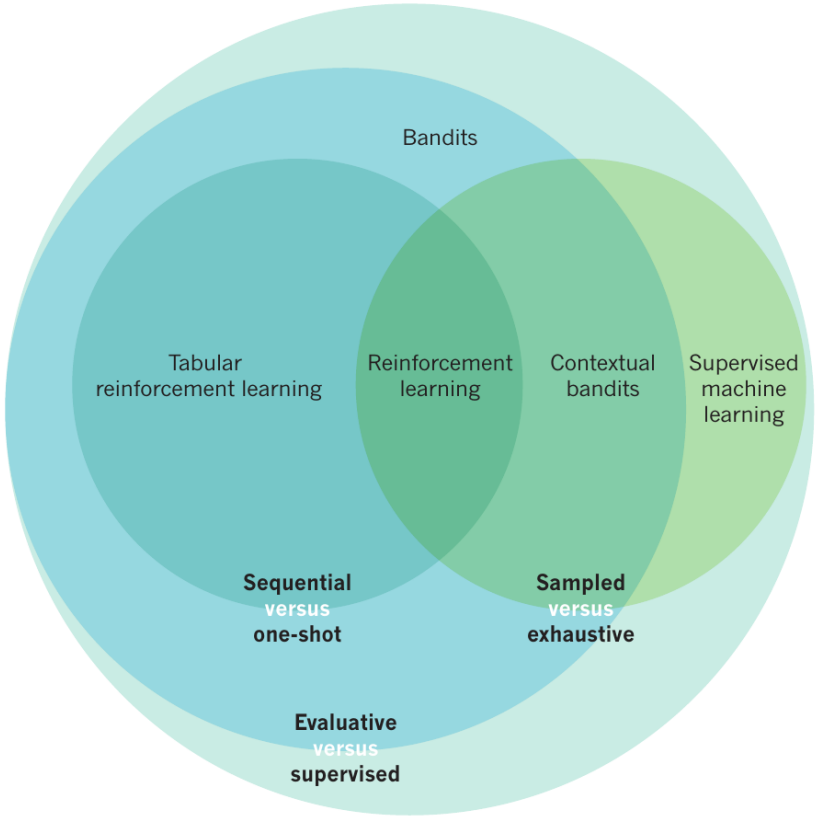
O histórico H pode ser dividido em episódios

- Muitas vezes divisão natural do problema
  - atingi um estado meta
  - acaba um tempo disponível
  - atingi um estado proibido
- Independência entre episódios
  - as transições em todos episódios seguem a mesma distribuição de probabilidade
  - o agente aprendiz acumula informação de um episódio para o outro
- Divisão arbitrária
  - reinicia um episódio de tempos em tempos
  - alerta: qual é a interferência dessa reinicialização nos algoritmos

RL versus Aprendizado Supervisionado

1. Aprendizado Supervisionado
   - recebe como entrada um conjunto de pares ordenados (s_i, a_i) (exemplos da política ótima)
   - objetivo: generalizar o conhecimento para estados s_j desconhecidos
   - estratégia: função genérica e controle de *overfitting*
   - interativo (*active learning* com oráculo) ou não-interativo
2. Aprendizado por Reforço
   - recebe como entrada pares ordenados (H_i, u_i) (histórico e avaliação)
   - escolha do agente interfere nas escolhas futuras
   - objetivo: descobrir melhor forma de agir
   - estratégia: tentativa e erro e generalização
   - dilema: exploração vs explotação

RL versus Aprendizado Supervisionado



Michael L. Littman. Reinforcement learning improves behaviour from evaluative feedback, Nature, 2015

Tomada de Decisões sob Incertezas

Modelo do Ambiente Desconhecido	Multi-Armed Bandits	Reinforcement Learning
Modelo do Ambiente Conhecido	Decision Theory	Markov Decision Process
	Ações não muda o estado do mundo	Ações mudam o estado do mundo

RL versus Algoritmos Genéticos

1. Algoritmos Genéticos
   - cromossomos equivalem às políticas π
   - função de *fitness*: soma de recompensa esperada

$$V^{\pi} = \mathrm{E}_{s_0 \sim \beta} \left[\sum_{t=0}^{T} r_t \middle| \pi \right]$$

   - Dilema Exploração-Explotação
     - seleção
     - mutação
     - *crossover*
   - Atribuição de Crédito
     - média de várias execuções? quantas?
     - armazenar estimativas?

2. Aprendizado por Reforço
   - Dilema Exploração-Explotação
     - algumas soluções triviais
     - ϵ-greedy
     - distribuição de Boltzman
     - existem soluções mais elaboradas
   - Atribuição de Crédito
     - Princípio da Otimalidade de Bellman
     - Programação Dinâmica
     - Assume-se ambiente Markoviano

Processo Markoviano de Decisão

Markov Decision Process (MDP) é definido pela tupla $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$

- $\mathcal{S}$ é o conjunto de estados
- $\mathcal{A}$ é o conjunto de ações
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ é a função de transição
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ é a função recompensa

Processo: em cada tempo t

- o processo encontra-se no estado s_t
- uma ação a_t é escolhida
- a recompensa $r_t = R(s_t, a_t)$ é gerada
- o processo transita para um estado s' com distribuição $\Pr(s_{t+1} = s' | s_t = s, a_t = a) = T(s, a, s')$

Função Valor

No RL considere que o ambiente comporta-se como um MDP.

O agente conhece o conjunto de ações $\mathcal{A}$, mas NÃO conhece a função de transição T e a função recompensa R.

Definição O valor para todo $s \in \mathcal{S}$ ao seguir π é dado por:

$$V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{T} \gamma^{t} r_t \middle| s_0 = s, \pi\right],$$

onde $0 \leqslant \gamma < 1$ é o fator de desconto.

Avaliação de Política: Monte Carlo

Dada uma política π, como estimar a função V^π?

- executa política π até o tempo T
- observa histórico $H = s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T$
- defina para todo $t \in \{0, 1, \ldots, T-1\}$

$$V_t^\pi \leftarrow \sum_{i=t}^{T} \gamma^{i-t} r_i$$

- para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$V^\pi(s) \leftarrow \frac{\sum_{\{t|s_t=s\}} V_t^\pi}{|\{t|s_t = s\}|}$$

Stochastic Approximation

Seja $H(y|x) = \Pr(Y_x \leqslant y)$ uma família de funções distribuição de probabilidade que correspondem a valores $x \in \mathbb{R}$. Defina:

$$M(x) = \mathrm{E}[Y_x] = \int_{-\infty}^{\infty} y \; dH(y|x).$$

Considere que se possa obter amostras de Y_x para x arbitrários. Deseja-se encontrar x tal que:

$$M(x) = \beta.$$

Dado um x_0 arbitrário, Robbins e Monro propõem o seguinte método:

$$x_{t+1} \leftarrow x_t + \alpha_t(\beta - y_t)$$

Stochastic Approximation

Exemplo: Média Incremental

Considere amostras $r_t \sim Z$ e considere $Y_x = x - Z$ e $\beta = 0$.

Então:

$$M(x) = \beta \Rightarrow x = \mathsf{E}[Z]$$

e

$$x_{t+1} \leftarrow x_t + \alpha_t(r_t - x_t)$$

Stochastic Approximation

Condições

Esperança e variância de Y_x são limitadas, isto é,

$$|\mathrm{E}[Y_x]| \leqslant C < \infty \quad \text{e} \quad \mathrm{Var}[Y_x] \leqslant \sigma^2 < \infty.$$

Existe $\theta \in \mathbb{R}$ tal que $M(\theta) = \beta$.

Para $x < \theta$, temos que $M(x) < \beta$.

Para $x > \theta$, temos que $M(x) > \beta$.

Stochastic Approximation

Considere que:

1. $\sum_{t}^{\infty} \alpha_t = \infty$ (decaimento não tão rápido)

2. $\sum_{t}^{\infty} \alpha_t^2 < \infty$ (decaimento não tão devagar)

Então o método de Robbins e Monro converge em probabilidade, isto é,

$$\lim_{t \to \infty} \mathsf{E}[(x_t - \theta)^2] = 0$$

Avaliação de Política: TD

Dada uma política π, como estimar a função valor V^{π}?

Temporal Difference:

- inicializa função valor V^{π} arbitrariamente
- em cada tempo t executa política π
- observa transição de s_t para s_{t+1} e recompensa r_t
- atualiza função valor $V^{\pi}(s_t)$

Avaliação de Política: TD

Utilizando Programação Dinâmica, pode-se descrever V^π como:

$$\begin{aligned} V^\pi(s) &= R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s,\pi(s),s') V^\pi(s') \\ &= \mathrm{E}_{s' \sim T(s,\pi(s),s')} \left[R(s,\pi(s)) + \gamma V^\pi(s') \right] \end{aligned}$$

Utilizando Aproximação Estocástica:

$$\delta_t \leftarrow r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$$

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha_t \delta_t$$

Algoritmo TD(0)

1. Faça $t = 0$
2. Inicialize $V(s)$ arbitrariamente
3. Repita para todo tempo t:
   (a) observe estado s_t
   (b) escolha ação $a_t = \pi(s_t)$ e atue no ambiente
   (c) receba recompensa r_t
   (d) perceba o próximo estado s_{t+1}
   (e) Faça:

$$\delta_t \leftarrow r_t + \gamma V(s_{t+1}) - V(s_t)$$

   (f) Faça:

$$V(s_t) \leftarrow V(s_t) + \alpha_t \delta_t$$

Contração

Um operador $J : \mathcal{C} \rightarrow \mathcal{C}$ é uma contração se existe $A < 1$ tal que:

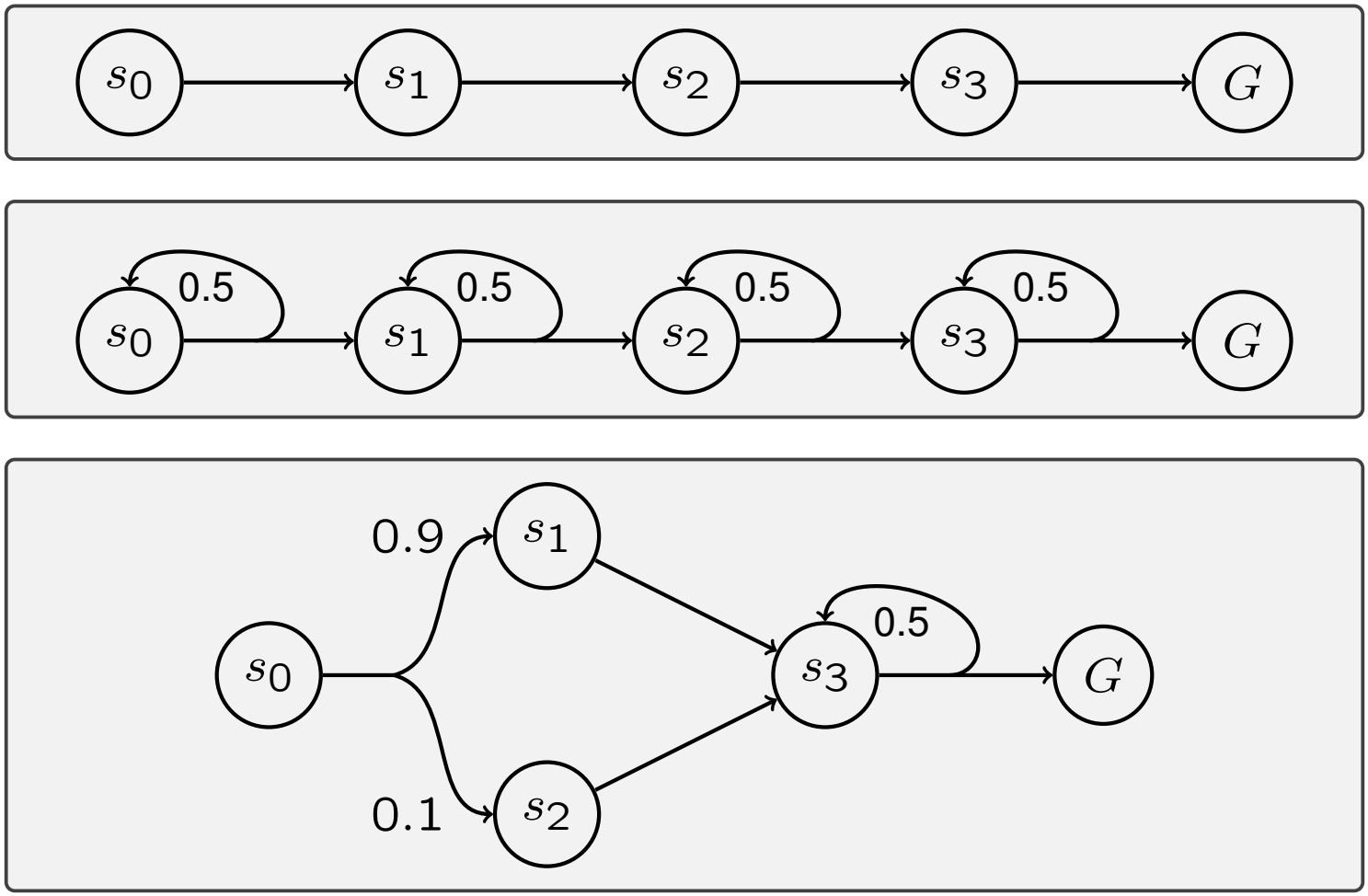
$$||JV_1 - JV_2||_\infty \leqslant A\,||V_1 - V_2||_\infty$$

TD(0) utiliza um operador que tem a propriedade de contração (notação vetorial):

$$||(\mathbf{R}^\pi + \gamma \mathbf{T}^\pi \mathbf{V}) - (\mathbf{R}^\pi + \gamma \mathbf{T}^\pi \mathbf{V}^\pi)||_\infty \leqslant \gamma\,||\mathbf{V} - \mathbf{V}^\pi||_\infty$$

Juntando a propriedade de contração e o método de Robbins e Monro pode-se provar que TD(0) converge.

Diferença Temporal x Monte-Carlo



Diferença Temporal x Monte-Carlo

Diferença Temporal:

- não necessita guardar histórico
- atualização ocorre em cada iteração
- a melhora da estimativa do valor de um estado influi na melhoria do valor de outro estado

Monte-Carlo:

- atribuição de crédito ocorre logo na primeira passagem pelo estado

Como obter a vantagem de ambos?

Algoritmo TD(λ)

Algoritmo TD(λ)

1. Faça $t = 0$
2. Inicialize $V(s)$ arbitrariamente e $e(s) = 0$
3. Repita para todo tempo t:
   (a) observe estado s_t
   (b) escolha ação $a_t = \pi(s_t)$ e atue no ambiente
   (c) receba recompensa r_t
   (d) perceba o próximo estado s_{t+1}
   (e) Faça: $\delta_t \leftarrow r_t + \gamma V(s_{t+1}) - V(s_t)$
   (f) para todo estado $s \in \mathcal{S}$ faça:

$$e(s) \leftarrow \begin{cases} \gamma\lambda e(s) + 1 & , \text{ se } s = s_t \\ \gamma\lambda e(s) & , \text{ caso contrário} \end{cases}$$

$$V(s) \leftarrow V(s) + \alpha_t e(s) \delta_t$$

Algoritmo TD(λ)

Temos:

$$\begin{aligned}\delta_0 &= r_0 + \gamma V(s_1) - V(s_0)\\ \delta_1 &= r_1 + \gamma V(s_2) - V(s_1)\\ \delta_2 &= r_2 + \gamma V(s_3) - V(s_2)\\ &\dots\end{aligned}$$

Para o estado s_0 temos:

$$\begin{aligned}V(s_0) \leftarrow V(s_0) \quad & +\alpha\gamma^0\lambda^0[r_0 + \gamma V(s_1) - V(s_0)]\\ & +\alpha\gamma^1\lambda^1[r_1 + \gamma V(s_2) - V(s_1)]\\ & +\alpha\gamma^2\lambda^2[r_2 + \gamma V(s_3) - V(s_2)]\\ & + \dots\end{aligned}$$

Se $\lambda = 1$, então TD(λ) comporta-se como Monte Carlo, pois:

$$V(s_0) \leftarrow V(s_0) + \alpha[r_0 + \gamma r_1 + \dots + \gamma^{T-1} r_{T-1} + \gamma^T V(s_T) - V(s_0)]$$

Controle Ótimo

Função Valor-Ação

O valor para todo $s \in \mathcal{S}$ ao executar ação a e então seguir a política ótima π^*:

$$Q(s,a) = \mathrm{E}\left[\sum_{t=0}^{T} \gamma^t r_t \middle| s_0 = s, a_0 = a, a_{t>0} = \pi^*\right].$$

Utilizando Programação Dinâmica, pode-se descrever $Q(s,a)$ como:

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \max_{a' \in \mathcal{A}} Q(s',a') \\ &= \mathrm{E}_{s' \sim T(s,a,s')}\left[R(s,a) + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\right] \end{aligned}$$

Algoritmo Q-Learning

1. Faça $t = 0$
2. Inicialize $Q(s, a)$ arbitrariamente
3. Repita para todo tempo t:
   (a) observe estado s_t
   (b) escolha ação a_t e atue no ambiente
   (c) receba recompensa r_t
   (d) perceba o próximo estado s_{t+1}
   (e) Faça:

$$\delta_t \leftarrow r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t)$$

   (f) Faça:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \delta_t$$

Dilema Exploração-Explotação

ϵ-greedy

- explora uniformemente com probabilidade ϵ
- explota com probabilidade $1 - \epsilon$

$$a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$$

- ϵ deve decair com o tempo

Dilema Exploração-Explotação

distribuição de Boltzmann (SoftMax)

- Escolhe a ação $a_t = a$ com probabilidade

$$p_a = \frac{e^{\tau Q(s_t,a)}}{\sum_{a' \in \mathcal{A}} e^{\tau Q(s_t,a')}}$$

- $\tau \geqslant 0$ é um parâmetro de temperatura

- τ deve aumentar com o tempo

Convergência

Condições de Convergência:

1. Todos pares estado-ação (s, a) devem ser visitados infinitamente

2. $\sum_{t}^{\infty} \alpha_t = \infty$ (decaimento não tão rápido)

3. $\sum_{t}^{\infty} \alpha_t^2 < \infty$ (decaimento não tão devagar)

Algoritmo SARSA(λ)

1. Faça $t = 0$
2. Inicialize $Q(s, a)$ arbitrariamente e $e(s, a) = 0$
3. Repita para todo tempo t:
   (a) observe estado s_t
   (b) escolha ação a_t e atue no ambiente
   (c) receba recompensa r_t
   (d) perceba o próximo estado s_{t+1}
   (e) escolha a próxima ação a_{t+1}
   (f) Faça:

$$\delta_t \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

   (g) para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$e(s, a) \leftarrow \begin{cases} \gamma\lambda e(s, a) + 1 & , \text{ se } s = s_t \text{ e } a = a_t \\ \gamma\lambda e(s, a) & , \text{ caso contrário} \end{cases}$$

   (h) para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$Q(s, a) \leftarrow Q(s, a) + \alpha_t e(s, a)\delta_t$$

On-Policy e Off-Policy

- On-Policy: função valor-ação aprendida depende da política executada
- Off-Policy: função valor-ação aprendida independe da política executada
- Exemplo: *Cliff Walk*

s_0	penhasco - retorna ao início								G

On-Policy e Off-Policy

SARSA(λ) é on-policy e Q-Learning é off-policy

Como criar uma versão SARSA(λ) off-policy?

Algoritmo Q(λ) de Watkin: propaga δ_t para estados anteriores enquanto política ótima for escolhida.

Algoritmo Q(λ)

1. Faça $t = 0$
2. Inicialize $Q(s, a)$ arbitrariamente e $e(s, a) = 0$
3. Repita para todo tempo t:
   (a) observe estado s_t
   (b) escolha ação a_t e atue no ambiente
   (c) receba recompensa r_t
   (d) perceba o próximo estado s_{t+1}
   (e) Faça:

$$\delta_t \leftarrow r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t)$$

   (f) para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$e(s, a) \leftarrow \begin{cases} \gamma\lambda e(s, a) + 1 & , \text{ se } s = s_t \text{ e } a = a_t \\ \gamma\lambda e(s, a) & , \text{ caso contrário} \end{cases}$$

   (g) para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$Q(s, a) \leftarrow Q(s, a) + \alpha_t e(s, a) \delta_t$$

   (h) se $Q(s_t, a_t) \neq \max_a Q(s_t, a)$ (melhor ação NÃO foi escolhida), então para todo estado $s \in \mathcal{S}$ e ação $a \in \mathcal{A}$ faça:

$$e(s, a) \leftarrow 0$$

Referências

Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction Second Edition, MIT Press, Cambridge, MA, 2018

Csaba Szepesvári. Algorithms for Reinforcement Learning, Morgan & Claypool Publishers, 2009

Herbert Robbins e Sutton Monro. A Stochastic Approximation Method, Annals of Mathematical Statistics, 1951