Inteligência Artificial para Robótica Móvel

Aprendizado por Reforço Livre de Modelo

Professor: Marcos Maximo

Roteiro

- Motivação;
- Predição de Monte Carlo (MC);
- *Temporal Difference* (TD);
- Controle de MC;
- Sarsa;
- *Q-Learning*;
- Amostragem por Importância.

Motivação

Aprendizado por Reforço

- Relembrando:

MDP: (S, A, p, r, γ)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

Aprendizado por Reforço Baseado em Modelo

- Predição: avaliar uma política.
- Controle: determinar a política ótima.
- Métodos de programação dinâmica se baseiam nas equações de Bellman:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a) + \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_\pi(s')$$

$$v_*(s) = \max_{a \in A} \left(r(s,a) + \sum_{s' \in S} p(s'|s,a) v_*(s') \right)$$

Aprendizado por Reforço Livre de Modelo

- Entretanto, métodos de programação dinâmica requerem que a dinâmica $p(s'|s,a)$ do MDP seja conhecida.
- Em problemas complicados, a dinâmica geralmente não é conhecida…
- Qual a probabilidade do robô fazer gol se ele está a 3 m do gol e chuta a bola?
- Qual a probabilidade de um carro autônomo colidir se ele freiar a 5 m do obstáculo?
- É muito comum conseguirmos simular “amostras” da dinâmica, mas não saber $p(s'|s,a)$.

Aprendizado por Reforço Livre de Modelo

- É possível estimar o modelo experimentalmente:

$$p(s'|s,a) \approx \frac{N(s'|s,a)}{N(s,a)}$$

- É uma ideia válida…
- Também é possível avaliar/aprender política diretamente a partir da experiência.
- Técnicas que aprendem diretamente a partir da experiência são chamadas “livres de modelo”, em contraponto a técnicas “baseadas em modelo”.

Predição de Monte Carlo (MC)

Predição de Monte Carlo (MC)

- MC aprende diretamente de episódios de experiência.
- MC é livre de modelo: não necessita do modelo do MDP.
- MC usa uma ideia muito simples: valor = retorno médio.
- Desvantagem: MC é adequado apenas para tarefas episódicas.
- Métodos de Monte Carlo se baseiam em calcular valores numéricos através da simulação de experimentos estocásticos.

Avaliação de Política de Monte Carlo

- Objetivo: aprender v_π de episódios de experiência seguindo a política π:

$$\tau_\pi = S_0, A_0, R_1, S_1 A_1, \ldots, R_T, S_T \sim \pi$$

- Retorno:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

- Função valor:

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

- Esperança pode ser aproximada por uma média experimental.
- Muito interessante se temos um simulador!

Avaliação de Política de Monte Carlo de Primeira Visita

- Rodar uma trajetória τ seguindo política π até o **final** do episódio.
- Para avaliar cada estado s visitado:
- Considerar o passo de tempo t em que s é visitado pela **primeira** vez no episódio.
- Incrementar contador de visitas: $N(s) = N(s) + 1$.
- Acumular retorno: $S(s) = S(s) + G_t$.
- Valor estimado pelo retorno médio: $V(s) = S(s)/N(s)$.
- $V(s) \to v_\pi(s)$ quando $N(s) \to \infty$.

Avaliação de Política de Monte Carlo de Cada Visita

- Rodar uma trajetória τ seguindo política π.
- Para cada estado s visitado.
- Considerar **cada** passo de tempo t em que s é visitado no episódio.
- Incrementar contador de visitas: $N(s) = N(s) + 1$.
- Acumular retorno: $S(s) = S(s) + G_t$.
- Valor estimado pelo retorno médio: $V(s) = S(s)/N(s)$.
- $V(s) \to v_\pi(s)$ quando $N(s) \to \infty$.
- Qual é mais eficiente é um pouco dependente do problema...

Média Incremental

- Para evitar guardar tantas variáveis $N(s)$, $S(s)$ e $V(s)$, há um algoritmo incremental para calcular média.

$$\mu_k = \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{1}{k}\left(x_k + \sum_{i=1}^{k-1} x_i\right) = \frac{1}{k}(x_k + (k-1)\mu_{k-1}) \Rightarrow$$

$$\mu_k = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

$$\mu_k = \mu_{k-1} + \alpha(x_k - \mu_{k-1}), \alpha = 1/k$$

Atualização Monte Carlo Incremental

- Pode-se usar equação incremental para cálculo de $V(s)$:

$$N(s) = N(s) + 1$$

$$V(s) = V(s) + \frac{1}{N(s)}\big(G_t - V(s)\big)$$

- Na prática, é muito comum o uso de média móvel:

$$V(s) = V(s) + \alpha\big(G_t - V(s)\big)$$

- Média móvel é essencial no caso de problemas não-estacionários (MDP muda com o tempo).

Temporal-Difference (TD)

Temporal-Difference (TD)

- TD aprende diretamente da experiência.
- TD é livre de modelo: não necessita do modelo do MDP.
- Diferentemente de MC, TD necessita de *bootstrapping*, i.e. TD começa com estimativa inicial para $V(s)$ (assim como DP fazia).
- Diferentemente de MC, TD também funciona para tarefas continuadas.

Temporal-Difference (TD)

- Perceba que:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) \Rightarrow$$
$$G_t = R_{t+1} + \gamma G_{t+1}$$
$$v_\pi(S_t) = R_{t+1} + \gamma v_\pi(S_{t+1})$$

- Ideia de TD(0) consiste em aprimorar estimativa de $V(S_t)$ através da amostra de R_{t+1} e da estimativa de $V(S_{t+1})$:

$$V(S_t) = V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

- Chama-se $R_{t+1} + \gamma V(S_{t+1})$ de alvo TD (*TD target*).
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ é o erro TD.

Temporal-Difference (TD)

- Alvo TD(0):

$$V(S_t) = V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

- No caso, o “alvo” Monte Carlo é G_t:

$$V(S_t) = V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

MC x TD

- TD pode aprender antes de saber o resultado do final do episódio.
  - TD(0) só usa informação do passo, então pode aprender a cada passo.
  - MC necessita esperar até o final do episódio para calcular G_t.
- TD pode aprender **sem** o resultado final.
  - TD pode aprender de sequências incompletas.
  - MC só pode aprender de sequências completas.
  - TD funciona para ambientes continuados.
  - MC funciona apenas para ambientes episódicos.

Trade-off Bias/Variância

- Alvo MC $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$ é uma estimativa não-enviesada de $v_\pi(S_t)$.
- O "verdadeiro" alvo TD é não-enviesado: $R_{t+1} + \gamma v_\pi(S_{t+1})$.
- Alvo TD "real" $R_{t+1} + \gamma V(S_{t+1})$ é enviesado pois usa valor estimado (errado) $V(S_{t+1})$.
- Alvo TD tem uma variância muito menor que alvo MC:
  - G_t depende de muitas ações, transições e recompensas aleatórias.
  - Alvo TD depende de uma ação, uma transição e uma recompensa aleatórias.

MC x TD

- MC tem alta variância, mas não possui *bias*.
  - Boas propriedades de convergência.
  - Não depende de “chute” inicial.
- TD tem baixa variância, mas possui *bias*.
  - Em geral é mais eficiente que MC.
  - TD(0) converge para $v_\pi(s)$ (se representação não for aproximada).
  - Depende de “chute” inicial de $V(s)$.

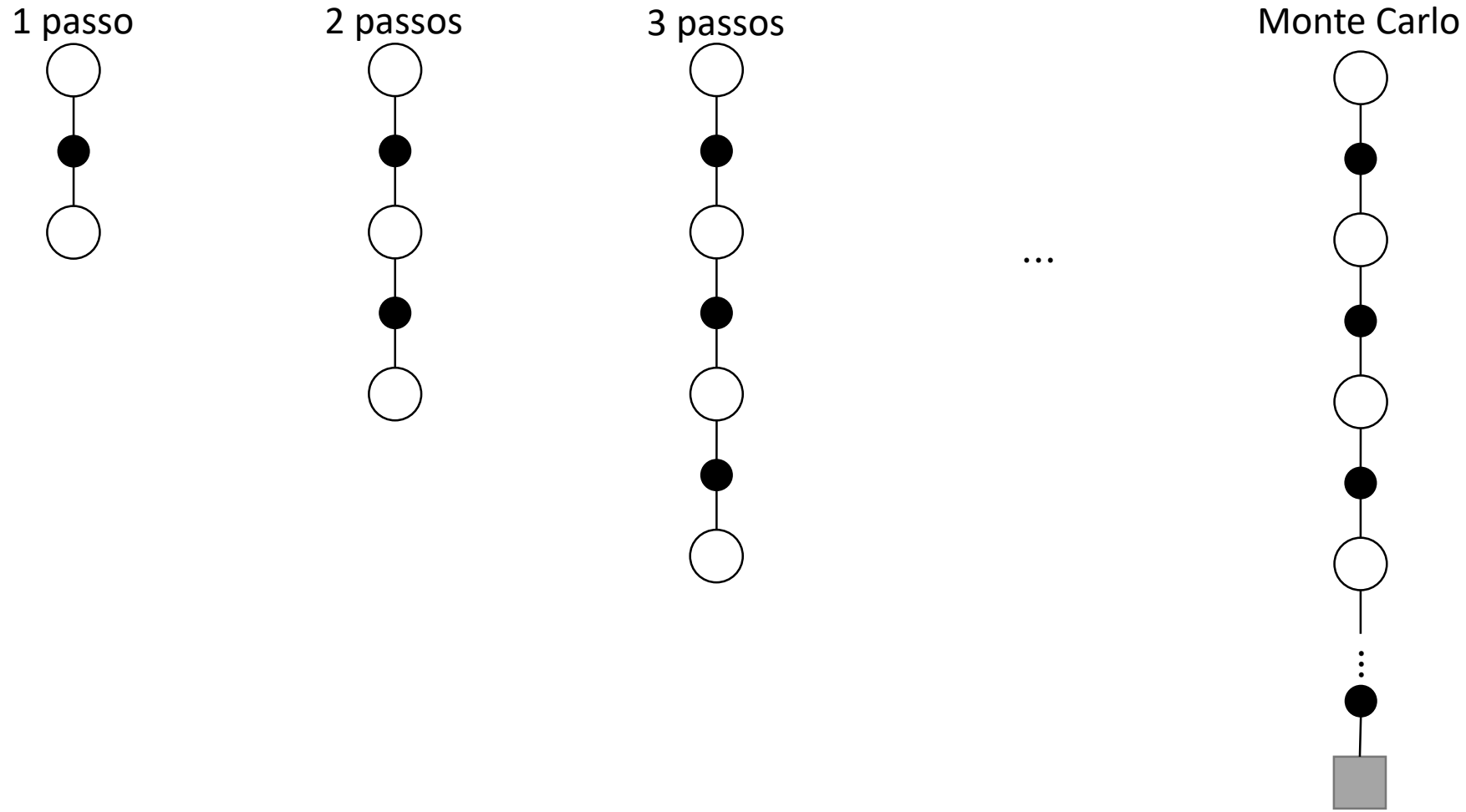
Predição de n Passos

- Por que ao invés de usar apenas 1 passo no TD, não usamos mais passos?

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma R_{t+2} + \gamma^2 G_{t+2}$$

- $n = 1$ (TD): $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$.
- $n = 2$: $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$.
- $n = 3$: $G_t^{(3)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})$.
- n: $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n V(S_{t+n})$.
- $n = \infty$ (MC): $G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$.

Predição de n Passos



TD de n Passos

- Retorno de n passos:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- TD de n passos:

$$V(S_t) = V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Retornos de n Passos

- Também é possível mesclar retornos com números diferentes de passos:

$$\frac{1}{2} G_t^{(2)} + \frac{1}{2} G_t^{(4)}$$

- Experimentos mostram que isso é uma boa ideia…
- Como combinar informação de todos os retornos $G_t^{(n)}$?

λ-*Return*

- λ-*return* combina infomações de todos os retornos de n passos $G_t^{(n)}$:

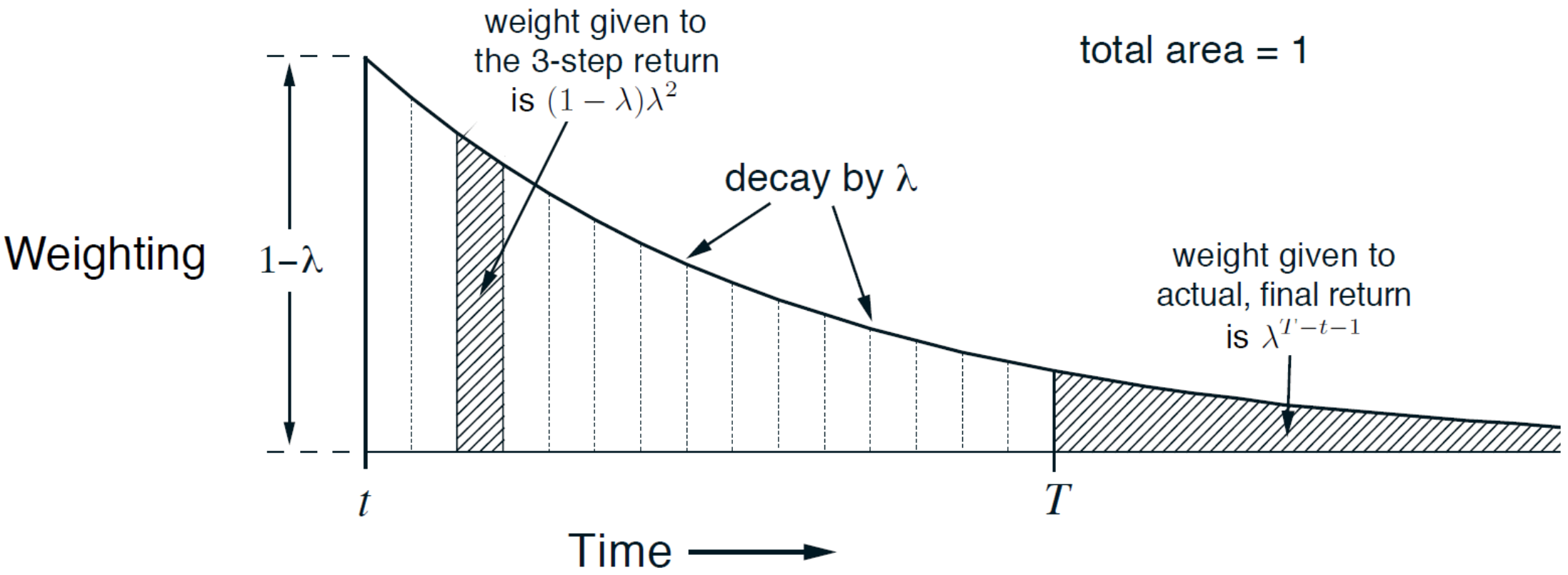
$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

- No caso episódico, fica-se com:

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t-1} G_t$$

- Chama-se esse algoritmo de $TD(\lambda)$. O nome TD(0) vem disso.
- Implementação eficiente usa *eligibility traces*.

λ-Return



Eligibility Traces

- TD(λ) sofre do mesmo problema de MC de ter que esperar até o final do episódio.
- *Eligibility traces* fornecem uma implementação de modo que isso não é necessário.
- Ideia: quando algo acontecer, atribuir crédito de acordo com:
  - Frequência: o mais frequente recebe mais crédito.
  - Recência: o mais recente recebe mais crédito.
- Para RL: quando for atualizar $V(s)$ com erro TD, dar mais crédito para estados visitados mais frequentemente e recentemente.

Eligibility Traces

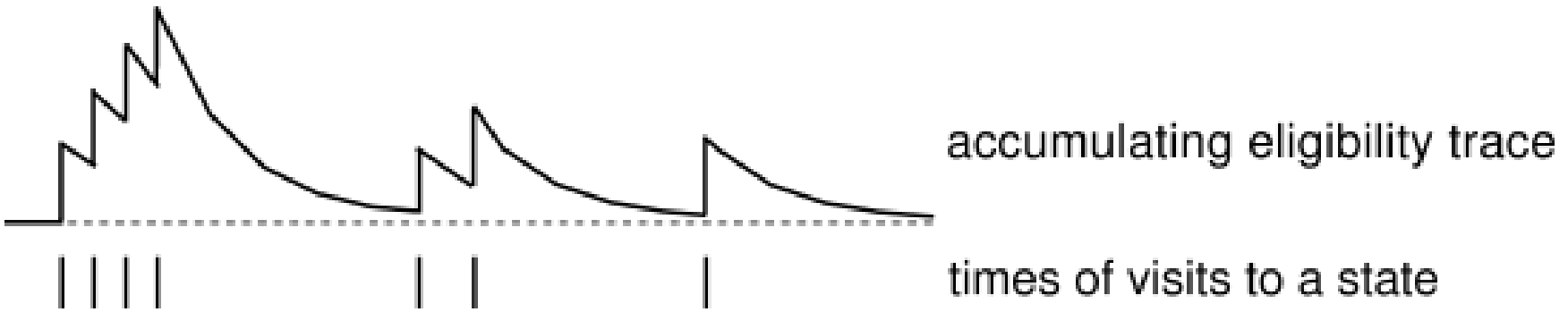
- No começo do episódio:

$$E_0(s) = 0, \forall s \in S$$

- A cada passo de tempo:

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$

- $E_t(s)$ decai ao longo do tempo, mas é reforçado se s for visitado.



Eligibility Traces

- Manter atualizado *eligibility trace* para cada estado s.
- A cada passo, atualizar valor de $V(s)$ de todos os estados de acordo com erro TD δ_t e eligibility trace $E_t(s)$:

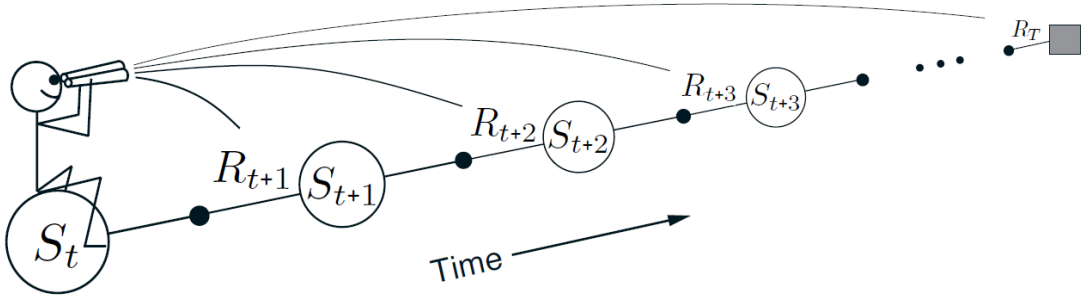
$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$
$$V(s) = V(s) + \alpha \delta_t E_t(s)$$

Forward e Backward View TD(λ)

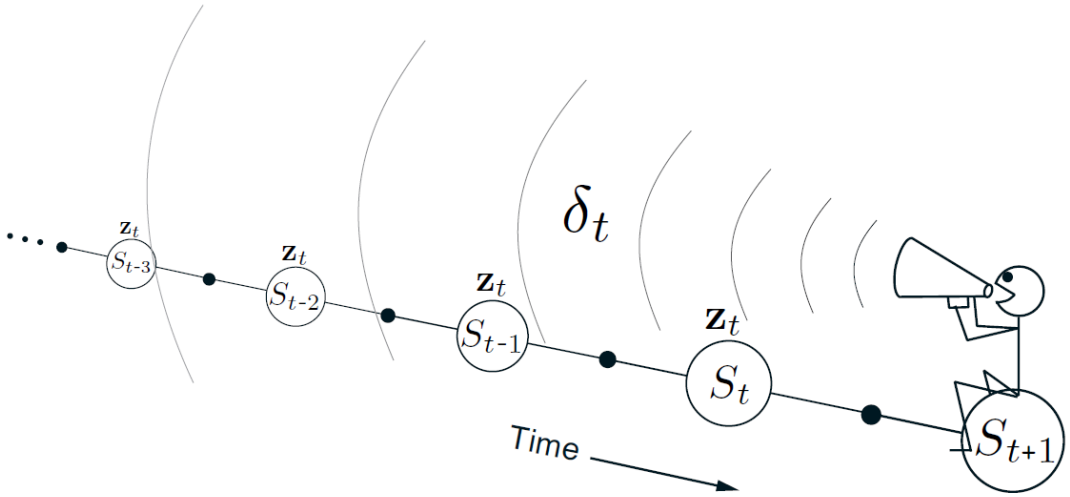
- *Forward View* TD(λ) é a versão inicial, sem *eligibility traces*.
- *Backward View* TD(λ) é a versão com *eligibility traces*.
- Pode-se mostrar que no final do episódio, ambas as versões *Forward* e *Backward* terão atualizado $V(s)$ exatamente da mesma forma.
- Assim, *Backward View* TD(λ) apenas fornece um mecanismo de implementação conveniente, pois não há mais necessidade de esperar até o final do espisódio para atualizar $V(s)$.

Forward e Backward View TD(λ)

***Forward View* TD(λ)**



***Backward View* TD(λ)**



Controle de MC

Aprendizado *On* e *Off-Policy*

- Aprendizado *on-policy*:
  - Aprender enquanto faz.
  - Aprende a política π enquanto executa a política π.
  - Aprende a política π de experiências amostradas através da política π.
- Aprendizado *off-policy*:
  - Aprender vendo outro fazer.
  - Aprender a política π enquanto executa a política μ.
  - Aprende a política π de experiências amostradas através da política μ.

Iteração de Política (Relembrando)

- Iniciar com política e função valor arbitrariamente.
- Loop:
  - Avaliar a política usando avaliação de política iterativa.

$$\boldsymbol{V}_{k+1} = \mathbf{r}_s^{\pi} + \gamma \mathbf{P}_{ss'}^{\pi} \boldsymbol{V}_k$$

  - Melhorar a política agindo de forma gulosa em relação a $V_k(s)$:

$$\pi'(s) = greedy(V_k(s))$$

Observação: durante a avaliação, não precisa iterar até convergir.

Esse algoritmo converge para a política ótima.

Iteração de Política com Avaliação de MC

- Ao invés de usar equação de expectativa de Bellman para avaliar política, pode-se usar MC!
- Se usamos $V(s)$, aprimoramento guloso de política requer modelo do MDP:

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, r(s,a) + \sum_{s'} p(s'|s,a) V(s')$$

- Solução: usar função ação-valor $Q(s,a)$:

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a)$$

Iteração de Política com Avaliação de MC

- Relembrando função ação-valor:

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$$

Iteração de Política com Avaliação de MC

- Iniciar com política e função ação-valor arbitrariamente.
- Loop:
  - Avaliar a política usando Monte Carlo:

$$Q(S_t, A_t) = S(S_t, A_t)/N(S_t, A_t)$$

  - Melhorar a política agindo de forma gulosa em relação a $Q(s, a)$:

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, Q(s, a)$$

- Problema: MC só converge para $q_\pi(s, a)$ se visitar todos os pares estado-ação infinitas vezes. π tem que garantir exploração.

Política ε-*greedy*

- Garante exploração contínua.
- Simples implementação.
- Todas as ações possíveis tem probabilidade não-nula.
- Escolher ação gulosa com probabilidade $1 - \varepsilon$.
- Escolher ação aleatoriamente com probabilidade ε.

$$\pi(a|s) = \begin{cases} \frac{\varepsilon}{m} + 1 - \varepsilon, a = \underset{a' \in A}{\operatorname{argmax}}\, Q(s, a') \\ \frac{\varepsilon}{m}, caso\ contrário \end{cases}$$

Controle de MC

- Iniciar com política e função ação-valor iniciais.
- Loop:
  - Avaliar a política usando Monte Carlo:

$$Q(S_t, A_t) = S(S_t, A_t)/N(S_t, A_t)$$

  - Aprimorar política ε-*greedy*.

$$\pi'(a|s) = \varepsilon\text{-}greedy(Q(s, a))$$

- **Observação:** se ε for constante, nunca converge para política ótima de verdade.

GLIE

- *Greedy in the Limit with Infinite Exploration* (GLIE).
- Ideia: reduzir ε ao longo do tempo. Exemplo: $\varepsilon_k = \frac{1}{k}$.
- Com isso:
  - Todos os pares estado-ação são visitados infinitas vezes:

$$\lim_{k\to\infty} N_k(s,a) = \infty$$

  - A política converge para uma política gulosa:

$$\lim_{k\to\infty} \pi_k(a|s) = \begin{cases} 1, a = \underset{a\prime\in A}{\operatorname{argmax}}\, Q_k(s,a') \\ 0, caso\ contrário \end{cases}$$

Controle de MC com GLIE

- Iniciar com política e função ação-valor arbitrariamente.
- Loop:
  - Amostrar k-ésimo episódio seguindo π: $\tau_k = S_0, A_0, R_1, \dots, S_T$.
  - Para cada estado S_t e ação A_t no episódio:

$$N(S_t, A_t) = N(S_t, A_t) + 1$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\big(G_t - Q(S_t, A_t)\big)$$

  - Aprimorar política ε-*greedy*.

$$\varepsilon = 1/k$$

$$\pi'(a|s) = \varepsilon\text{–}greedy(Q(s, a))$$

- Converge para $q_*(s, a)$ e $\pi_*(a|s)$. *On-policy*.

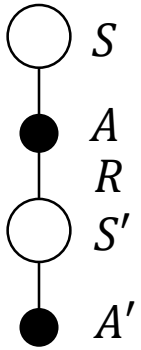
Sarsa

MC x TD para Controle

- Em Predição, vimos que TD tem várias vantagens sobre MC:
  - Menor variância.
  - *On-line* (não precisa esperar terminar o episódio).
  - Pode usar trajetórias incompletas.
  - Funciona para tarefas continuadas.
- Usar TD ao invés de MC para avaliar política:
  - Aplicar TD para aprender $Q(s, a)$.
  - Usar aprimoramento ε-*greedy*.
  - Com isso, pode-se atualizar a cada passo, ao invés de precisar usar o episódio inteiro.

Sarsa

$$Q(S,A) = Q(S,A) + \alpha\big(R + \gamma Q(S',A') - Q(S,A)\big)$$



- Nome vem de S, A, R, S', A'.
- A' escolhido com mesma política π.
- *On-policy* pois avalia política sendo executada por amostragem.

Sarsa

Inicializar $Q(s, a)$ arbitrariamente. Inicializar $\pi = \varepsilon$-*greedy*(Q).

Loop (para cada episódio):

Inicializar S

$A \sim \pi(a|S) = \varepsilon$-*greedy*$(Q)$

Loop (para cada passo do episódio):

Tomar ação A, observar R, S'

$A' \sim \pi(a|S') = \varepsilon$–greedy$(Q)$

$Q(S, A) = Q(S, A) + \alpha\big(R + \gamma Q(S', A') - Q(S, A)\big)$

$S = S'; A = A'$

Até o fim do episódio

Convergência do Sarsa

- Teorema: Sarsa converge para a função ação-valor ótima, $Q(s,a) \rightarrow q_*(s,a)$ sobre as seguintes condições:
  - Sequência de políticas $\pi_k(a|s)$ GLIE.
  - Sequência de passos de aprendizado Robbins-Monro:

$$\sum_{t=1}^{\infty} \alpha_t = \infty\,, \qquad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

- Na prática, Sarsa costuma funcionar bem mesmo quando não nos preocupamos com essas condições.
- É comum usar α fixo com Sarsa.

Sarsa de n Passos

Assim como foi feito com TD, pode-se pensar em “Q-retornos” de n passos para Sarsa:

- $n = 1$ (Sarsa): $Q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$.
- $n = 2$: $Q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2})$.
- n: $Q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n Q(S_{t+n}, A_{t+n})$.
- $n = \infty$ (MC): $Q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$.

Sarsa de n Passos

- Q-retorno de n passos:

$$Q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n Q(S_{t+n}, A_{t+n})$$

- Sarsa de n passos atualiza $Q(S_t, A_t)$ usando $Q_t^{(n)}$ como alvo:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left(Q_t^{(n)} - Q(S_t, A_t) \right)$$

- Também existe um Sarsa(λ).

Q-Learning

Q-Learning

- Aprendizado *off-policy*.
- Segue política de comportamento μ.
- Aprende política alvo ótima π.
- μ é ε-*greedy*.
- π é *greedy*.
- Atualização de $Q(S_t, A_t)$:
  $$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_{a' \in A} Q(S_{t+1}, a') - Q(S_t, A_t) \right),$$
- No fundo, usa equação de otimalidade de Bellman com amostragem.

Q-Learning

- Durante aprendizado, μ e π são aprimoradas.
- Como *Q-Learning* aprende política ótima diretamente, não há necessidade de reduzir ε ao longo do aprendizado.

Q-Learning

Inicializar $Q(s, a)$ arbitrariamente. Inicializar $\mu = \varepsilon$-*greedy*(Q).

Loop (para cada episódio):

Inicializar S

$A \sim \mu(a|S) = \varepsilon$-*greedy*$(Q)$

Loop (para cada passo do episódio):

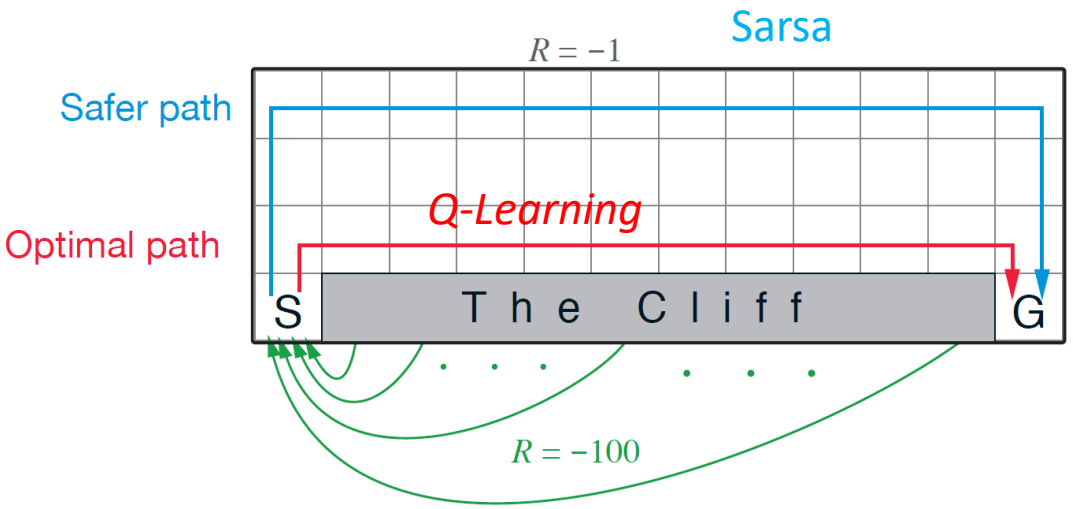
Tomar ação A, observar R, S'

$$Q(S, A) = Q(S, A) + \alpha \left(R + \gamma \max_{a' \in A} Q(S', a') - Q(S, A) \right)$$

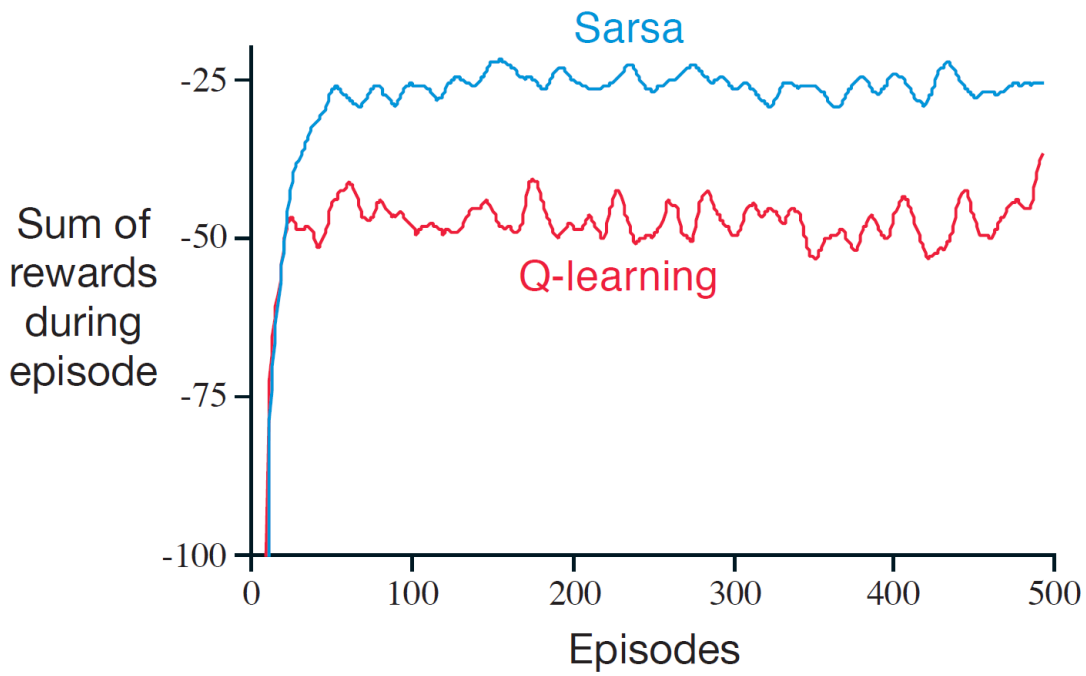
$$S = S'$$

Até o fim do episódio

Sarsa x *Q-Learning*



$\varepsilon = 0{,}1$



Sarsa x *Q-Learning*

- *Q-Learning* encontra o caminho ótimo, enquanto o Sarsa fica limitado pelo *ε-greedy*.
- Apesar disso, o Sarsa tem melhor desempenho em execução.
- Isso acontece pois Sarsa leva em conta que executa *ε-greedy*, assim aprende política mais conservadora.

DP x TD

	Programação Dinâmica	*Temporal-Difference*
Equação de expectativa de Bellman para $v_\pi(s)$	Avaliação de Política	Aprendizado TD (Predição)
Equação de expectativa de Bellman para $q_\pi(s,a)$	Iteração de Política	Sarsa
Equação de otimalidade de Bellman para $q_*(s,a)$	Iteração de Valor	Q-Learning

Equações de Atualização

	Programação Dinâmica	Temporal-Difference
Equação de expectativa de Bellman para $v_\pi(s)$	$V(s) = E[R + \gamma V(s')\|s]$	$V(s) \to R + \gamma V(s')$
Equação de expectativa de Bellman para $q_\pi(s,a)$	$Q(s,a) = E[R + \gamma Q(s',a')\|s,a]$	$Q(S,A) \to R + \gamma Q(S',A')$
Equação de otimalidade de Bellman para $q_*(s,a)$	$Q(s,a) = E[R + \gamma \max_{a' \in A} Q(s',a')\|s,a]$	$Q(S,A) \to R + \gamma \max_{a' \in A} Q(S',a')$

Amostragem por Importância

Aprendizado *Off-Policy*

- Avaliar política $\pi(a|s)$ para calcular $v_\pi(s)$ ou $q_\pi(s,a)$, enquanto segue política $\mu(a|s)$:

$$\tau = S_0, A_0, R_1, S_1, A_1, \ldots, S_T \sim \mu$$

- Utilidades:
  - Aprender observando humanos e outros agentes.
  - Reusar experiência gerada por políticas antigas $\pi_1, \pi_2, \ldots, \pi_{t-1}$ (com *on-policy*, as experiências anteriores são descartadas).
  - Aprender política ótima enquanto segue política exploratória.

Amostragem por Importância

- Inglês: *Importance Sampling* (IS).
- Estimar esperança usando uma distribuição diferente:

$$E_{X\sim P}[f(X)] = \sum_X P(X)f(X) = \sum_X Q(X)\frac{P(X)}{Q(X)}f(X) \Rightarrow$$

$$E_{X\sim P}[f(X)] = E_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]$$

- Usa-se geralmente no caso em que queremos amostrar de p, mas amostrar de P é difícil e amostrar de Q é fácil.
- No caso de RL, $P = \pi$ e $Q = \mu$.

IS para MC *Off-line*

- Retornos gerados seguindo μ.
- Pode-se mostrar que o retorno corrigido para π é dado por:

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \frac{\pi(A_{t+2}|S_{t+2})}{\mu(A_{t+2}|S_{t+2})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

- Atualização da função valor:

$$V_\pi(S_t) = V_\pi(S_t) + \alpha(G_t^{\pi/\mu} - V_\pi(S_t))$$

- IS pode aumentar muito a variância.
- Na prática, MC *off-line* com IS não funciona devido à variância muito alta.

IS para TD *Off-Policy*

- Usar alvo TD de μ para avaliar π.
- Considerar IS para corrigir o alvo TD.
- Equação de atualização:

$$V_\pi(S_t) = V_\pi(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\left(R_{t+1} + \gamma V(S_{t+1})\right) - V_\pi(S_t)\right)$$

- Variância muito menor que MC *off-line* com IS.

Para Saber Mais

- Curso do David Silver (aulas 4 e 5): http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.
- Capítulos 5, 6 e 7 do livro: SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction, second edition*. The MIT Press, 2017.

Laboratório 12

Laboratório 12

- Usar Sarsa e/ou Q-Learning para aprender política de robô seguidor de linha.