Monte Carlo Methods

Machine Learning

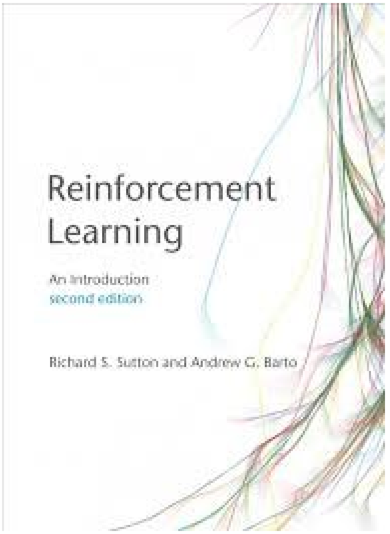
Daniele Loiacono

POLITECNICO
MILANO 1863

Outline and References

- Outline
  - Policy Evaluation
  - Policy Iteration
  - ϵ-Soft Policy Iteration
  - Off-Policy Learning

- References
  - Reinforcement Learning: An Introduction [RL Chapter 5]
  - Sample-based Learning Methods (Coursera)



Why do we need sample-based methods?

- With Dynamic Programming we are able to find the optimal value function and the corresponding optimal policy
- For example, we can use Value Iteration:

$$V_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_k(s') \right], \forall s \in \mathcal{S}$$

- Which is the major limitation of this approach?
- We generally do not know the problem dyanmics!
- We want methods to learn the optimal policy directly from data!

Overview of Monte Carlo Methods

- Monte Carlo methods relies only on the experience (**data**) to learn value functions and policy
- Monte Carlo methods can be used in two ways:
  - **model-free:** no model necessary and still attains optimality
  - **simulated:** needs only a **simulation**, not a **full** model
- Monte Carlo methods learn from **complete** sample returns
- Only defined for episodic tasks

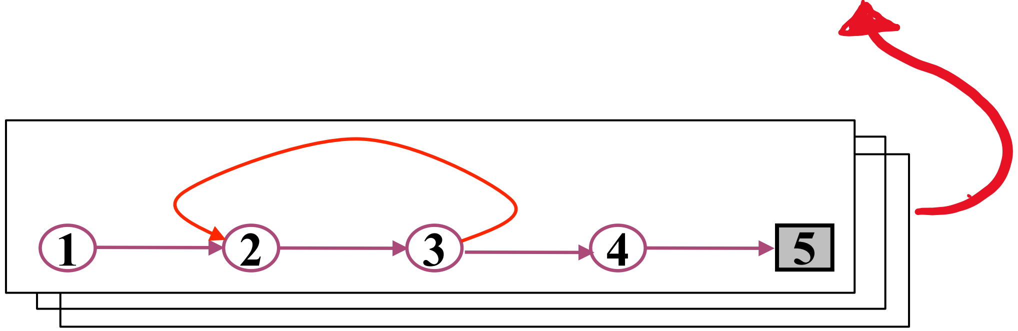
Policy Evaluation

Monte Carlo Policy Evaluation

- Goal: learn $V_\pi(s)$
- Given: some number of episodes under π which contain s
- Idea: Average returns observed after visits to s

$$V_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] \qquad \Rightarrow \qquad V_\pi(s) \approx average[G_t|S_t = s]$$

$$\boxed{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s}$$



- Every-Visit MC: average returns for **every** time s is visited in an episode
- First-visit MC: average returns only for **first** time s is visited in an episode
- Both converge asymptotically

First-visit Monte Carlo policy evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

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

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

Loop forever (for each episode):

  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

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

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

    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

      Append G to $Returns(S_t)$

      $V(S_t) \leftarrow \text{average}(Returns(S_t))$

First-visit Monte Carlo policy evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

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

$Returns(s) \leftarrow$ an empty list, for all $s \in$

Loop forever (for each episode):

Generate an episode following π: S_0, A_0

$G \leftarrow 0$

Loop for each step of episode, $t = T-1,$

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

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Incremental Updates

$$N(s_t) \leftarrow N(s_t) + 1$$

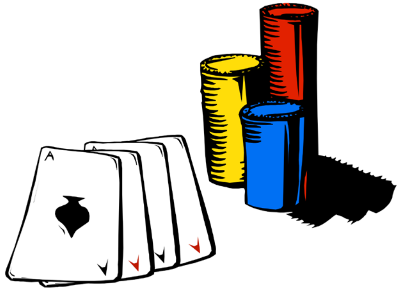
$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(G - V(s_t))$$

or

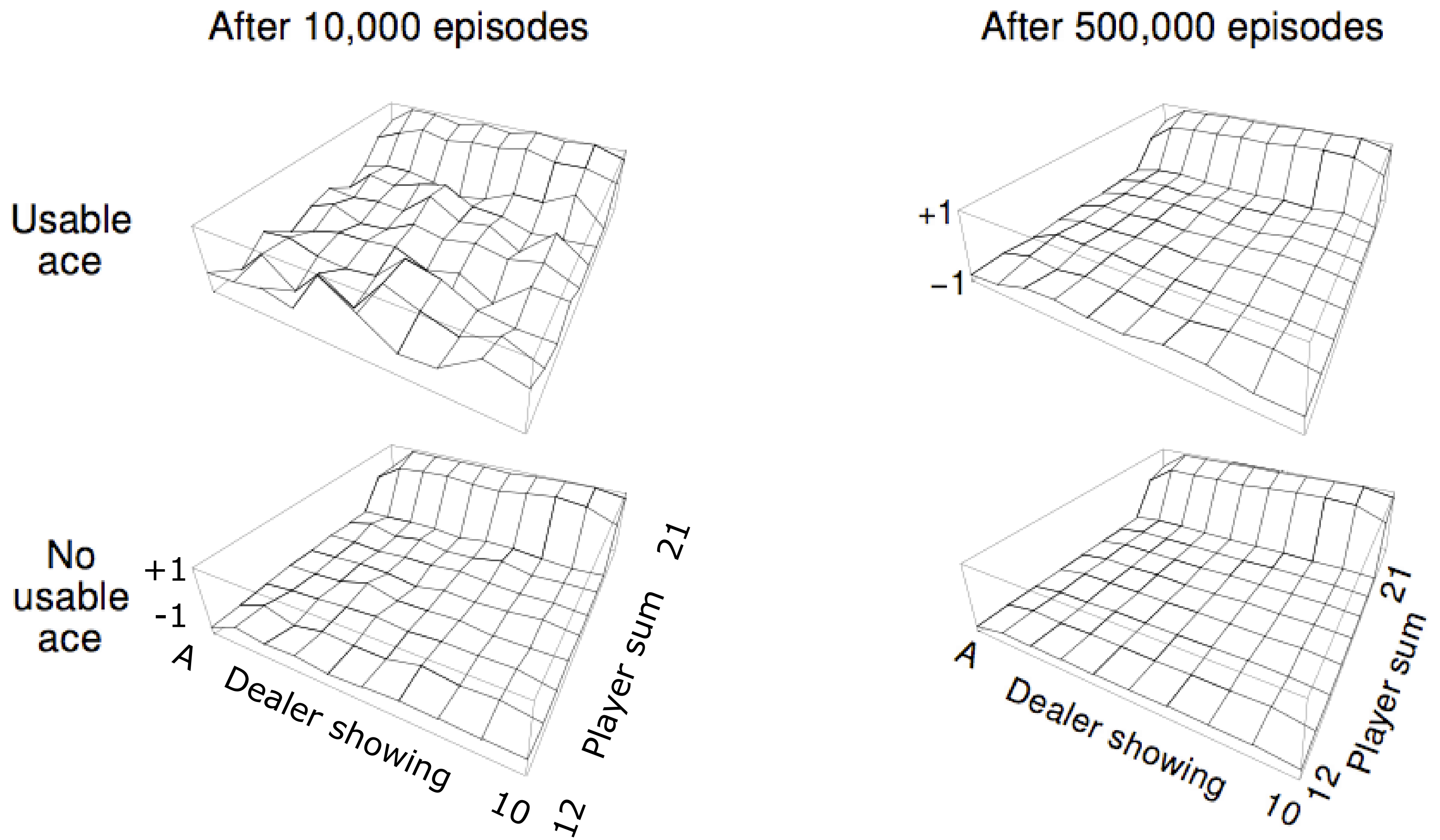
$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$$

Blackjack example

- Goal: Have your card sum be greater than the dealer's without exceeding 21
- State space (200 states):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace? (yes/no)
- Reward: +1 for winning, 0 for a draw, -1 for losing
- Actions: stick (stop receiving cards), hit (receive another card)
- Policy: Stick if my sum is 20 or 21, else hit
- No discounting ($\gamma = 1$)

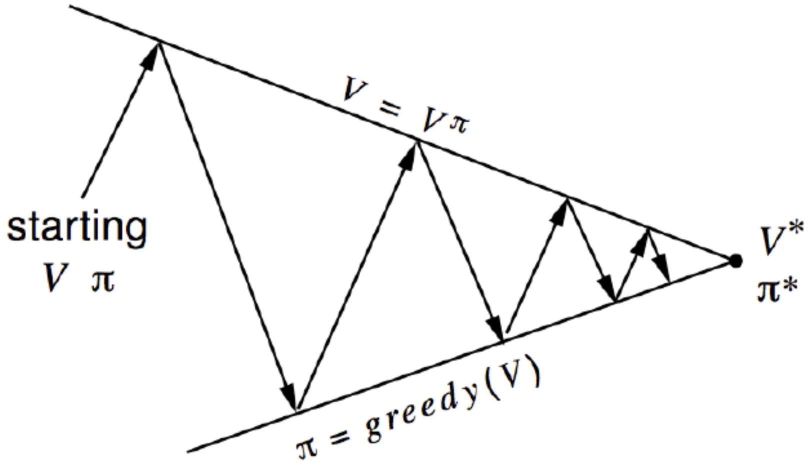


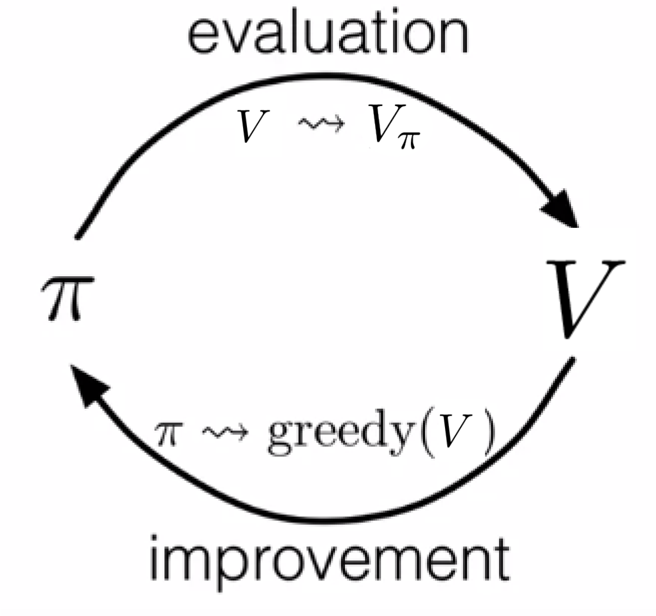
Blackjack example (2)



Policy Iteration

Let's go back to Generalized Policy Iteration



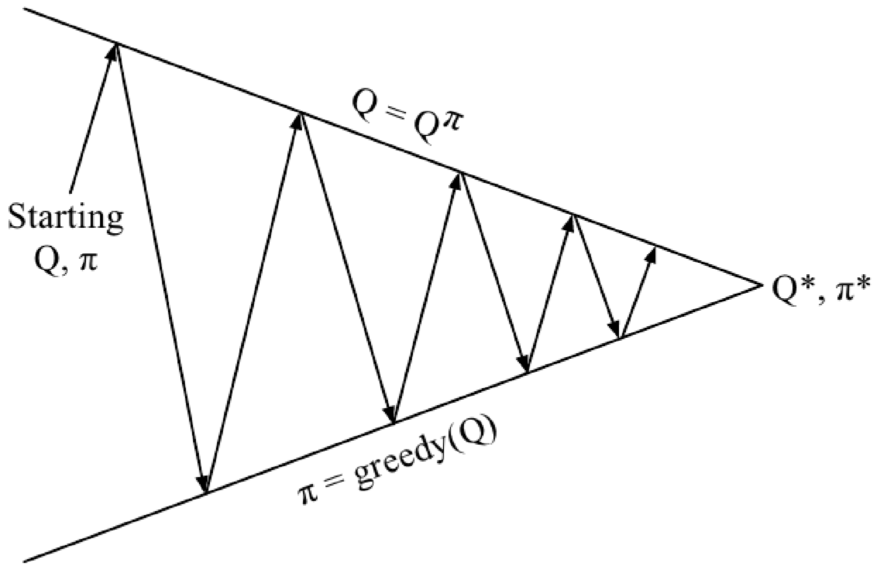


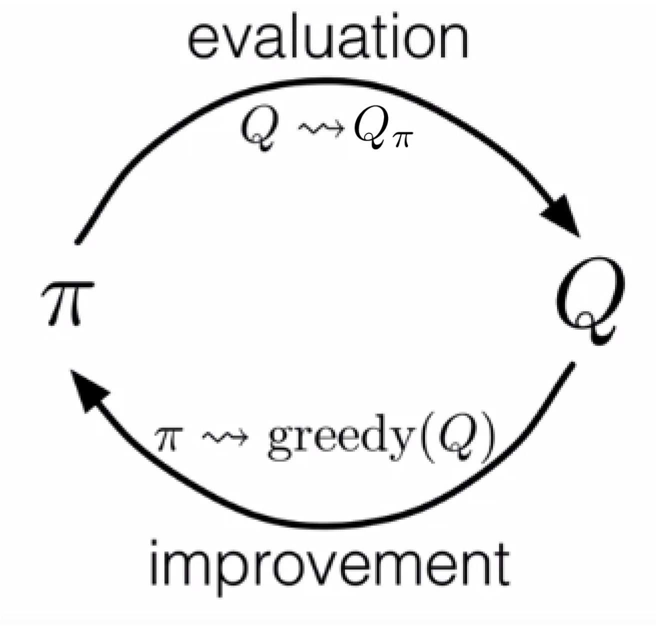
- Evaluation
  - Monte Carlo Policy Evaluation
- Improvement

$$\pi'(s) = \arg\max_a \left\{ \boxed{r(s,a)} + \gamma \sum_{s' \in \mathcal{S}} \boxed{p(s'|s,a)} V_\pi(s') \right\}$$

? ?

Let’s go back to Generalized Policy Iteration





- Evaluation
  - Monte Carlo Action Values Evaluation
- Improvement

$$\pi'(s) = \arg\max_a Q_\pi(s,a)$$

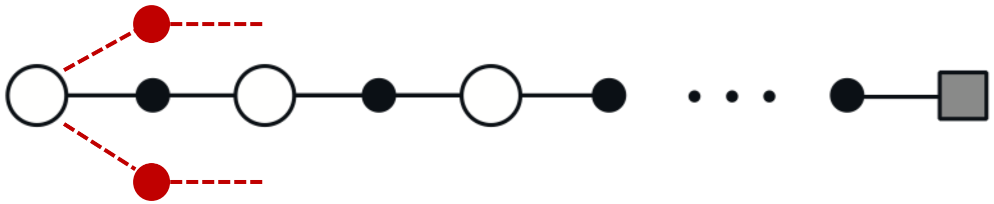
Monte Carlo Action Values Evaluation

- We average return starting from state *s* and action *a* following π:

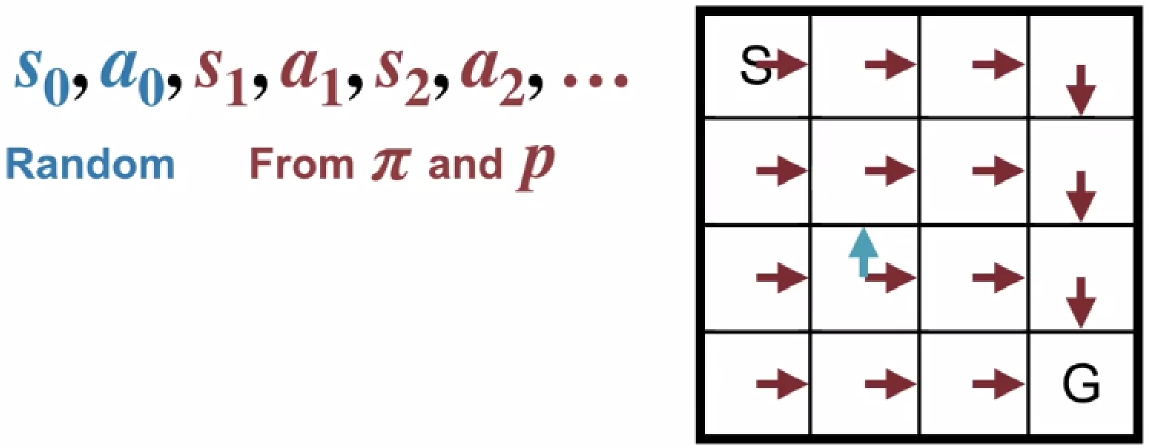
$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t=s, A_t=a] \Rightarrow Q_\pi(s,a) \approx average[G_t|S_t=s, A_t=a]$$

- Converges asymptotically **if every** state-action pair is visited:

Exploration



- Exploring Starts



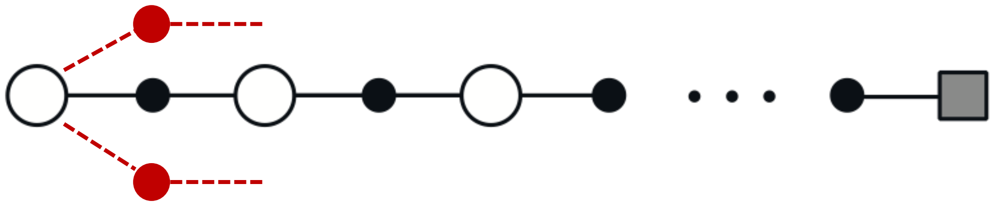
Monte Carlo Action Values Evaluation

- We average return starting from state *s* and action *a* following π:

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \Rightarrow Q_\pi(s,a) \approx average[G_t | S_t = s, A_t = a]$$

- Converges asymptotically **if every** state-action pair is visited:

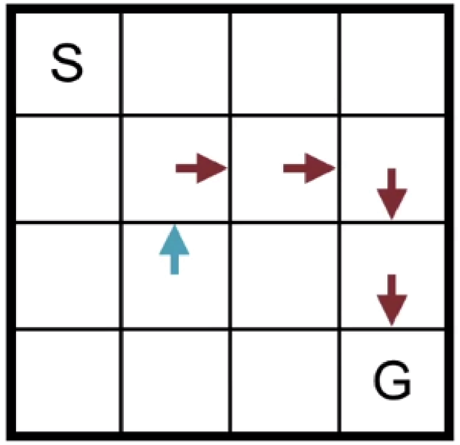
Exploration



- Exploring Starts

$s_0, a_0, s_1, a_1, s_2, a_2, \ldots$

Random   From π and p



Monte Carlo Policy Iteration with Exploring Starts

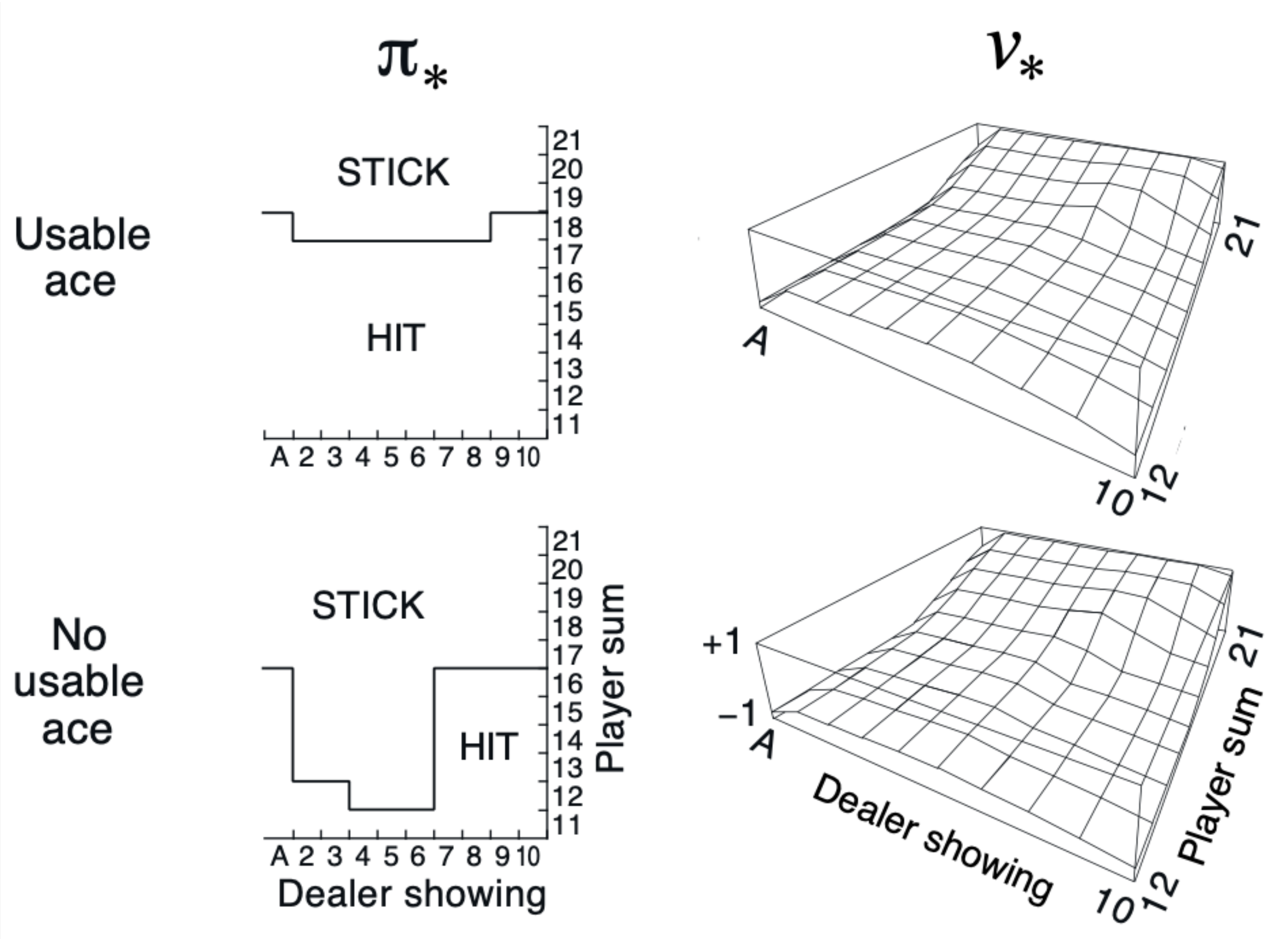
Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

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: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Blackjack example

- We start from the same policy previously described
- Here is the policy we found with MC and exploring starts



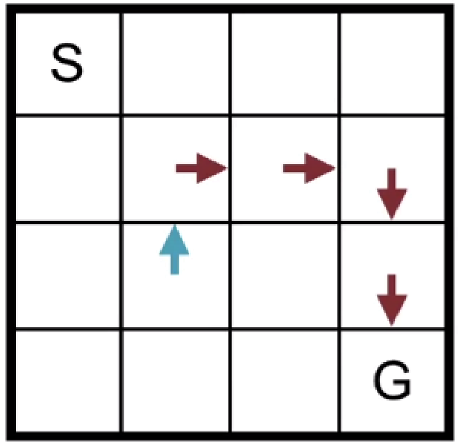
ε-Soft Monte Carlo Policy Iteration

Why ε-Soft Policies?

- Exploring starts is a simple idea but it is not always possible:

$s_0, a_0, s_1, a_1, s_2, a_2, \ldots$

Random   From π and p

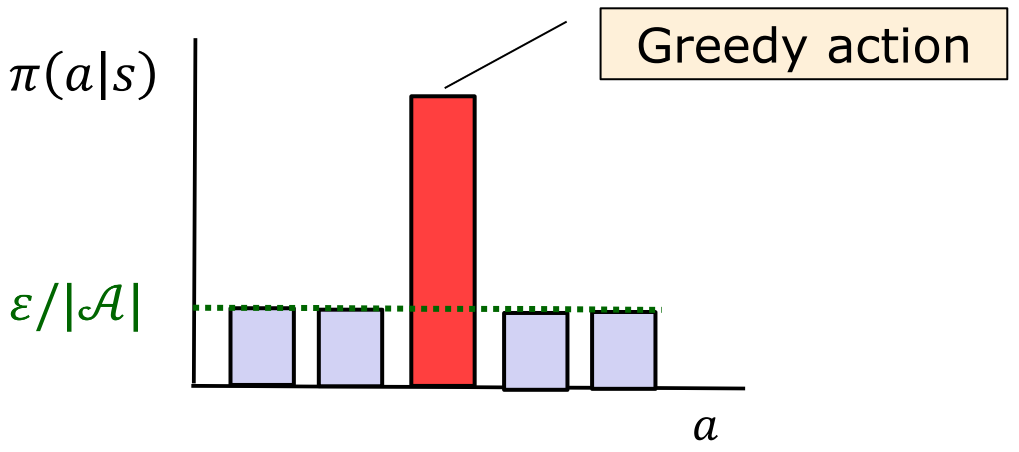


- But, we need to keep exploring during the learning process!
- This leads to a key problem in RL: the **Exploration-Exploitation Dilemma**

ε-Greedy Exploration

- It is the simplest solution to the exploration-exploitation dilemma
- Instead of searching the optimal deterministic policy we search the optimal **ε-soft policy**, i.e., a policy that selects each action with a proability that is at least $\varepsilon/|\mathcal{A}|$
- In particular we use **ε-greedy** policy:

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



ε-Soft Monte Carlo Policy Iteration

Algorithm parameter: small $\varepsilon > 0$

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: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t, A_t)$

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

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

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

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

ε-Greedy Policy Improvement

- **Thorem**
  Any ε-greedy policy π' with respect to Q_π is an improvement over any ε-soft policy π
- **Proof**

$$
\begin{aligned}
V_\pi(s) &= Q_\pi(s, \pi'(s)) \\
&= \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} Q^\pi(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q_\pi(s,a) + (1-\epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} Q_\pi(s,a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a) = V_\pi(s)
\end{aligned}
$$

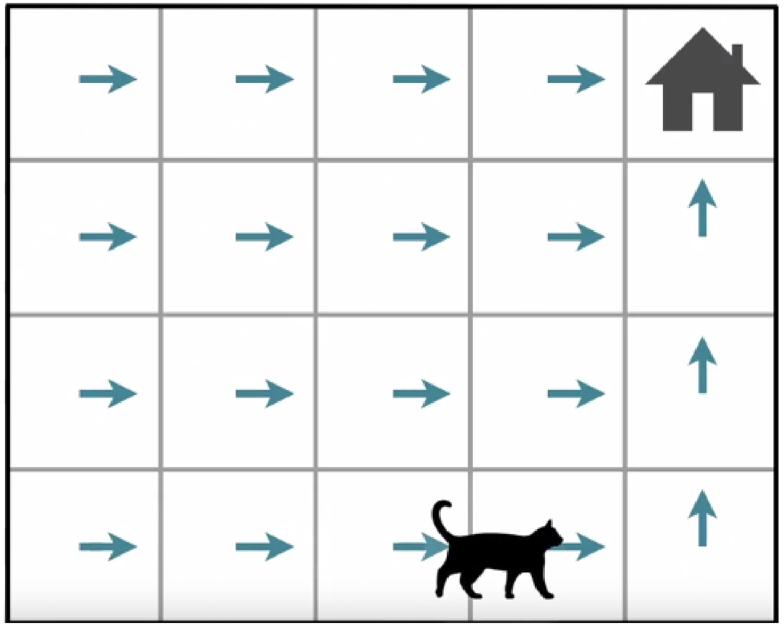
Off-policy Learning

On-Policy vs Off-Policy Learning

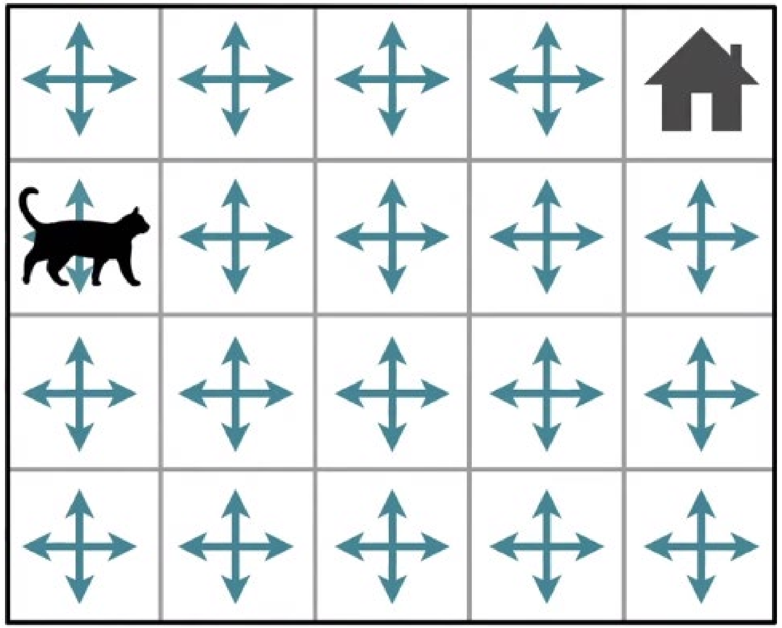
- On-Policy Learning
  - The agent learns the value functions of the **same** policy used to select actions
  - Exploration/Exploitation Dilemma: we cannot easily learn an optimal deterministic policy
- Off-Policy Learning
  - The agent select action using a **behavior policy** $b(a|s)$
  - These data is used to learn the value functions of a **different target policy** $\pi(a|s)$
  - While $b(a|s)$ can be an explorative policy, the agent can learn an optimal deterministic policy $\pi^*(a|s)$

Target and Behavior Policy

Target Policy: $\pi(a|s)$



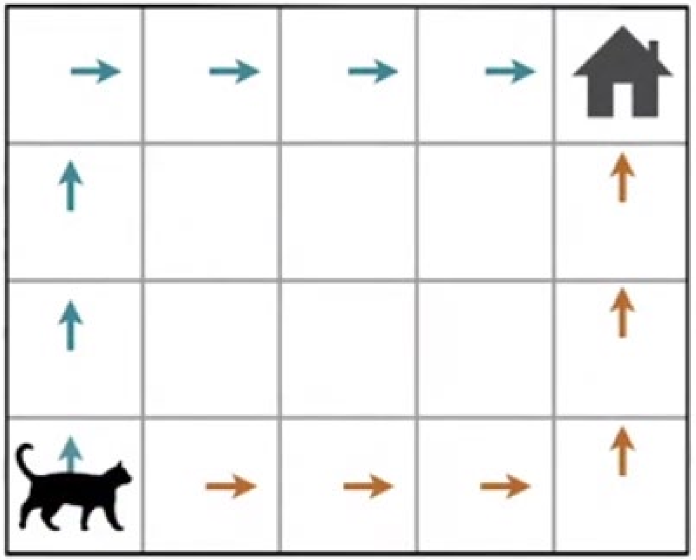
Behavior Policy: $b(a|s)$



Target and Behavior Policy (2)

- We can only learn a target policy that is covered by behavior policy:

$$\pi(a|s) > 0 \textbf{ where } b(a|s) > 0$$



- How is this possible? **Importance Sampling**

Importance Sampling

- Importante sampling allow to estimate expectation of a distribution **different** w.r.t. the distribution used to draw the samples

$$\rho(x) = \frac{p(x)}{q(x)}$$

$$\mathbb{E}_p[x] = \sum_{x \in X} x p(x) = \sum_{x \in X} x \frac{p(x)}{q(x)} q(x) = \sum_{x \in X} x \rho(x) q(x) = \mathbb{E}_q[x \rho(x)]$$

- Accordingly, we can use this for sample-based estimate:

$$\mathbb{E}_p[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i \text{ if } x_i \sim p(x) \quad \Rightarrow \quad \mathbb{E}_p[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i \rho(x_i) \text{ if } x_i \sim q(x)$$

Importance Sampling for Policy Evaluation

- When following policy π we computed the state value function as:

$$V_\pi(s) \approx average(Returns[0], Returns[1], Returns[2], \dots)$$

- But when following policy b, the value function becomes:

$$V_\pi(s) \approx average(\rho_0 Returns[0], \rho_1 Returns[1], \rho_2 Returns[2], \dots)$$

- Where ρ_i is the probability of performing the trajectory observed in episode i while following policy π over the probability of observing the same trajectory while following policy b

$$\rho = \frac{Prob[\text{trajectory under } \pi]}{Prob[\text{trajectory under } b]}$$

Importance Sampling for Policy Evaluation (2)

$$
\begin{aligned}
\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\}
&= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&= \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

$$
\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}
$$

Ordinary Sampling vs Weighted Sampling

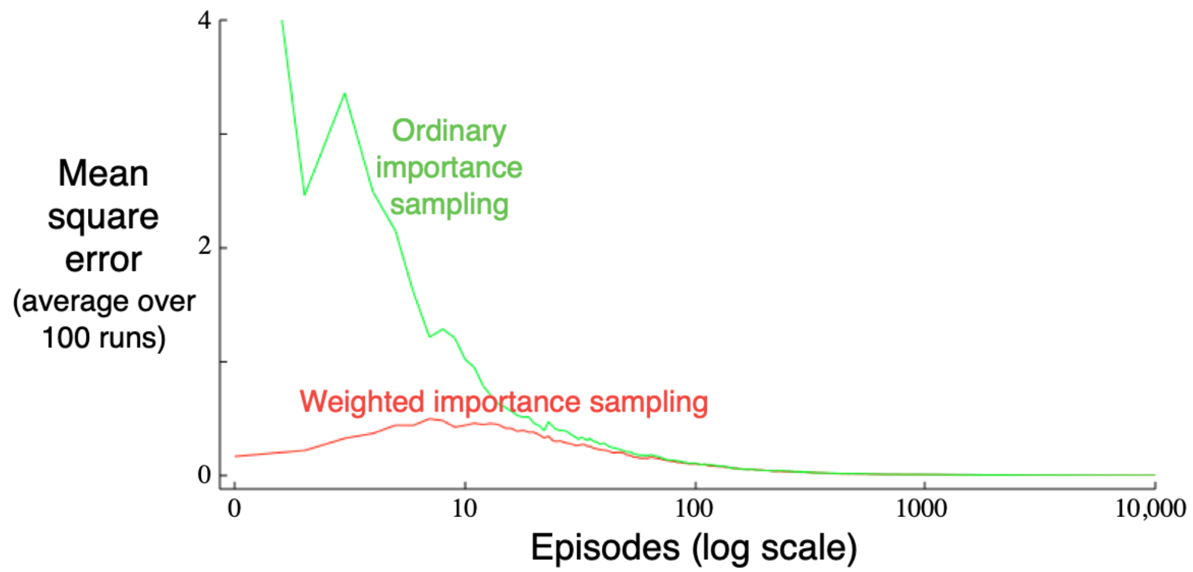
- Ordinary sampling

$$V_\pi(s) \approx \frac{\sum_i \rho[i] \cdot Return[i]}{N(s)}$$

  - Unbiased
  - Higher Variance
- Weighted sampling

$$V_\pi(s) \approx \frac{\sum_i \rho[i] \cdot Return[i]}{\sum_i \rho[i]}$$

  - Biased (bias converges to zero)
  - Lower Variance



Off-Policy every visit MC prediction (ordinary sampling)

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Loop forever (for each episode):

Generate an episode following $b : S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0 \quad W \leftarrow 1$

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

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

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$