



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA



Intelligenza Artificiale

Model Free Control

Outline

- ▶ Introduzione
- ▶ On-policy Vs Off-Policy
- ▶ On-policy Monte-Carlo
- ▶ On-policy TD learning (SARSA)
- ▶ Off-policy TD (Q-learning)

Model Free Prediction/Control

- ▶ **Model-free prediction**
 - ▶ Stimare la value function di un MDP non noto
- ▶ **Model-free control**
 - ▶ Ottimizzare la value function di un MDP non noto

Model Free Control

- ▶ Robot walking
 - ▶ Robocup Soccer
 - ▶ Portfolio Management
 - ▶ Go
 - ▶ Protein Folding
 - ▶ Bioreactor
-
- ▶ Per la maggior parte di questi problemi, o:
 - ▶ Il modello del MDP **non è noto**, ma l'esperienza può essere campionata
 - ▶ Il modello del MDP **è noto**, ma è **troppo grande** per essere utilizzato, se non per campionamento
 - ▶ Il Model Free Control può risolvere questi problemi

On-policy & Off-policy Learning

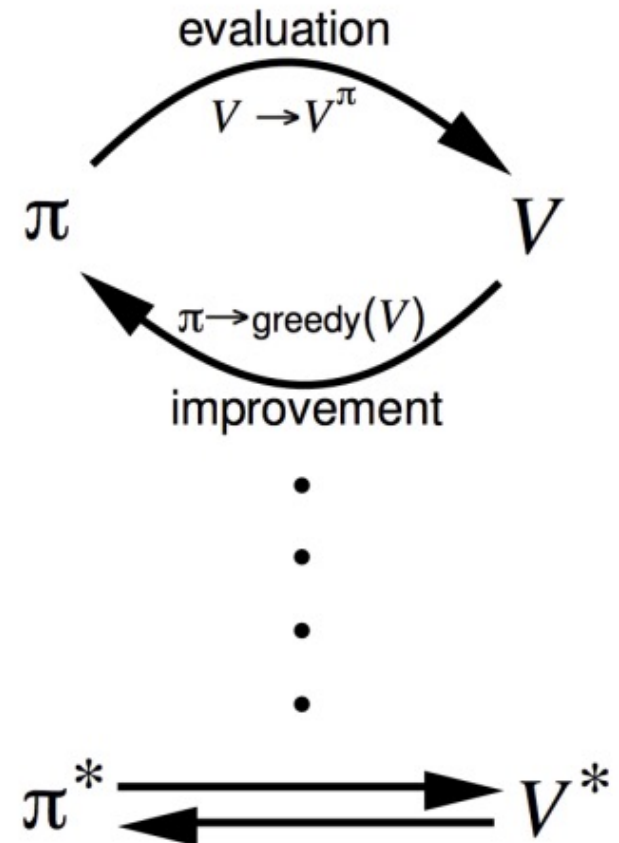
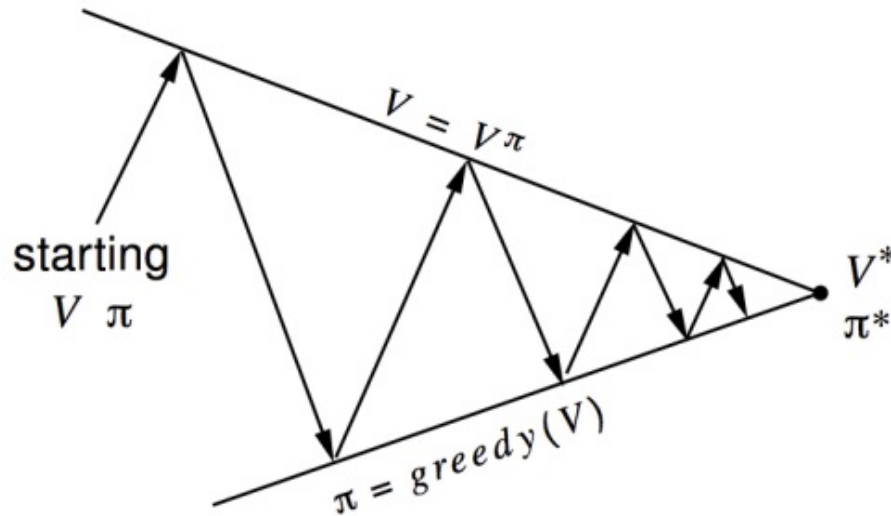
- ▶ **On-policy** learning

- ▶ Apprendere sul campo
- ▶ Apprendere la policy π dall'esperienza campionata da π

- ▶ **Off-policy** learning

- ▶ Osservare qualcuno
- ▶ Apprendere la policy π dall'esperienza campionata da μ

Policy Iteration Generalizzata



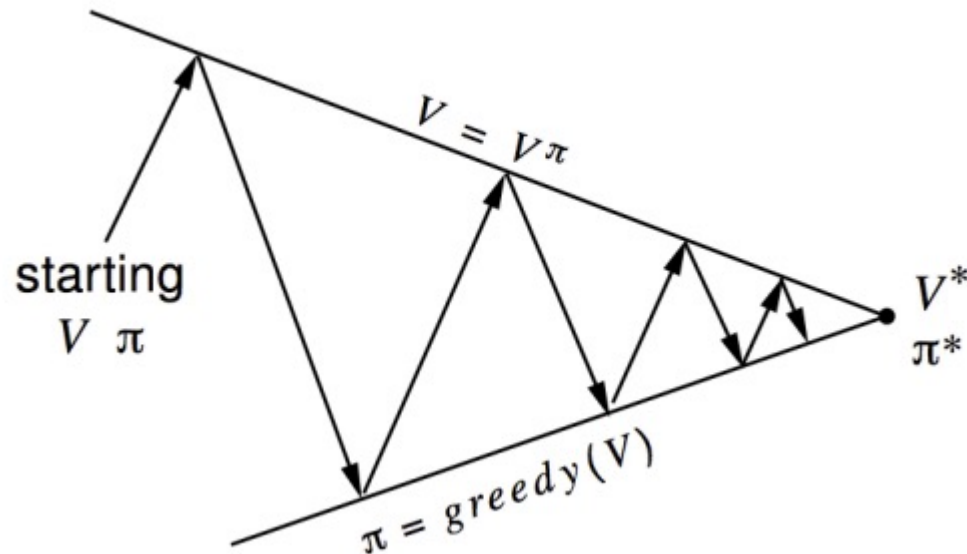
Valutazione della Policy Stima di v_π

Qualsiasi algoritmo di policy evaluation

Miglioramento della Policy Generazione di $\pi' \geq \pi$

Qualsiasi algoritmo di policy improvement

Policy Iteration Generalizzata con On-policy MC



- ▶ **Valutazione della Policy** – Monte-Carlo policy evaluation, $V = v_\pi$?
- ▶ **Miglioramento della Policy** – Genera un miglioramento greedy della policy?

Model-Free Policy Iteration tramite la Action-Value Function

- ▶ Il miglioramento greedy della policy **rispetto a $V(s)$** **necessita** del modello di un MDP

$$\pi'(s) = \arg \max_{a \in \mathcal{A}} \mathcal{R}_s^a + P_{ss'}^a V(s')$$

- ▶ Il miglioramento greedy della policy **rispetto a $Q(s,a)$** è **model-free**

$$\pi'(s) = \arg \max_{a \in \mathcal{A}} Q(s, a)$$

Monte Carlo Exploring Starts

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0 , following π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

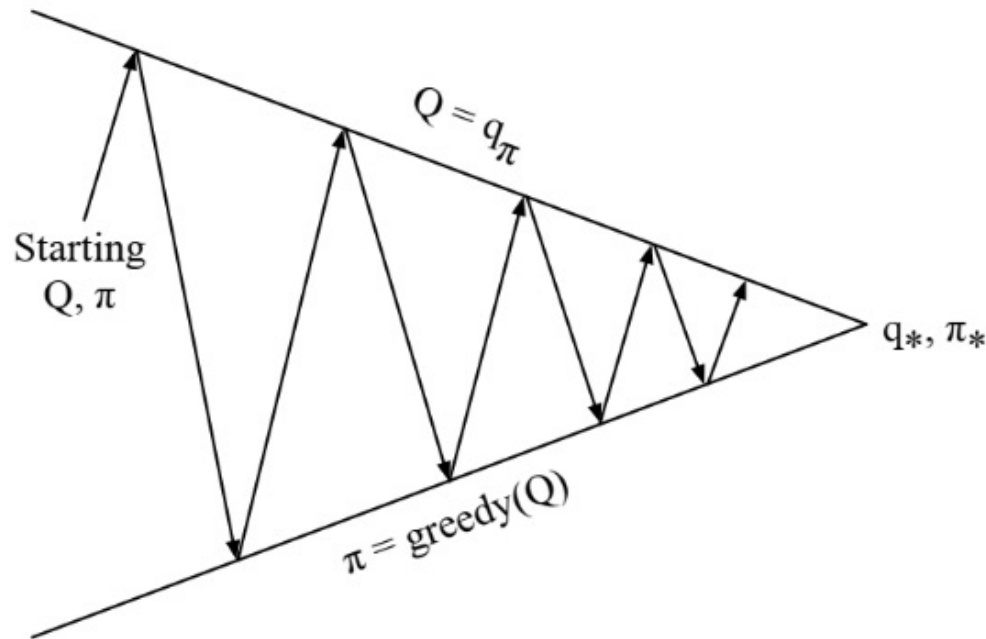
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$

Policy Iteration Generalizzata con Action-Value Function



- **Valutazione della Policy** – Monte-Carlo policy evaluation, $Q = q_\pi$?
- **Miglioramento della Policy** – Genera un miglioramento greedy della policy?

Esempio di selezione di azioni greedy



- ▶ Ci sono due porte
- ▶ Apri la porta di sinistra e ottieni ricompensa 0 – $V(\text{sinistra}) = 0$
- ▶ Apri la porta di destra e ottieni ricompensa +1 – $V(\text{destra}) = +1$
- ▶ Apri la porta di destra e ottieni ricompensa +3 – $V(\text{destra}) = +2$
- ▶ Apri la porta di destra e ottieni ricompensa +2 – $V(\text{destra}) = +2$

...

Sei sicuro di aver scelto la porta migliore?

ϵ -greedy Exploration

- ▶ L'idea più semplice per garantire un'esplorazione continua
- ▶ Tutte le m azioni vengono sperimentate con **una probabilità non nulla**
- ▶ Con probabilità $1-\epsilon$ viene selezionata l'azione greedy
- ▶ Con probabilità ϵ viene selezionata un'azione random

$$\pi(a|s) = \begin{cases} \epsilon/m + (1 - \epsilon) & \text{if } a^* = \arg \max_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

ϵ -greedy Exploration

On-policy first-visit MC control (for ϵ -soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\epsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ϵ -soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

ϵ -greedy Policy improvement

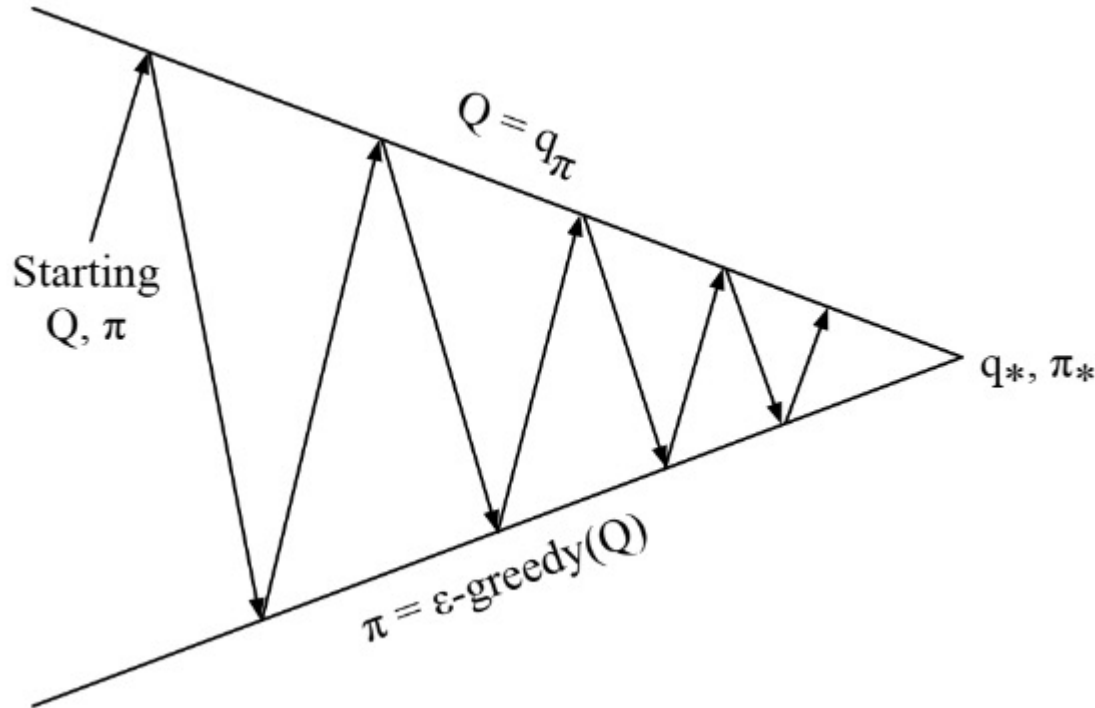
► Teorema:

- Per ogni policy ϵ -greedy π , la policy ϵ -greedy π' rispetto a q_π è un miglioramento, $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\ &= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\ &\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s) \end{aligned}$$

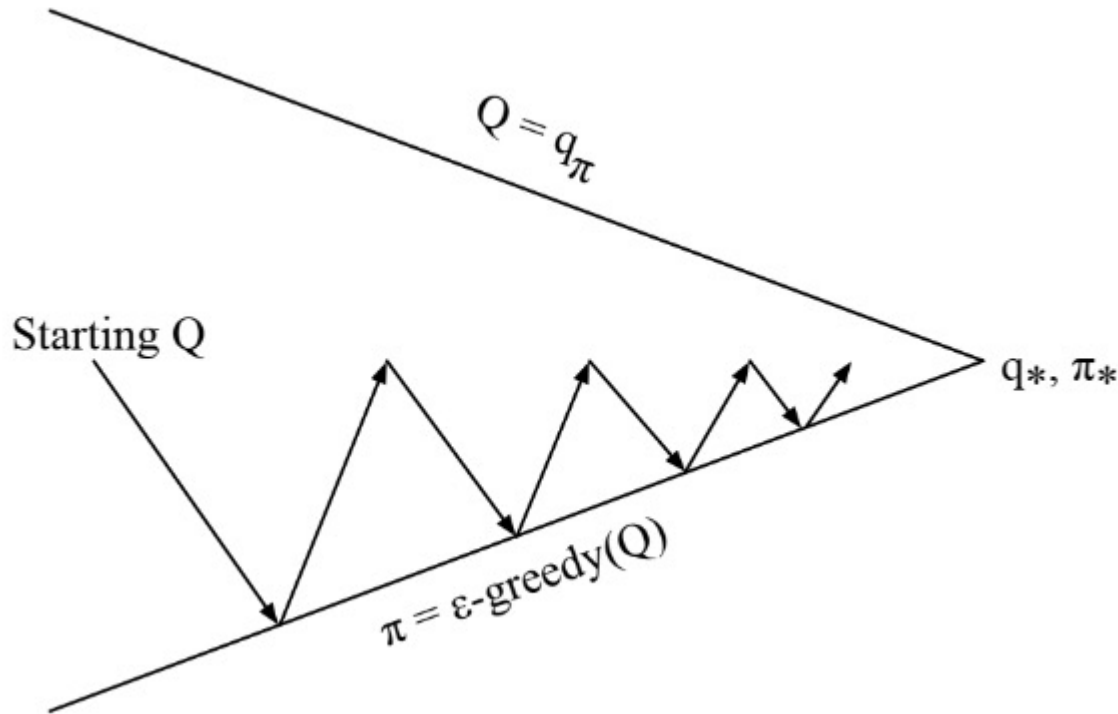
- Pertanto, dal **policy improvement theorem** $v_{\pi'}(s) \geq v_\pi(s)$

Monte-Carlo Policy Iteration



- ▶ Valutazione della Policy – Monte-Carlo policy evaluation, $Q = q_\pi$
- ▶ Miglioramento della Policy – ϵ -greedy policy improvement

Monte-Carlo Control



Ogni episodio

- ▶ Valutazione della Policy – Monte-Carlo policy evaluation, $Q \approx q_\pi$
- ▶ Miglioramento della Policy – $\epsilon\text{-greedy}$ policy improvement

Greedy in the Limit with Infinite Exploration (GLIE)

- ▶ Definizione (GLIE)

- ▶ Tutte le coppie stato-azione vengono esplorate **un numero infinito** di volte

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

- ▶ La policy converge ad una policy greedy

$$\lim_{k \rightarrow \infty} \pi_k(a|s) = \mathbf{1}\left(a; \arg \max_{a' \in \mathcal{A}} Q_k(s, a')\right)$$

- ▶ ϵ -greedy è GLIE se ϵ si azzera a $\epsilon_k = \frac{1}{k}$

GLIE Monte-Carlo Control

- ▶ Campionare il k -esimo episodio utilizzando

$$\pi: \{S_1, A_1, R_2, \dots, S_T\} \sim \pi$$

- ▶ Per ogni stato S_t e azione A_t nell'episodio:

$$\begin{aligned} N(S_t, A_t) &\leftarrow N(S_t, A_t) + 1 \\ Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t)) \end{aligned}$$

- ▶ Migliorare la policy in base alla nuova action-value function

$$\begin{aligned} \epsilon &\leftarrow \frac{1}{k} \\ \pi &\leftarrow \epsilon - \text{greedy}(Q) \end{aligned}$$

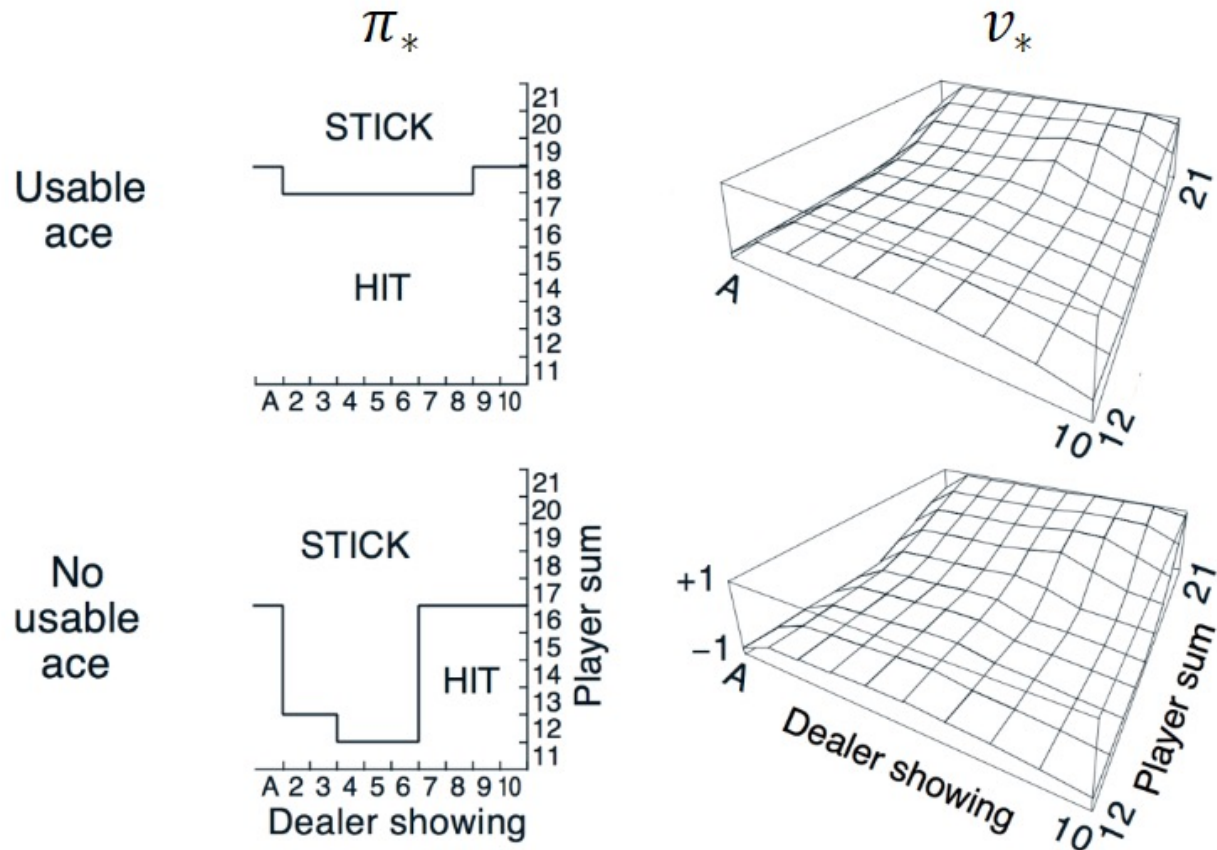
- ▶ Teorema:

- ▶ GLIE Monte-Carlo Control converge verso la action-value function ottimale

Esempio Blackjack



Esempio Blackjack – MC Control



On-Policy TD Control

MC Vs TD Control

- ▶ TD learning presenta diversi vantaggi rispetto a MC
 - ▶ Varianza inferiore
 - ▶ Online
 - ▶ Sequenze incomplete
- ▶ Intuizione - Utilizzare TD piuttosto che MC nel nostro control loop
 - ▶ Applicare TD a $Q(s,a)$
 - ▶ Utilizzare ϵ -greedy policy improvement
 - ▶ Aggiornare ad ogni time-step

Aggiornamento delle Action-Value Function con SARSA

● (S, A)

R

○ S'

● A'

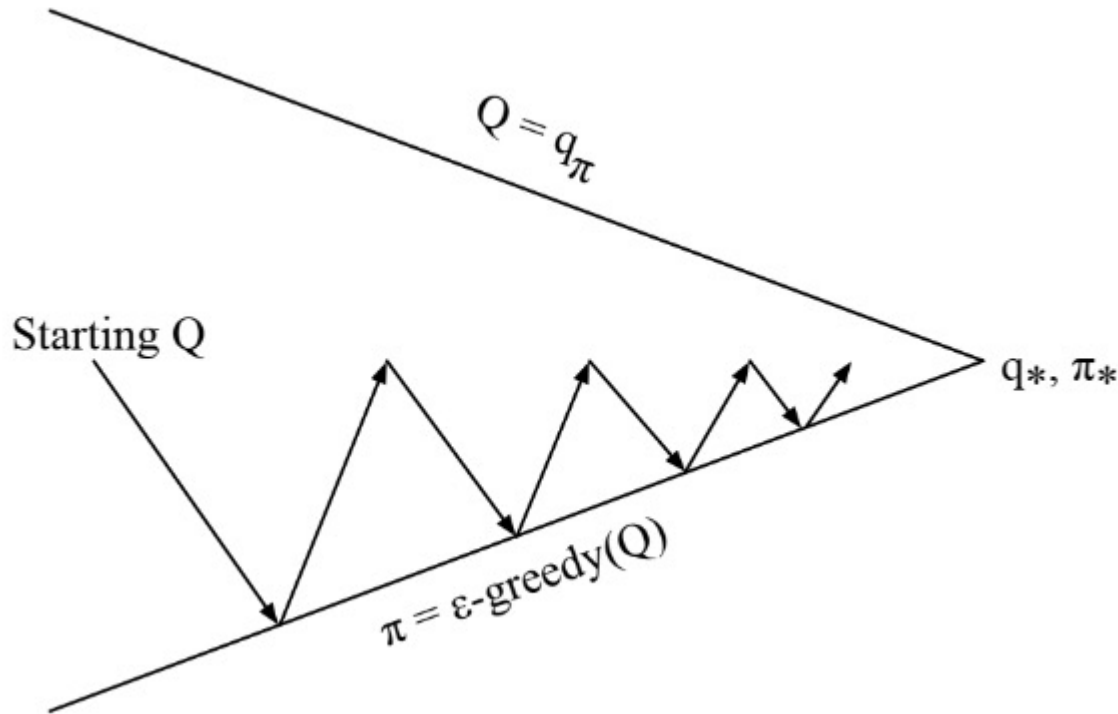
Expected SARSA

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') q_*(s', a')$$

Campioniamo anche l'azione futura A'
(invece di sfruttare la policy per
calcolare l'expectation)

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

On-Policy Control con SARSA



Ogni time-step

- ▶ Valutazione della Policy – **SARSA**, $Q \approx q_\pi$
- ▶ Miglioramento della Policy – ϵ -greedy policy improvement

Algoritmo SARSA per On-Policy Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Loop for each step of episode:

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal

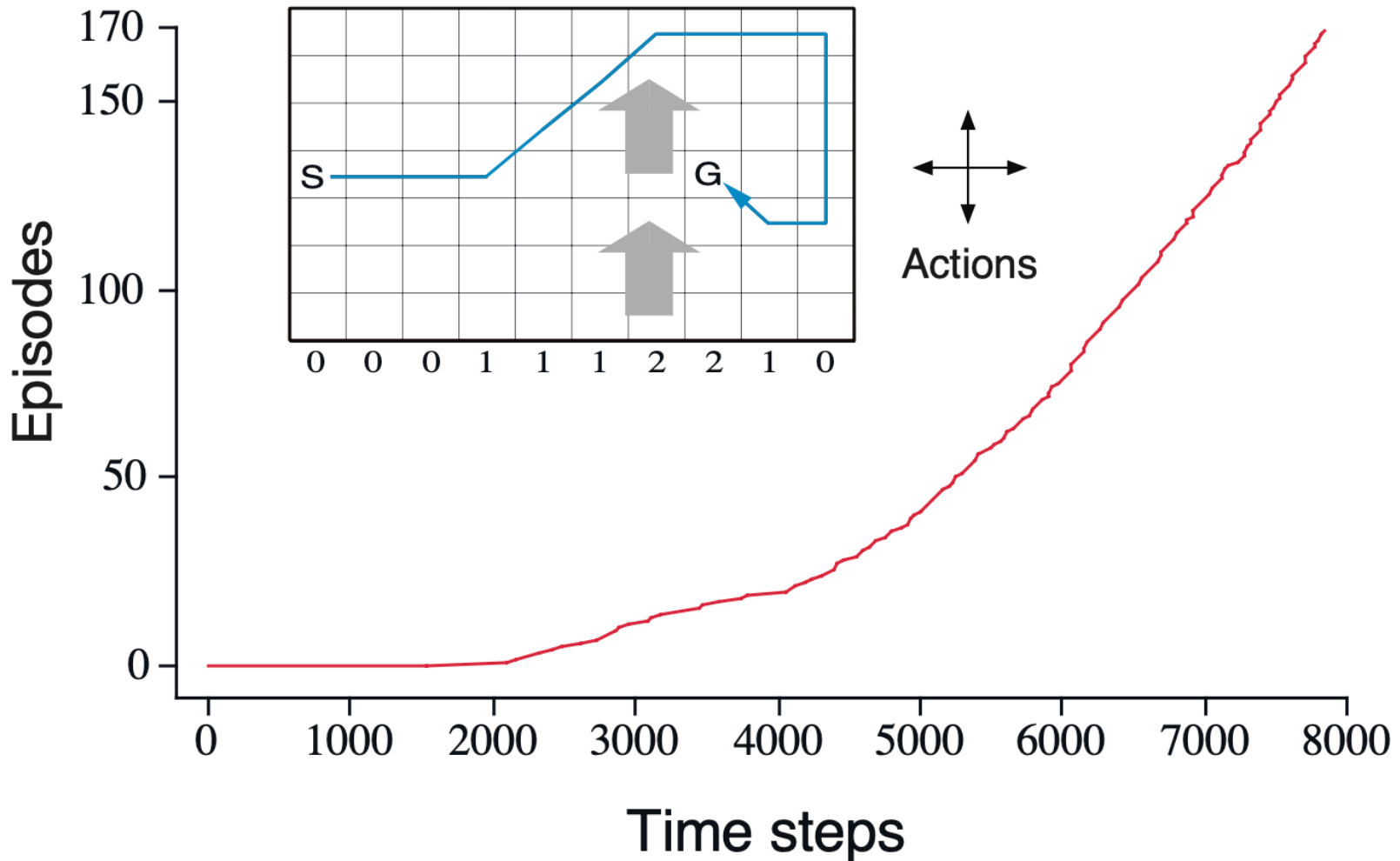
Convergenza di SARSA

- ▶ Teorema:

- ▶ L'algoritmo SARSA converge verso la action-value function ottimale ($Q(s,a) \rightarrow q_*(s,a)$) a condizione che:
 - ▶ Sequenza di policy $\pi_t(a|s)$ GLIE
 - ▶ Sequenza di Robbins-Monro di step-size α_t

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

Windy Gridworld



Temporal Difference Learning Gridworld Demo

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

SARSA(λ)

n-step SARSA

- ▶ Consideriamo i seguenti *n*-step return, per $n = 1, 2, \dots, \infty$

$$n = 1 \quad (\text{SARSA}) \quad q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

$$n = 2 \quad q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma Q(S_{t+2})$$

...

$$n = \infty \quad (\text{MC}) \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- ▶ Definiamo il ***n*-step Q-return**

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n})$$

- ▶ *n*-step SARSA aggiorna $Q(S, A)$ verso il *n*-step Q-return

$$Q(S, A) \leftarrow Q(S, A) + \alpha (q_t^{(n)} - Q(S, A))$$

SARSA backups

1-step Sarsa
aka Sarsa(0)



2-step Sarsa



3-step Sarsa



...

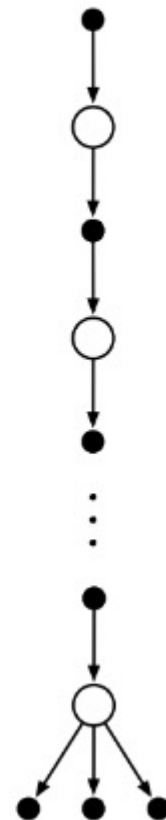
n-step Sarsa



∞ -step Sarsa
aka Monte Carlo



n-step
Expected Sarsa



Algoritmo n -step SARSA

n -step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$

Initialize π to be ε -greedy with respect to Q , or to a fixed given policy

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations (for S_t , A_t , and R_t) can take their index mod $n + 1$

Loop for each episode:

 Initialize and store $S_0 \neq \text{terminal}$

 Select and store an action $A_0 \sim \pi(\cdot | S_0)$

$T \leftarrow \infty$

 Loop for $t = 0, 1, 2, \dots$:

 If $t < T$, then:

 Take action A_t

 Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

 If S_{t+1} is terminal, then:

$T \leftarrow t + 1$

 else:

 Select and store an action $A_{t+1} \sim \pi(\cdot | S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

 If $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

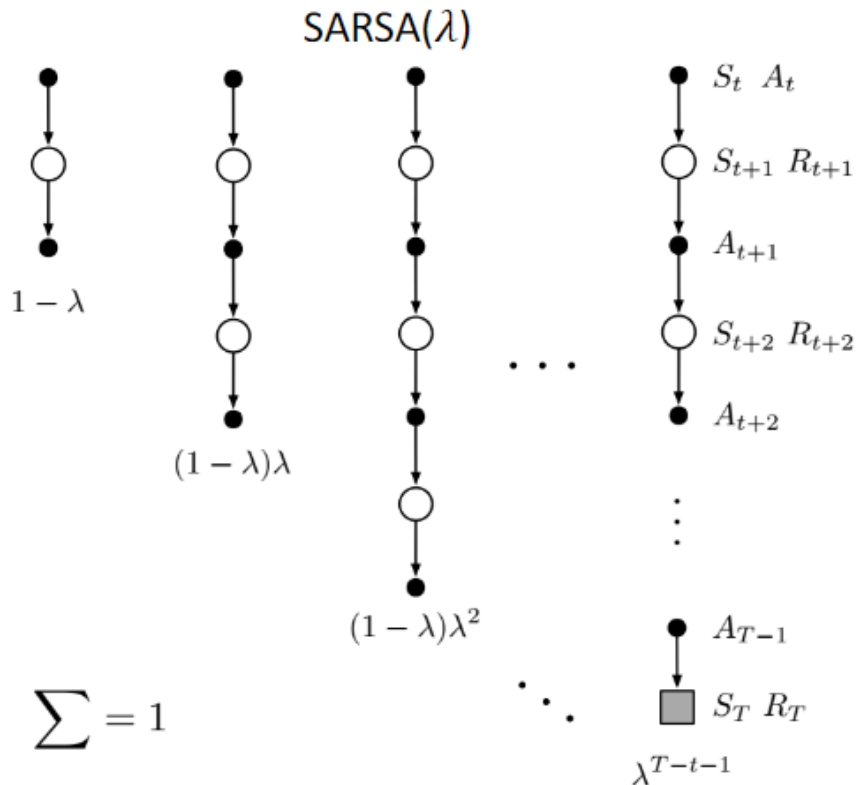
 If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ ($G_{\tau:\tau+n}$)

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

 If π is being learned, then ensure that $\pi(\cdot | S_\tau)$ is ε -greedy wrt Q

 Until $\tau = T - 1$

SARSA(λ) - Forward View



- ▶ Il q^λ return combina tutti gli n -step Q-return q_t^λ
- ▶ Utilizzando i pesi $(1 - \lambda) \lambda^{n-1}$

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

- ▶ Forward SARSA update

$$Q(S, A) \leftarrow Q(S, A) + \alpha (q_t^\lambda - Q(S, A))$$

SARSA(λ) - Backward View

- ▶ Il ritorno delle eligibility trace
- ▶ SARSA(λ) necessita di **una eligibility trace per ogni coppia stato-azione**

$$E_0(s, a) = 0$$
$$E_t(s, a) = \gamma\lambda E_{t-1}(s, a) + \mathbf{1}(S_t, A_t; s, a)$$

- ▶ $Q(s, a)$ viene aggiornato per ogni stato s e azione a in proporzione al TD-error δ_t e alla eligibility trace $E_t(s, a)$

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \delta_t E_t(s, a)$$

Algoritmo SARSA(λ)

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Repeat (for each episode):

$E(s, a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

 Initialize S, A

 Repeat (for each step of episode):

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ϵ -greedy)

$\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$

$E(S, A) \leftarrow E(S, A) + \delta$

 For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$

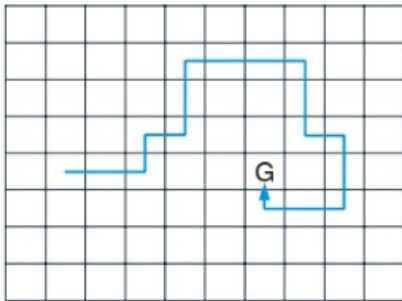
$E(s, a) \leftarrow \gamma \lambda E(s, a)$

$S \leftarrow S'; A \leftarrow A'$

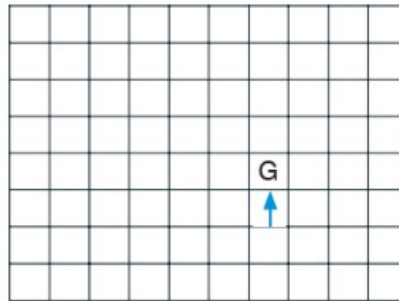
 until S is terminal

SARSA(λ) - Gridworld

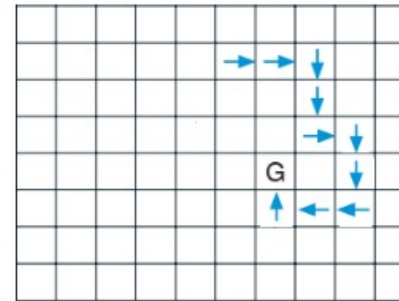
Path taken



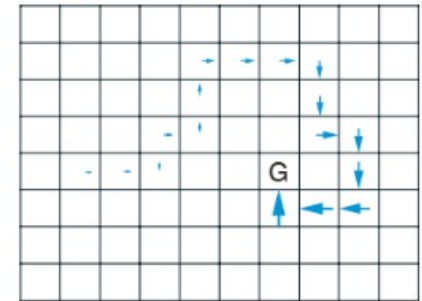
Action values increased
by one-step Sarsa



Action values increased
by 10-step Sarsa



Action values increased
by Sarsa(λ) with $\lambda=0.9$



Off-policy TD Learning

Off-Policy Learning

- ▶ Valutare la policy target $\pi(a|s)$ per calcolare $v_\pi(s)$ o $q_\pi(s,a)$
- ▶ Seguendo la policy comportamentale $\mu(a|s)$
$$\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$$
- ▶ Perché è importante?
 - ▶ Apprendere dall'imitazione (esseri umani, altri agenti,...)
 - ▶ Riutilizzare l'esperienza generata dalle vecchie policy
 - ▶ Apprendere una policy ottimale seguendo una policy esplorativa
 - ▶ Apprendere più policy seguendo una sola policy

Importance Sampling

- ▶ Stimare l'expectation facendo leva su una importance distribution esterna

$$\begin{aligned}\mathbb{E}_{X \sim P}[f(X)] &= \sum P(X) f(X) \\ &= \sum Q(X) \frac{P(X)}{Q(X)} f(X) \\ &= \mathbb{E}_{X \sim Q} \left[\frac{P(X)}{Q(X)} f(X) \right]\end{aligned}$$

Estrarre campioni dalla
importance distribution
 $Q(X)$ piuttosto che da $P(X)$

Assegnare pesi tali che
l'expectation empirica (su
campioni di $Q(X)$)
corrisponda all'expectation
sotto $P(X)$

Importance Sampling per Off-Policy Monte Carlo

- ▶ Utilizzare i guadagni generati da μ per valutare π
- ▶ Pesare il guadagno G_t in base alla somiglianza tra le policy
- ▶ Moltiplicare le correzioni dell'importance sampling lungo l'intero episodio

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

- ▶ Aggiornare il valore verso il guadagno corretto

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

- ▶ L'importance sampling può aumentare drasticamente la varianza

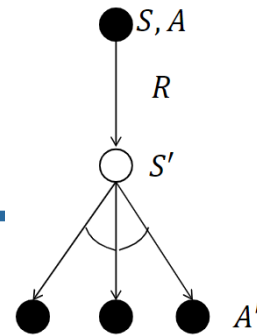
Importance Sampling per Off-Policy TD

- ▶ Utilizzare i target TD generati da μ per valutare π
- ▶ Pesare i target TD $R + \gamma V(S')$ tramite l'importance sampling
- ▶ È necessaria una singola correzione dell'importance sampling

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

- ▶ Varianza molto più bassa rispetto al MC
- ▶ Le policy devono essere simili solo per un singolo step

Q-Learning



- ▶ Off-policy learning di action-value $Q(s,a)$
- ▶ L'importance sampling **non è necessario**
- ▶ **L'azione successiva** viene scelta utilizzando la policy comportamentale $A_{t+1} \sim \mu(\cdot | St)$
- ▶ Ma prendiamo in considerazione **azioni alternative** $A' \sim \pi(\cdot | St)$
- ▶ E aggiorniamo $Q(S_t, A_t)$ in base al valore dell'azione alternativa

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Off-policy Control tramite Q-Learning

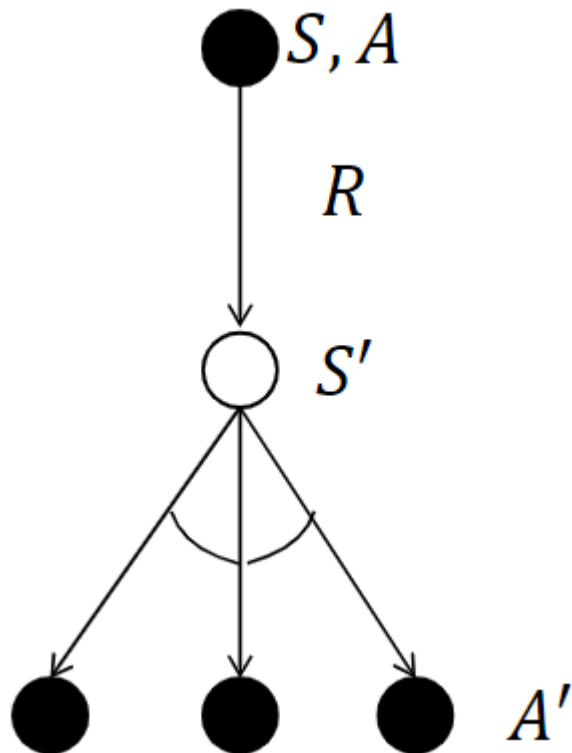
- ▶ Consente di migliorare sia il comportamento che le policy target
- ▶ La policy target π è **greedy** rispetto a $Q(S_t, A_t)$

$$\pi(S_{t+1}) = \arg \max_{a'} Q(S_{t+1}, a')$$

- ▶ La policy comportamentale μ è **ϵ -greedy** rispetto a $Q(s, a)$
- ▶ L'obiettivo del Q-learning si riduce a

$$\begin{aligned} R_{t+1} + \gamma Q(S_{t+1}, A') &= R_{t+1} + \gamma Q\left(S_{t+1}, \arg \max_{a'} Q(S_{t+1}, a')\right) \\ &= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

Algoritmo Q-Learning Control



► Teorema

- Il Q-learning control converge alla action-value ottimale, $Q(s,a) \rightarrow q_*(s,a)$

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

Algoritmo Q-Learning per Off-policy Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize S

Loop for each step of episode:

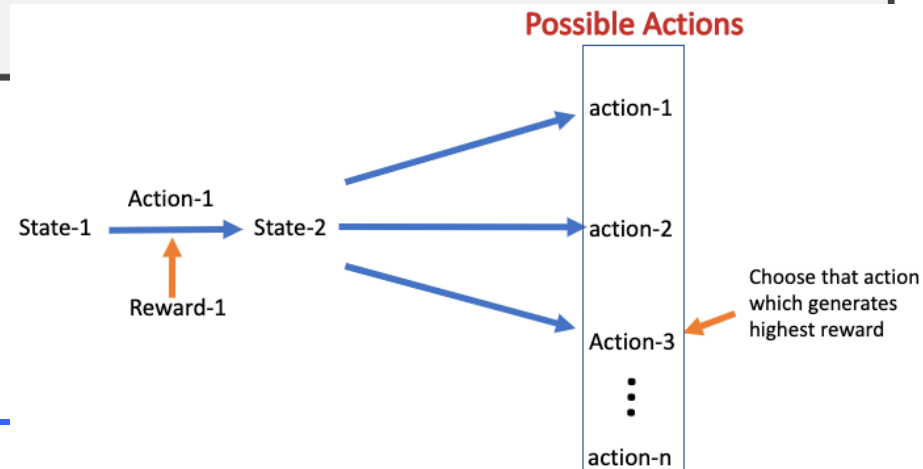
Choose A from S using policy derived from Q (e.g., ε -greedy)

Take action A , observe R , S'

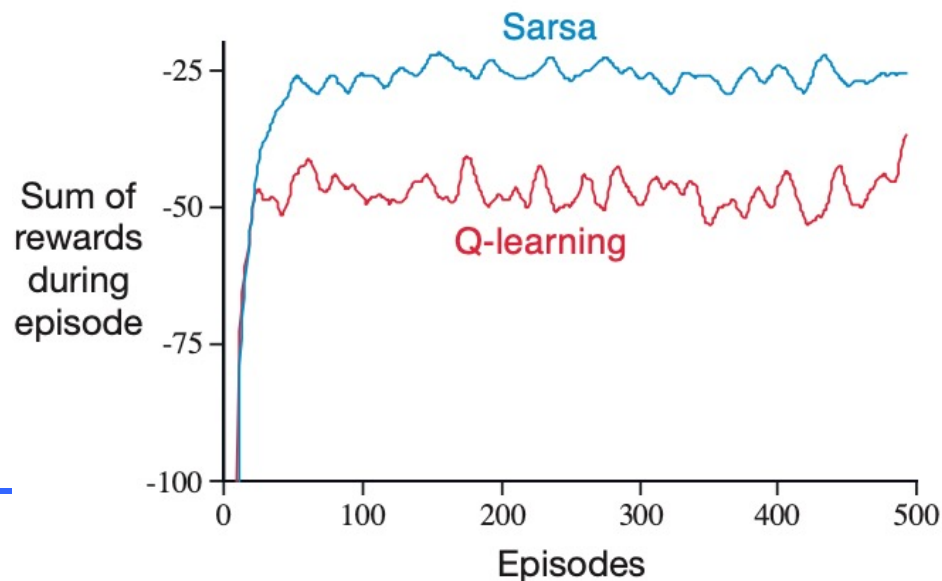
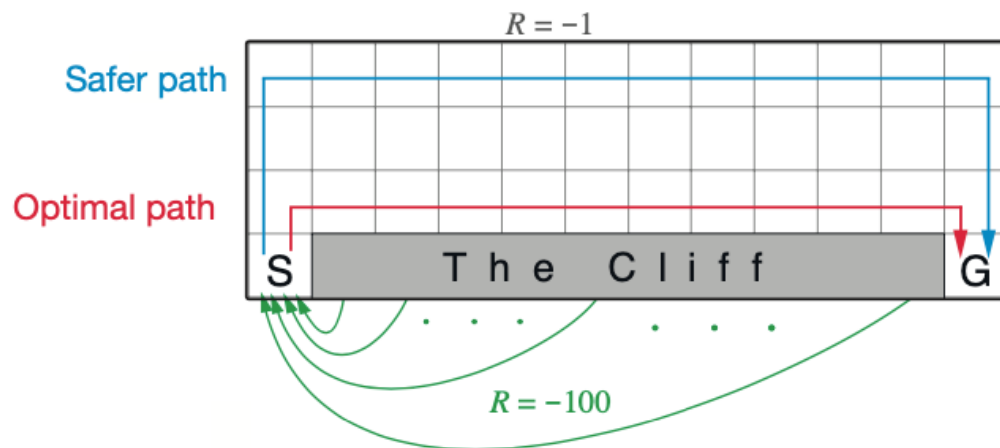
$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

until S is terminal



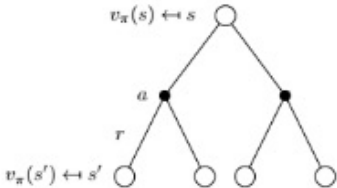

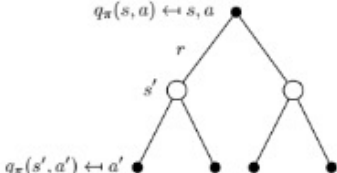
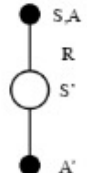
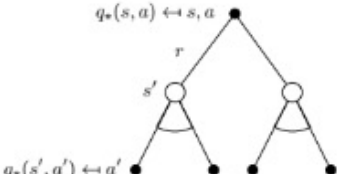
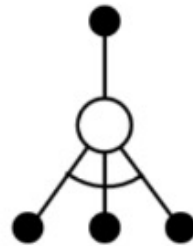
Q-learning & Exploration Demo



Q-learning & Exploration Demo

<https://www.aslanides.io/aixijs/demo.html>

Dynamic Programming Vs Temporal Difference Learning

	Full Backup (DP)	Sample Backup (TD)
Bellman Expectation Equation for $v_{\pi}(s)$	 <p>Iterative Policy Evaluation</p>	 <p>TD Learning</p>
Bellman Expectation Equation for $q_{\pi}(s, a)$	 <p>Q-Policy Iteration</p>	 <p>Sarsa</p>
Bellman Optimality Equation for $q_{*}(s, a)$	 <p>Q-Value Iteration</p>	 <p>Q-Learning</p>

Take home messages

- ▶ Model-Free control sfrutta la action-value function
 - ▶ Il miglioramento della politica non necessita di un MDP
 - ▶ Generalized policy iteration
- ▶ È necessario mantenere una esplorazione sufficiente (ϵ -greedy)
- ▶ Off-policy control
 - ▶ Apprendimento della value-function di una policy target a partire dai dati generati da una diversa policy comportamentale
 - ▶ Importance sampling per far coincidere le expectation di due policy
- ▶ TD control
 - ▶ On-policy: SARSA(λ)
 - ▶ Off-policy: Q-learning