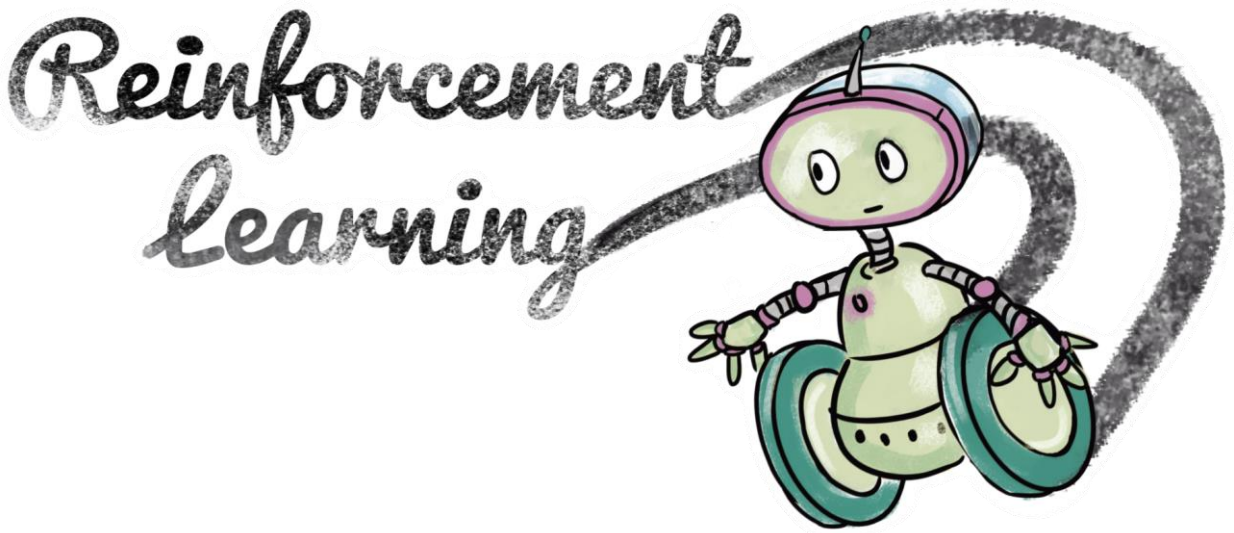


Monte Carlo Methods (MCM)

Mohammad **Dehghani**

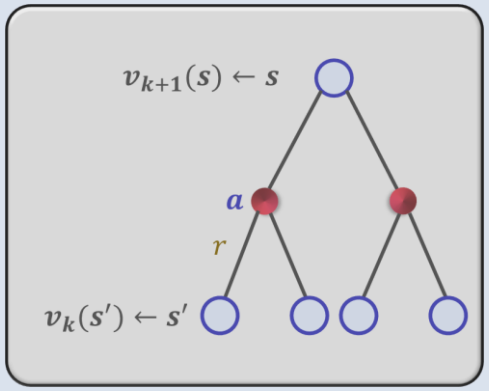
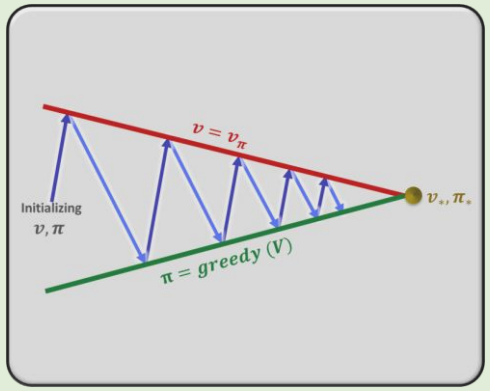
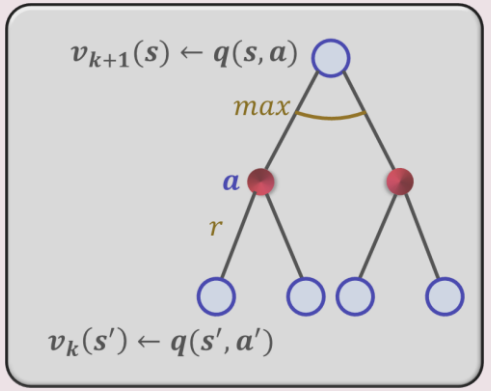
Mechanical and Industrial Engineering Department

Northeastern University

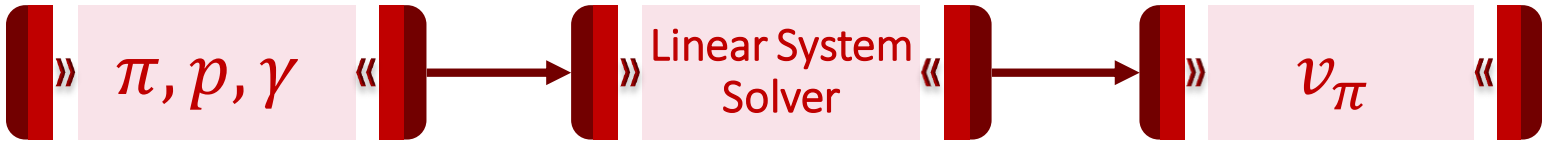
Feb-20

Introduction

Dynamic Programming Algorithms

	Policy Evaluation	Policy Iteration	Value Iteration
Algorithm			
Bellman Equation	Bellman Expectation Equation	Bellman Expectation Equation + Greedy Policy Improvement	Bellman Optimality Equation
Problem	Prediction	Control	Control

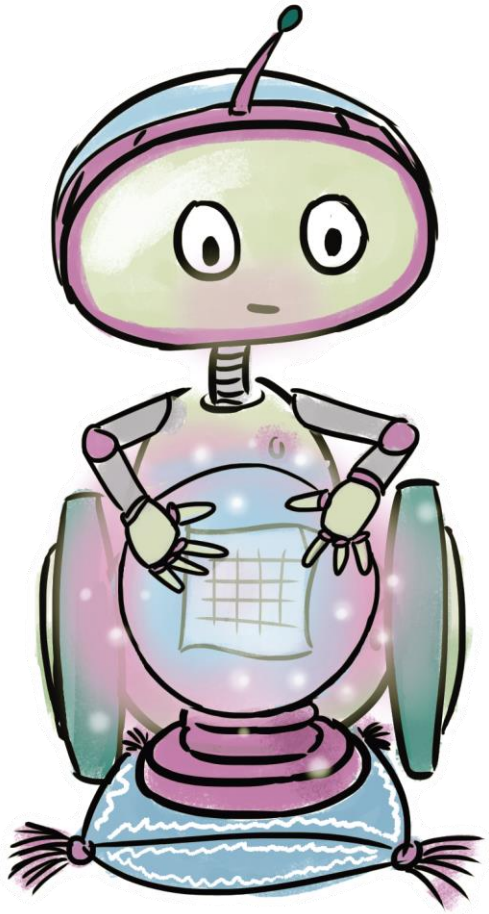
Simple MDPs: Linear Programing



General MDP: Dynamic Programing

MDP with unknown transition model: Monte Carlo Methods

Monte Carlo Methods (MCM)



MONTE CARLO METHODS

Monte Carlo Methods

History

(a) Monte Carlo, Monaco

(b) Monte Carlo, Casino

Law of Large Numbers (LLN)

Given an independent and identically distributed (i.i.d) sequence of random variables $X_1, X_2, \cdots, X_n$ with $\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\mathbb{E}[X_i] = \mu$, then for every $\varepsilon > 0$

$$P\left(\left|\overline{X}_n - \mu\right| > \varepsilon\right) \to 0,$$

as $n \to \infty$

- **While nothing is more uncertain than the duration of a single life, nothing is more certain than the average duration of a thousand lives.**

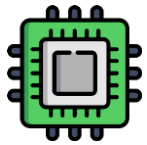
~Elizur Wright (1804 - 1885),

Introduction

The term *Monte Carlo* is often used more broadly for any estimation method whose operation involves a significant random component.

Why do we say Monte Carlo methods?

- Simply because the same principle can be used to solve different problems and to each one of these problems is associated a different technique or algorithm.
- What all these algorithm have in common is their use of ***random (or stochastic) sampling.***

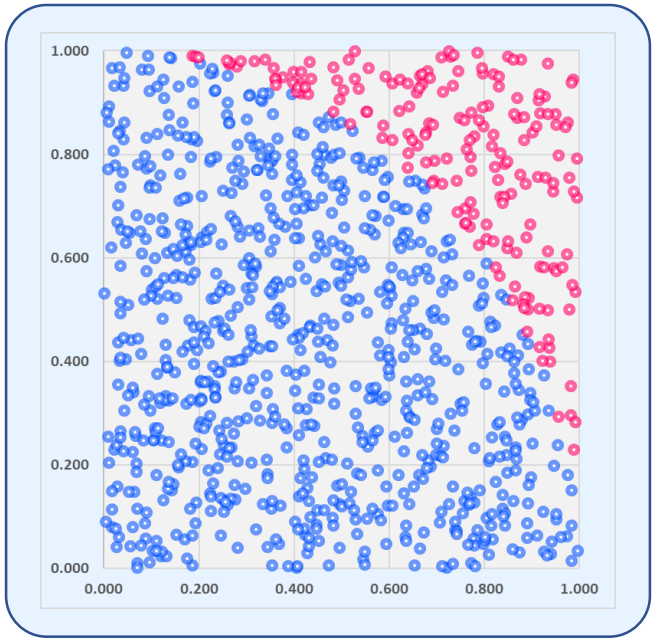


Advancement of computing technology in the 1940s, made Monte Carlo methods' popular

Example: Application in Practice (1)

- **Calculating π**
  - Hit-or-Miss Monte Carlo Method

$$\frac{Area_{\circ}}{Area_{\square}} = \frac{\pi r^2}{(2r)(2r)} = \frac{\pi}{4}$$

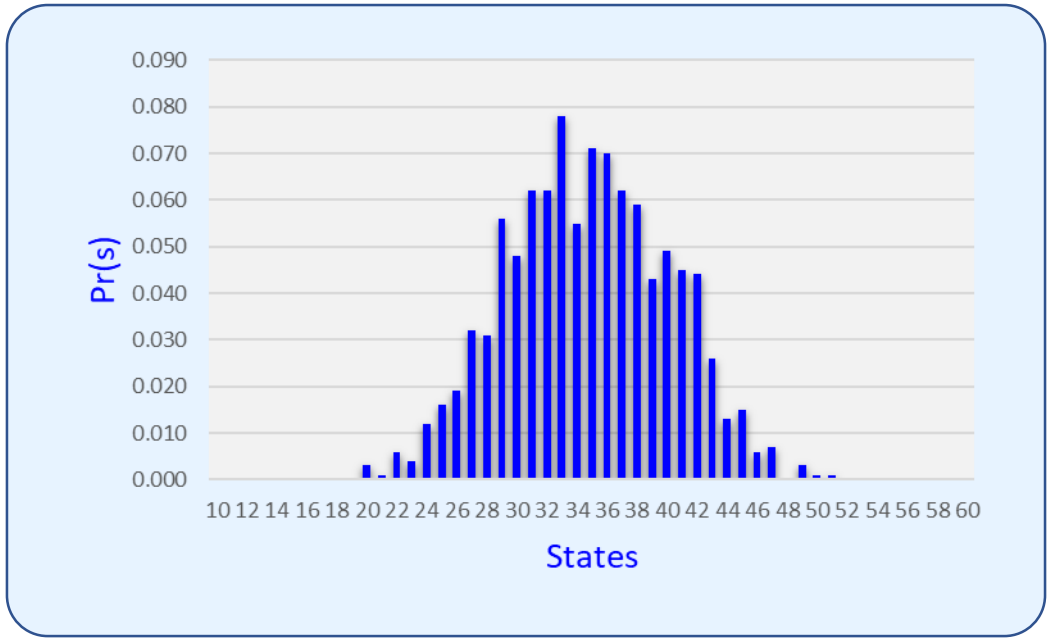


ExceLab 5-1

  - Is it an efficient way!!?
    - 100 million samples is required to approximate the number π to its fourth decimal (3.1415)

Example: Application in Practice (2)

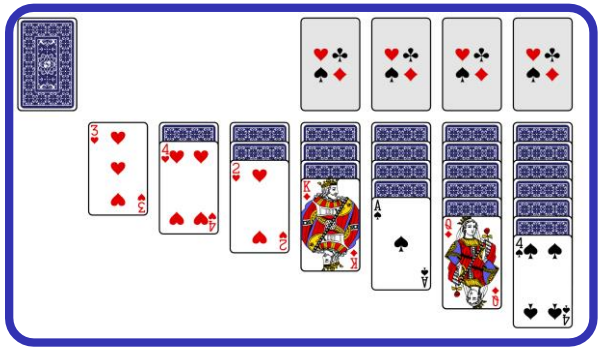
- **Throwing a die 10 times**
  - (51) States: Total of outcomes
    - **Min:** 10, **Max:** 60
    - Num of transition functions: 51×51
  - Monte Carlo Estimation



ExceLab 5-1

Example: Application in Practice (3)

- **Combinatorial problems**
  - Many equations do not have such closed-form solutions
  - or, complexity is such that they could only be solved given infinite time (intractable)
    - However, it's often better to have some predictions about the possible outcomes of a given problem, than not having any prediction at all
- **Solitaire**
  - To calculate the probability of a successful outcome of a game of solitaire is a completely intractable task.
    - The practical procedure is to produce a large number of examples of any given game and then to examine the relative proportion of successes.



MCM Requirements

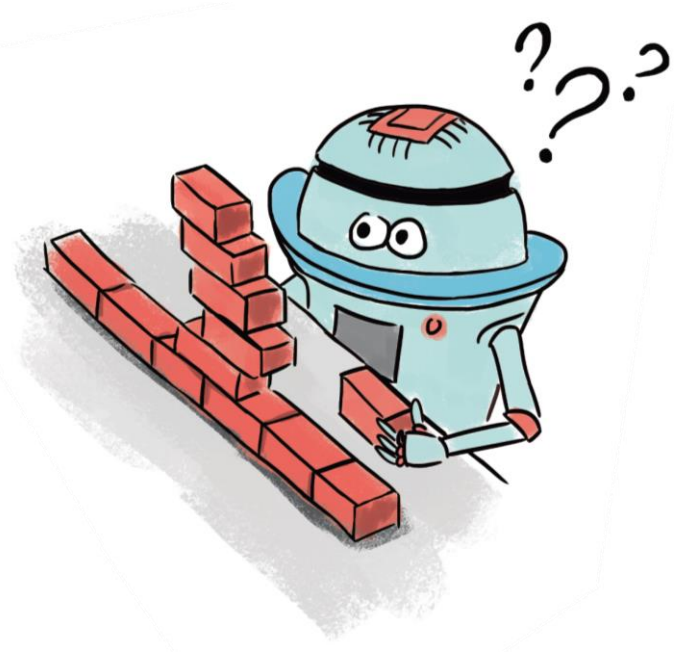
- **Monte Carlo methods require only *experience***
  - *Sample sequences of states, actions, and rewards from actual or simulated interaction with an environment*
  - *Requires no prior knowledge of the environment's dynamics*

- ***MCM train from not all the states and actions but the trajectories.***
  - *Trajectory is a sequence of*
    - *States* (s)
    - *Actions* (a)
    - *Rewards* (r)
  - *Monte Carlo samples trajectories*

- **Model-based Monte Carlo**
  - Learn directly from episodes of experience
  - The *simulated* experience
    - uses the model only to generate ***sample transitions***
    - does not require the complete probability distributions of all possible transitions

  - For ***Prediction***:
    - Input: MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma \rangle$ and policy π
    - Output: value function v_π

  - For ***Control***:
    - Input: MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma \rangle$
    - Output:
      I. Optimal value function v_*
      II. Optimal policy π_*



General Policy Iteration (GPI)

- **To handle the nonstationarity, we adapt the idea of general policy iteration (GPI)**
  - Whereas there we computed value functions from knowledge of the MDP, here we learn value functions from sample returns with the MDP

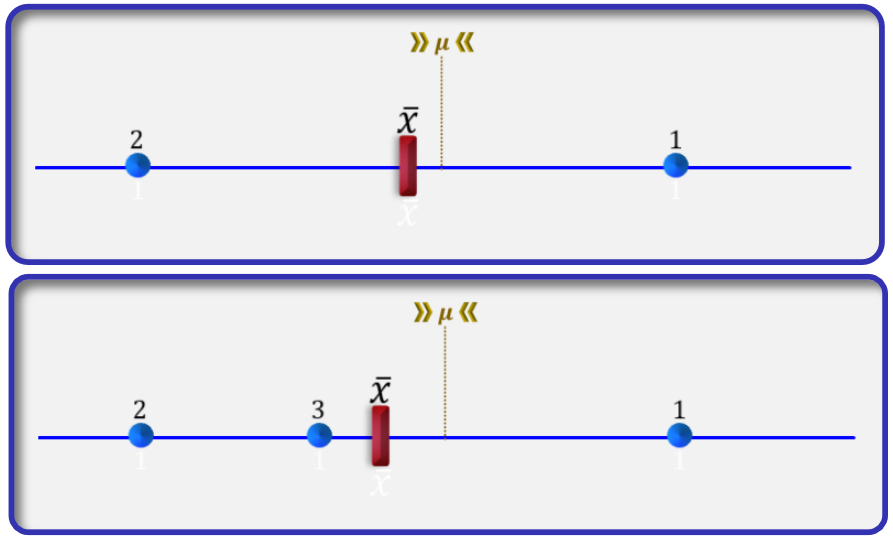
1. **Policy evaluation**
   - calculating value function for an arbitrary policy
2. **Policy Improvement**
   - obtain the new policy based on the new selected action
3. **Policy Iteration**
   - calculating an optimal value function (and policy) by iteratively calculating value function and then improving policy

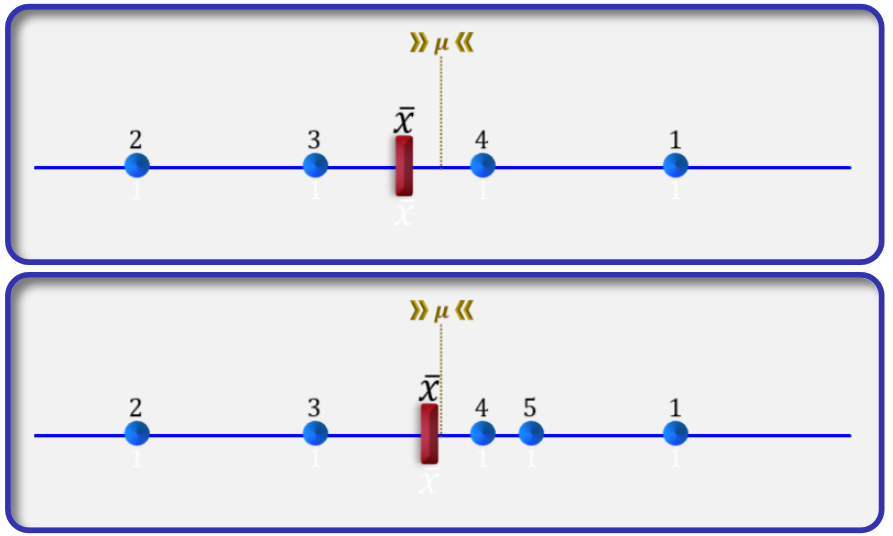
Monte Carlo Prediction

Monte Carlo Prediction

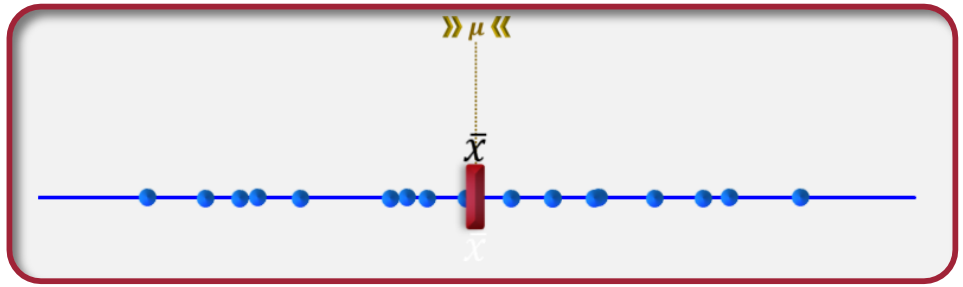
Monte Carlo Methods for Policy Evaluation

- **Each occurrence of state s in an episode is called a visit to s.**





- **As more returns are observed, the average should converge to the expected value.**



Monte Carlo Methods

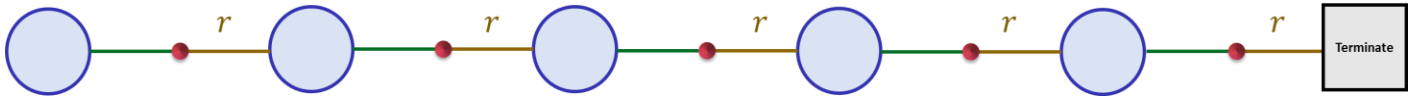
$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$$

- **Learning in Monte Carlo method is based on observations from states**
  - We observe multiple returns from the same state
  - To estimate the expected return from that state, we average over all observed returns

- **Convergence:**
  - **Classical probability rule:** As the number of samples increases, the spread of the distribution becomes more narrow
  - **Samples = episodes:** as the number of samples from episodes increases, the average tends to get closer and closer to the expected return.

- **Returns Calculation**

$$Go = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5$$



  - In Monte Carlo, returns can only be observed at the end of an episode **(Episodic Tasks)**

Monte Carlo Methods

Monte Carlo Methods for Policy Evaluation

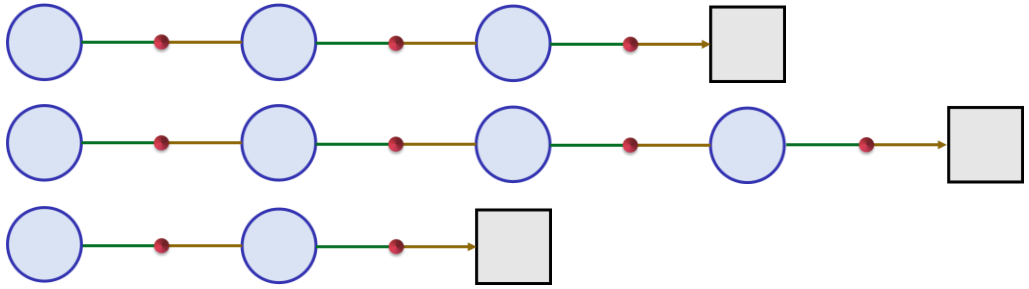
Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$

Monte Carlo Methods

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

- **General Goal of RL is to maximize *expected cumulative future discounted reward***
- **Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns.**
  - Same concept but with sampling approach
- **Monte Carlo methods can thus be incremental in an *episode-by-episode* sense, but not in a *step-by-step* (online) sense**

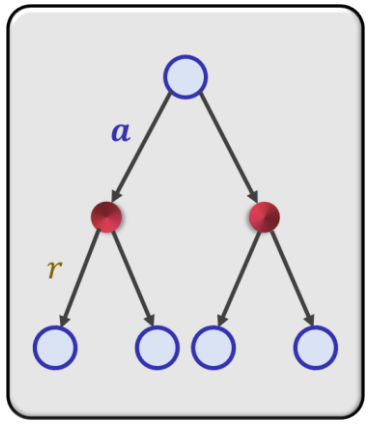


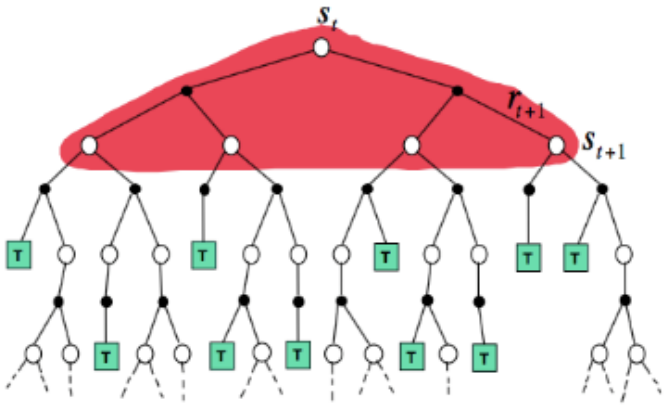
Monte Carlo Methods

Back up Diagram

Dynamic Programming

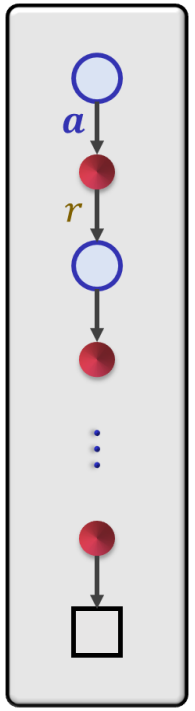
$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$

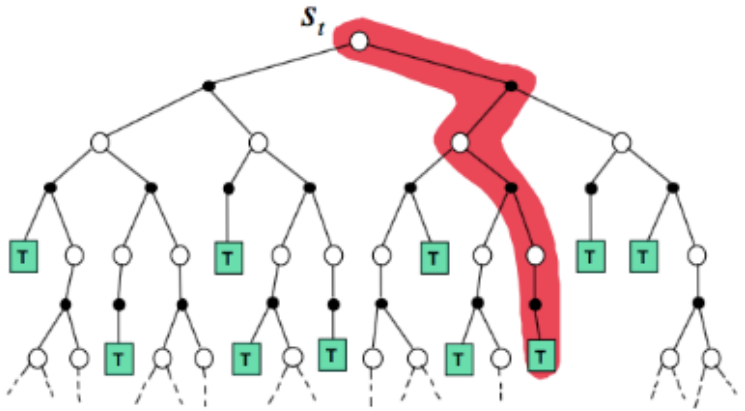




Monte Carlo Methods

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



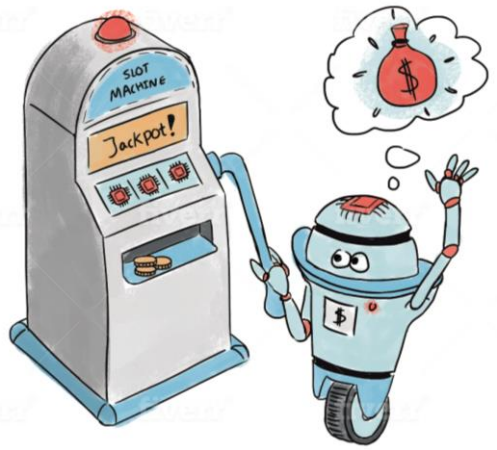


Monte Carlo Methods

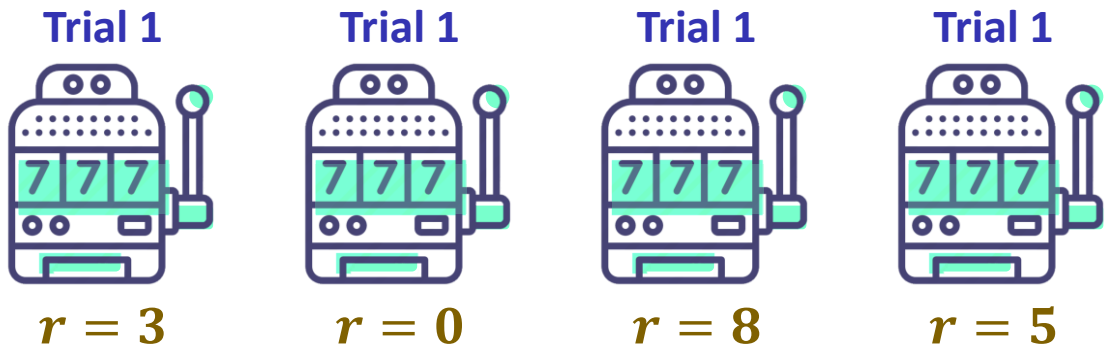
Contextual Bandits (1)

- **Monte Carlo methods sample and average returns for each state–action pair much like the bandit methods**
  - A contextual bandit is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$
  - At each step t Environment generates
    - State $s_t \in \mathcal{S}$
    - Agent selects action $a_t \in \mathcal{A}$
    - Environment generates reward $r_t \sim r(a_t, s_t)$

- **Associative-search or contextual bandit**
  - Multiple States
    - States are interrelated.
  - Nonstationary problem



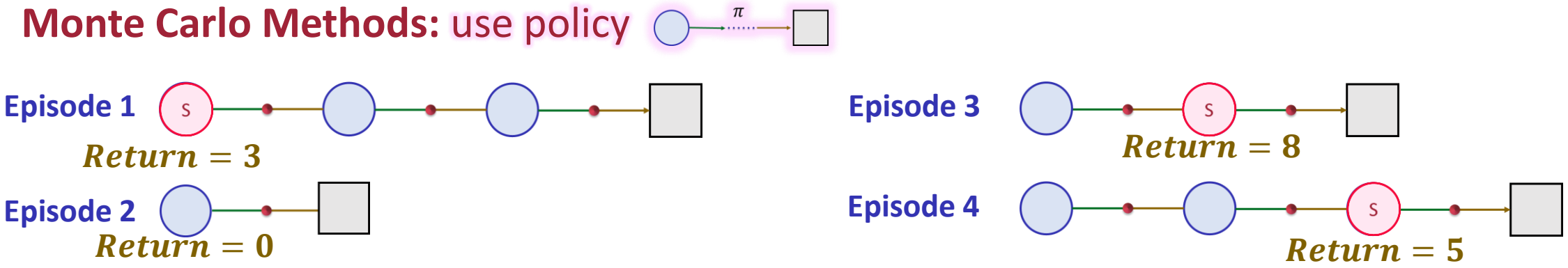
- **In Bandits:** use arm



$$Q_t(a) \doteq \frac{\sum_{i-1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i-1}^{t-1} \cdot \mathbb{1}_{A_i=a}}$$

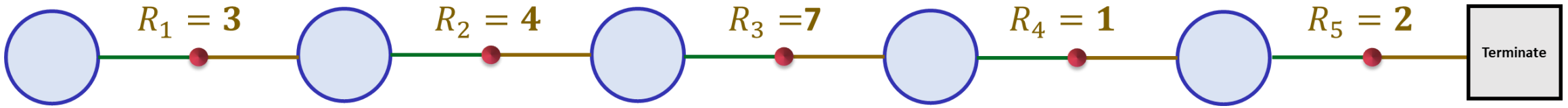
$$Q_t(a) = \frac{3+0+8+5}{4} = 4$$

- **Monte Carlo Methods:** use policy



$$v_\pi(s) = \frac{3+0+8+5}{4} = 4$$

- **Monte Carlo allows us to estimate values directly from experience**



$\gamma = 0.5$

$$Go = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5 = R_1 + \gamma G_1 = 7$$

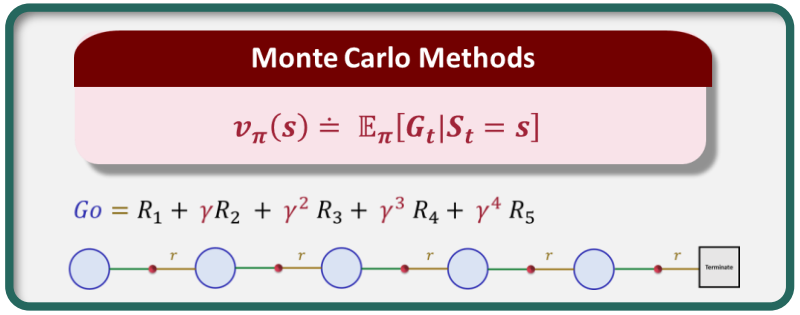
$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_4 + \gamma^3 R_5 = R_2 + \gamma G_2 = 8$$

$$G_2 = R_3 + \gamma R_4 + \gamma^2 R_5 = R_3 + \gamma G_3 = 8$$

$$G_3 = R_4 + \gamma R_5 = R_4 + \gamma G_4 = 2$$

$$G_4 = R_5 = R_5 + \gamma G_5 = 2$$

$$G_5 = 0$$

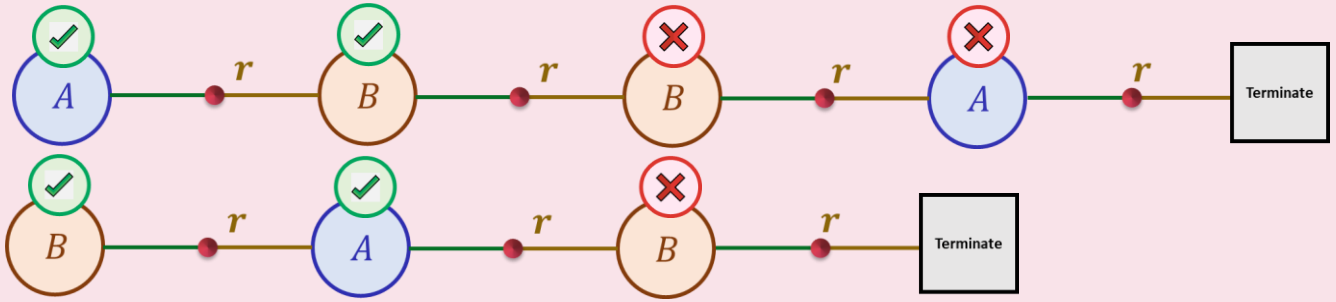


Monte Carlo Prediction

First-visit MC vs Every-visit MC

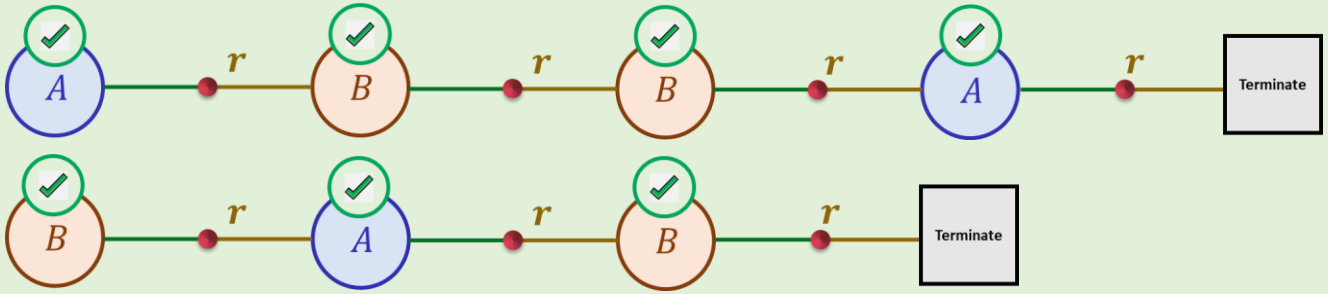
First-visit MC

- The first-visit MC method estimates $v_\pi(s)$ as the average of the returns following *first visits* to s



Every-visit MC

- To estimates estimates $v_\pi(s)$, the *every-visit* MC method averages the returns following all visits to s



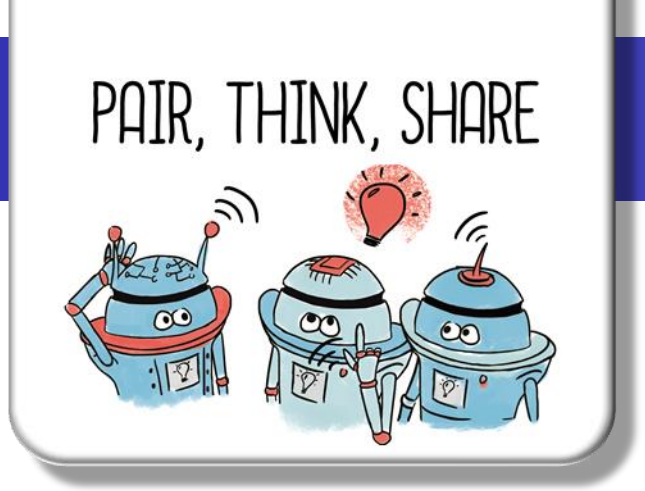
Monte Carlo Prediction

Convergence of First-visit MC and Every-visit MC

- **Both *first-visit MC* and *every-visit MC* converge to $v_\pi(s)$ as the number of visits (or first visits) to s goes to infinity.**

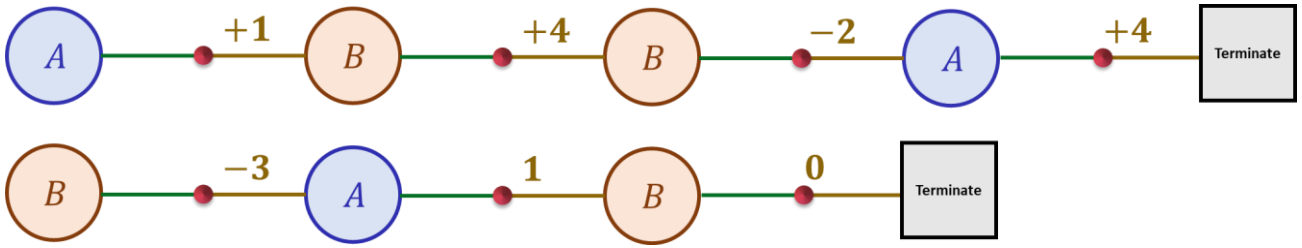
- **Each return is an *independent*, *identically* distributed estimate of $v_\pi(s)$ with finite variance.**

- **Law of large numbers**
  - **The sequence of averages of these estimates converges to their expected value.**
  - **What is the bias and standard deviation of error?**
    - each average is itself an unbiased estimate
    - standard deviation of its error falls as $1/\sqrt{n}$, where n is the number of returns averaged



- **Mystery game:** Suppose there is an environment with 2 states
  - State A
  - State B

Lets say we observed 2 sample episodes



- **Use the *(i) first-visit* MC method, and *(ii) every-visit* MC method to estimate v_A and v_B**
  - $\gamma = 1$

Monte Carlo Prediction

Pseudocode: First-visit MC method

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 $Returns(S_t)$

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

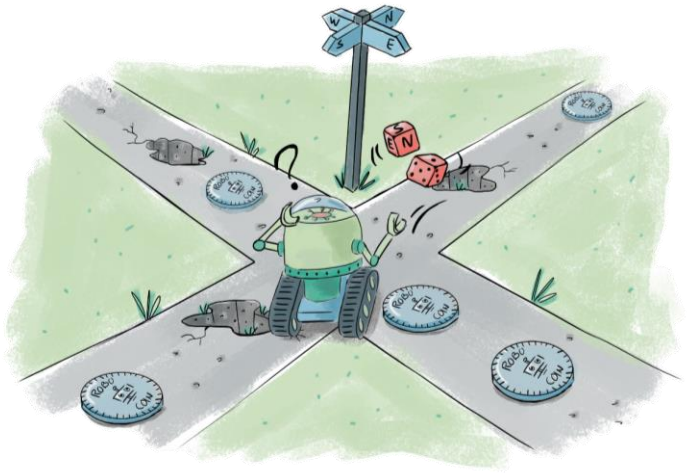
$$Q_{n+1} \doteq Q_n + \frac{1}{n}[R_n - Q_n] \qquad NewEstiamte \leftarrow OldEstimate + StepSize\ [Target - OldEstiamte]$$

Monte Carlo Prediction

Lab 2-a: The Small Gridworld Example

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

States



ExceLab 5-2

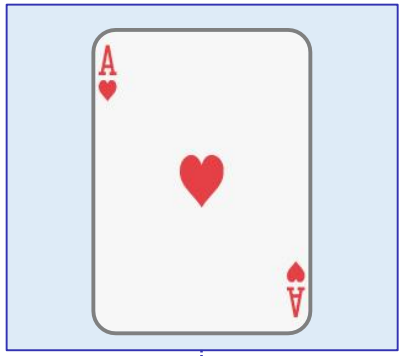
	-10	-10	-10
-1	-1	-1	-1
-10	-10	-10	-1
-1	-1	-1	-1

Reward

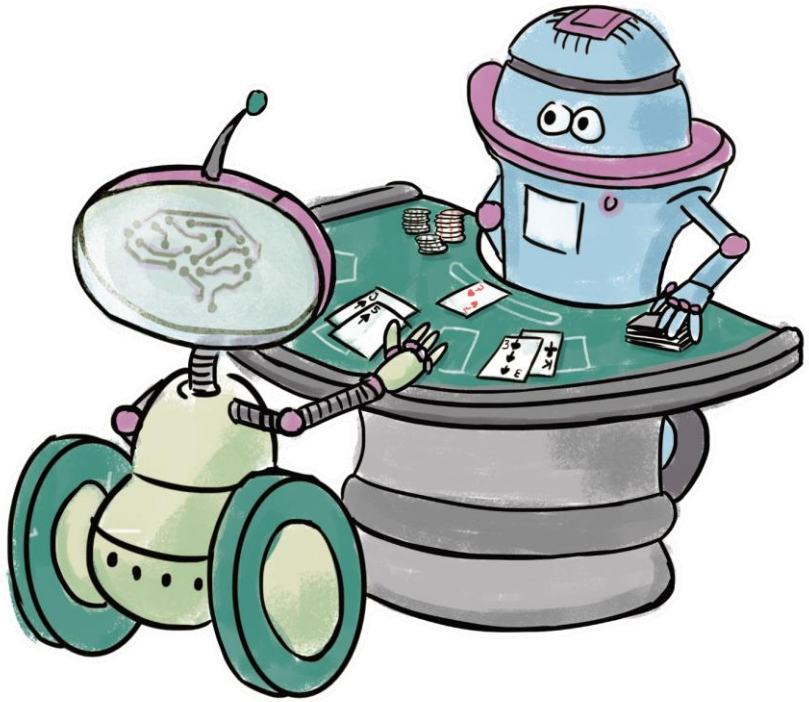
Monte Carlo Prediction

Example: Blackjack

Face Cards = 10



Ace = 1 or 11



Monte Carlo Prediction

Blackjack Problem Formulation

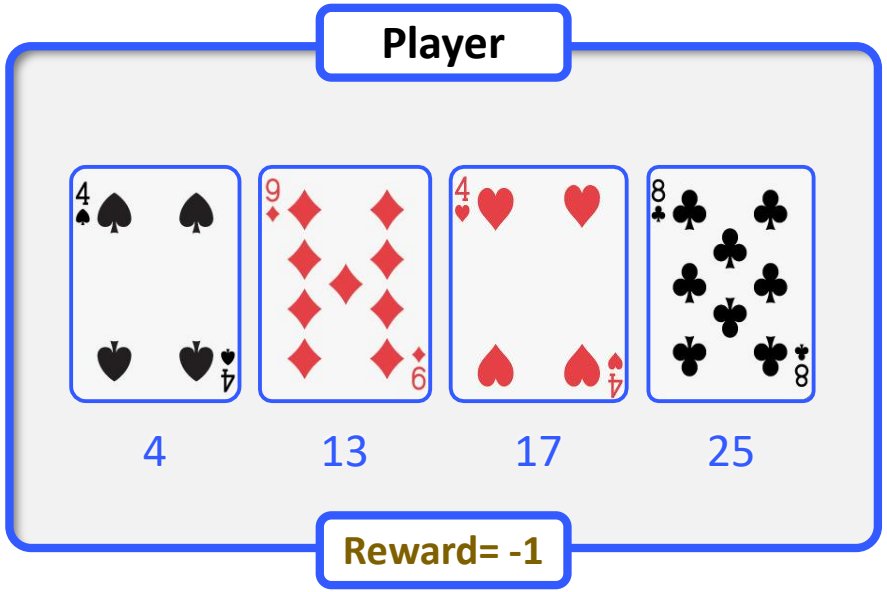
- **Blackjack is a Undiscounted MDP**
  - **Reward:**
    - -1 for loss
    - 0 for a draw
    - 1 for a win
  - **Actions:**
    - **Hit** or **Stick**
  - States
    - Player sum: $(12:21) \rightarrow 10$
    - Dealer Card: $(Ace:10) \rightarrow 10$
    - Total states: $(10 \times 10) \rightarrow 100$

- **Policy:**
  - Any arbitrarily policy:
    - **Deterministic Policy:** **stick** when the player's sum is 20 or 21
    - **Stochastic Policy:** when the player's sum is 18 or greater: $P(\boldsymbol{Stick} = 0.8)$, $P(\boldsymbol{Hit} = 0.2)$, otherwise $P(\boldsymbol{Stick} = 0.2)$, $P(\boldsymbol{Hit} = 0.8)$

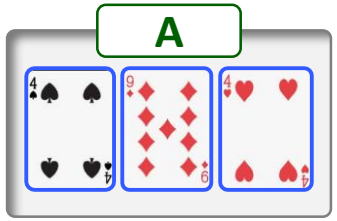
- **Assumptions:**
  - Episodes are independent
  - Cards are dealt from a deck with replacement

Monte Carlo Prediction

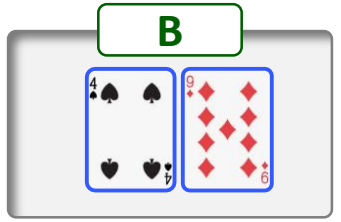
Example: Blackjack



S (Usable Ace, Sum, Dealer)	Returns (S)	V(S)
A= (NoAce, **17**, **10**)	Return(A)=[-1]	-1

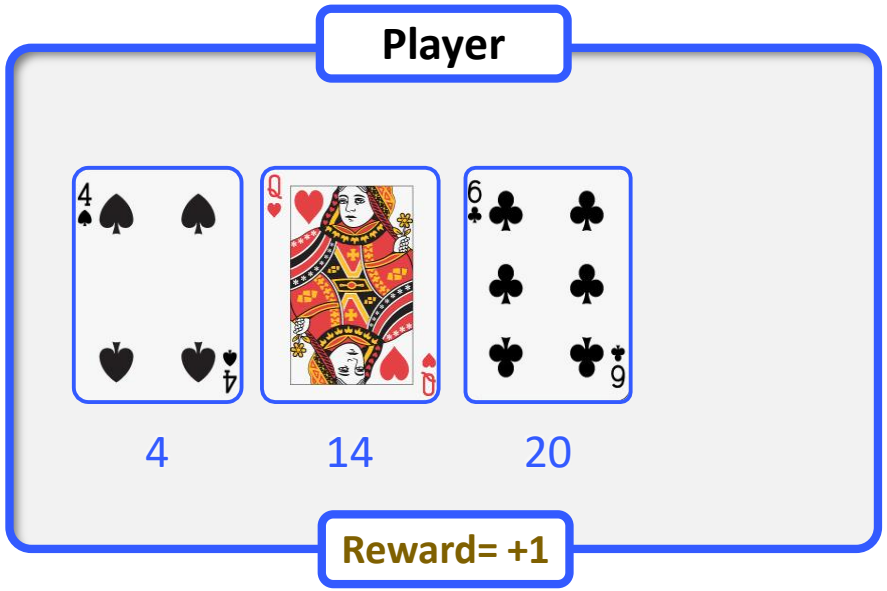


S (Usable Ace, Sum, Dealer)	Returns (S)	V(S)
B= (NoAce, **13**, **10**)	Return(B)=[-1]	-1



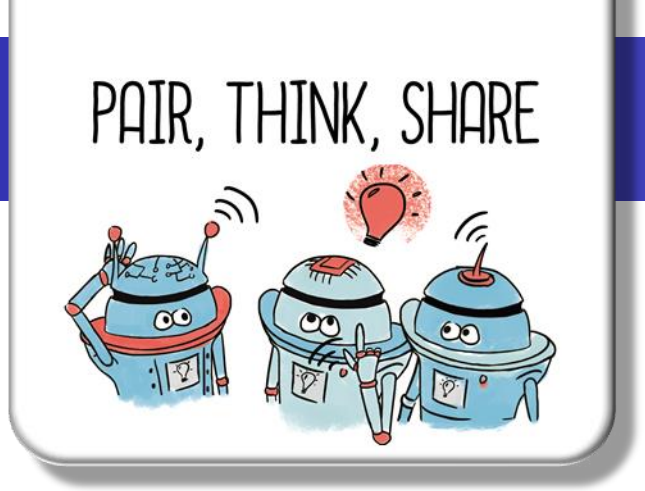
Monte Carlo Prediction

Example: Blackjack



S (Usable Ace, Sum, Dealer)	Returns (S)	V(S)
C= (NoAce, 20, **10**)	Return(C)=[+1]	1

S (Usable Ace, Sum, Dealer)	Returns (S)	V(S)
D= (NoAce, 14, **10**)	Return(D)=[+1]	1



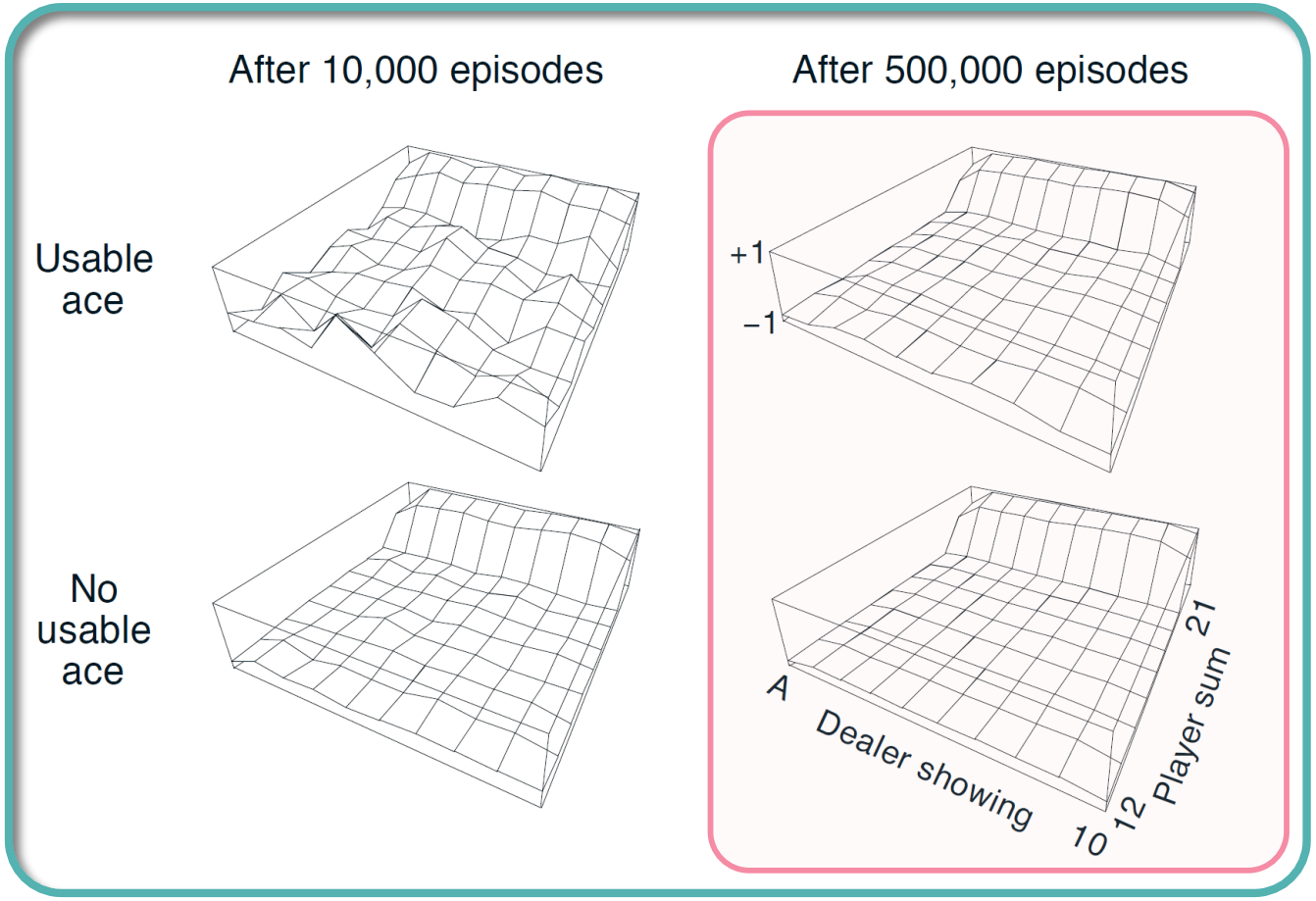
Blackjack : Policy Evaluation results

Consider the diagrams on the right

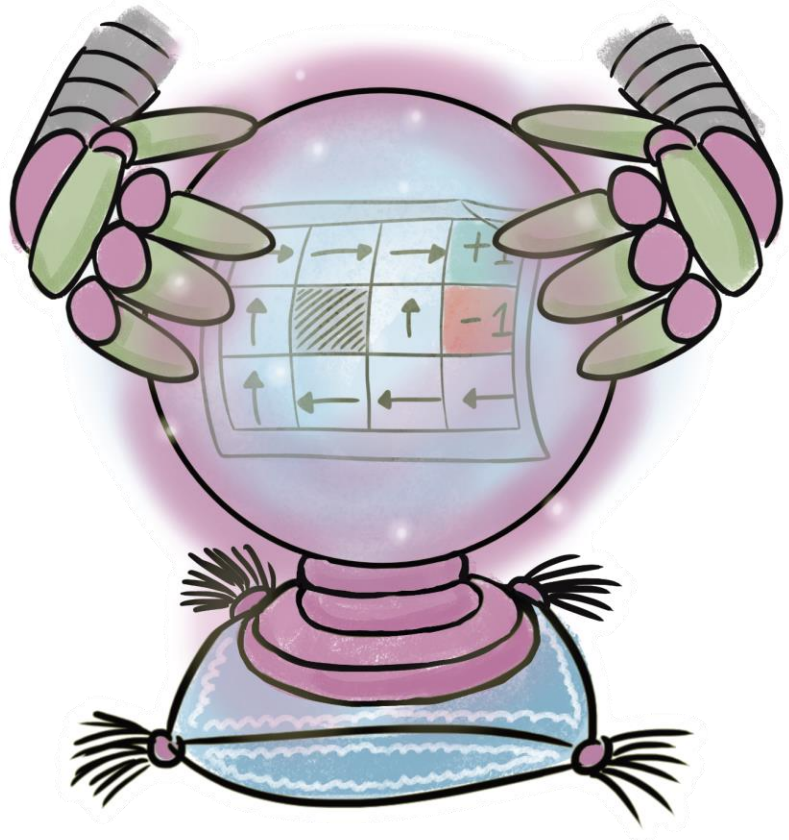
a. Why does the estimated value function jump up for the last two rows in the rear?

b. Why does it drop off for the whole last row on the left? Why are the frontmost values higher in the upper diagram

c. Suppose every-visit MC was used instead of first-visit MC on the blackjack task. Would you expect the results to be very different? Why or why not?



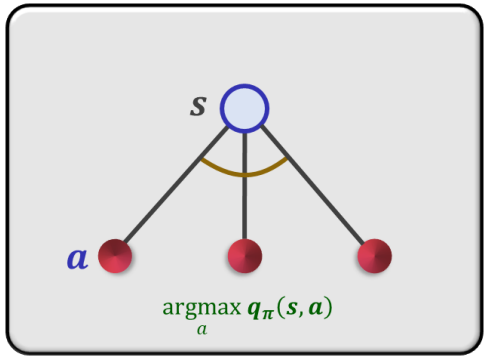
Monte Carlo for Control



Monte Carlo for Control

Monte Carlo Estimation of Action Values

- **Why action Values?**
  - Without a model, state values alone are not sufficient.
  - If a model is not available, we can use action values (rather than state values) to learn a policy.



1. **We can compare multiple action in any state**
2. **Select the best action**

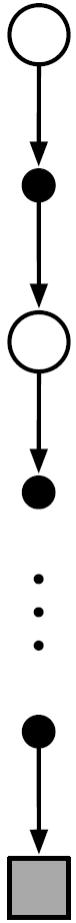
- **Therefore, we need to explicitly estimate the value of each action in order for the values to be useful in suggesting a policy**

» The primary goals for Monte Carlo methods is to estimate q_* «

Monte Carlo for Control

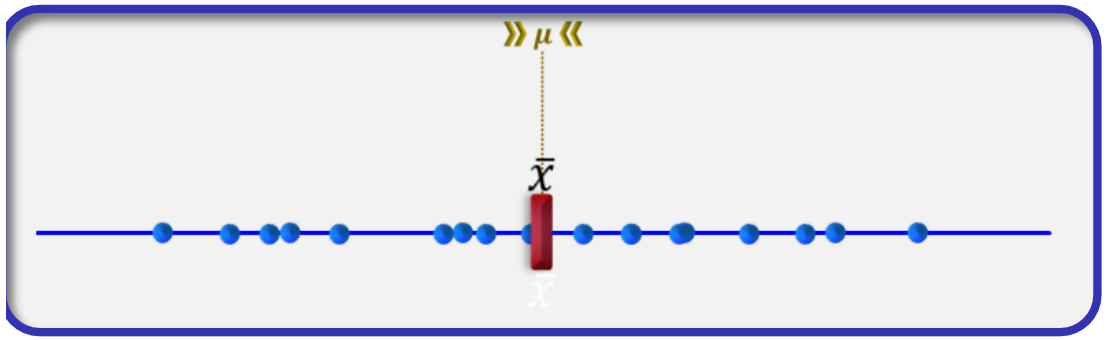
Monte Carlo Estimation of Action Values

- **If a model is not available, then it is particularly useful to estimate action values (the values of state–action pairs) rather than state values.**



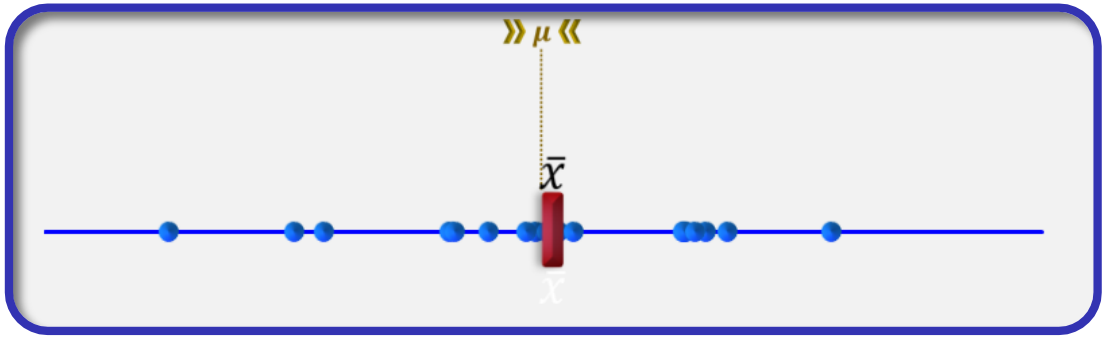
State Value Estimation

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$



Action Value Estimation

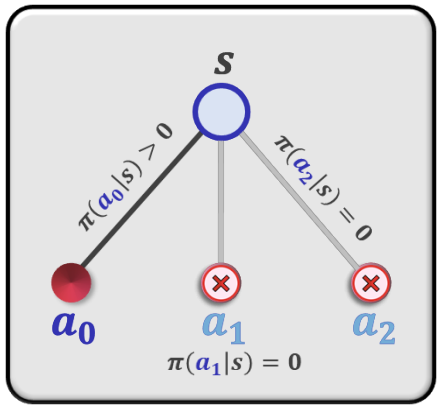
$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$



Monte Carlo for Control

Maintaining exploration

- **So far, our policies have been *deterministic*, mapping s always to $\pi(s)$**
  - **Problem:** won't even see (s, a) if $a \neq \pi(s)$
    - With no returns to average, the Monte Carlo estimates of the other actions will not improve with experience.



  - **Solution:** need π to explore explicitly (problem of *maintaining exploration*)
    - Alternatively, we can use a non-deterministic policy which allows us to explore each state and action infinitely often

» For policy evaluation to work for action values, we must assure *"continual exploration"* «

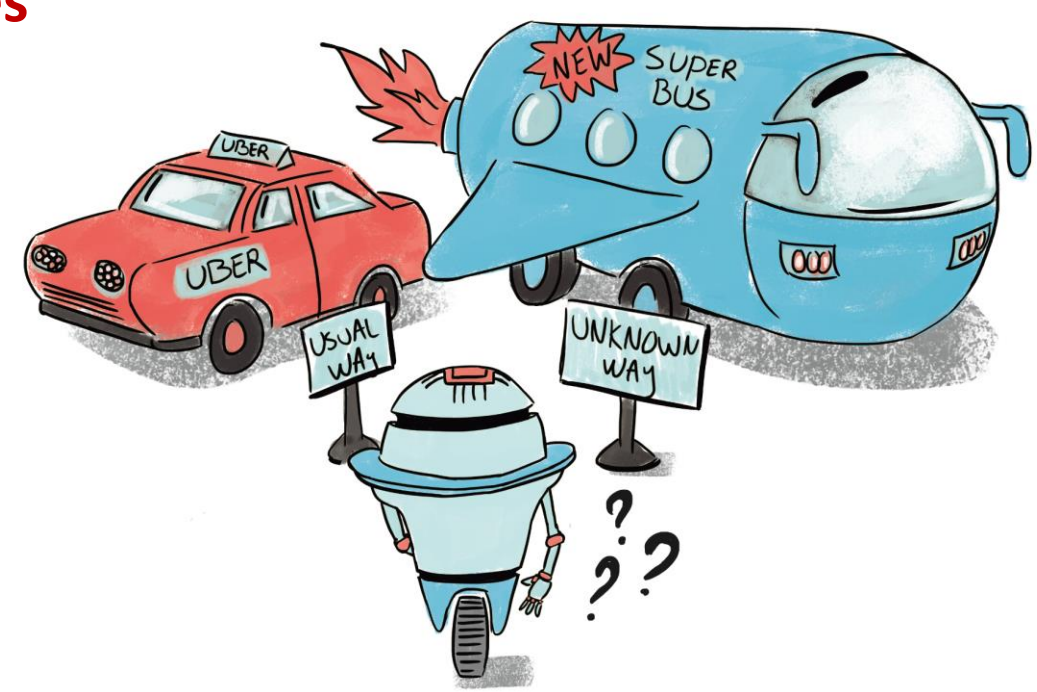
Monte Carlo for Control

Maintaining exploration

- **We discuss to approaches that help us to main exploration**

1. **Exploring Starts**

2. **Stochastic Policies**



Monte Carlo for Control

Idea 1: Exploring Starts

- **One way to maintain exploration is called exploring starts.**
  - Every pair has a nonzero probability of being selected as the start.

1. **In exploring starts, we must guarantee that episodes start in every state-action pair.**
2. **Afterwards, the agent simply follows its policy.**

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

Random — Follow policy π

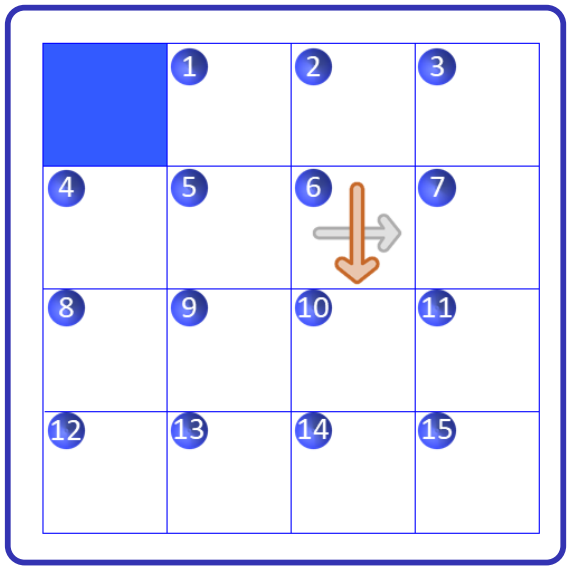
Monte Carlo for Control

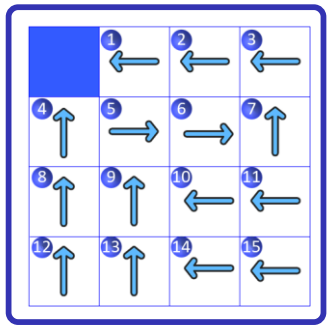
Idea 1: Exploring Starts Example

S_0, A_0

Random

$6, \downarrow$

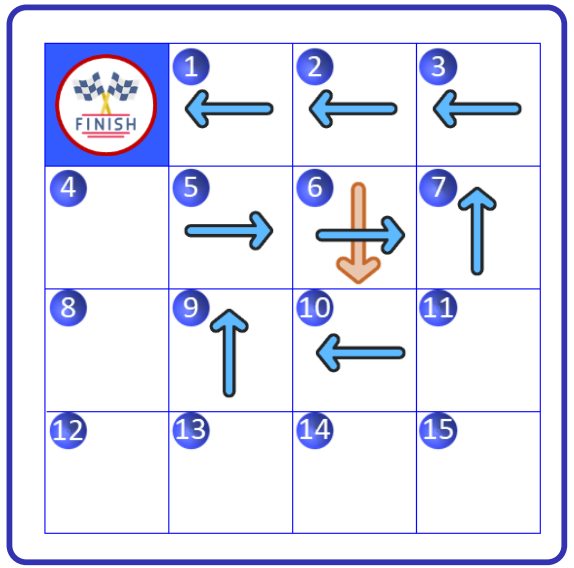




Policy

$S_1, A_1, S_2, A_2, \cdots$

Follow policy π



Monte Carlo for Control

General Policy Iteration (GPI)

- **To handle the nonstationarity, we adapt the idea of general policy iteration (GPI)**
  - Whereas there we computed value functions from knowledge of the MDP, here we learn value functions from sample returns with the MDP

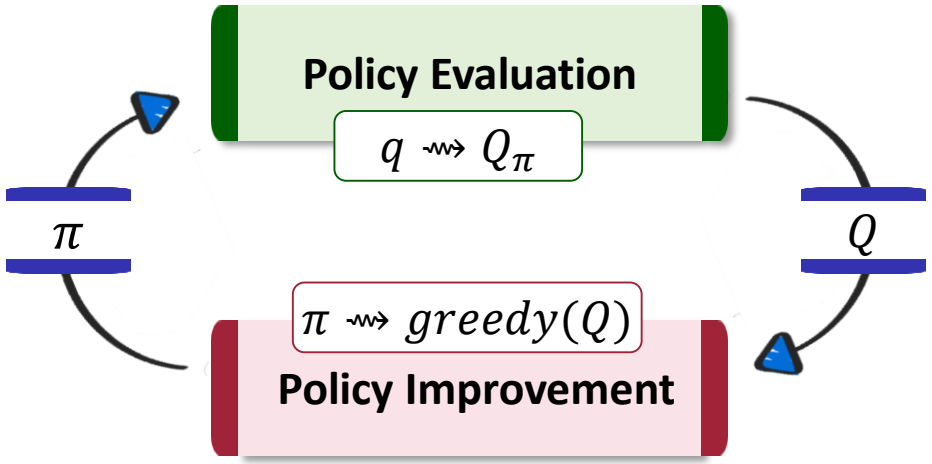
$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$$

$\xrightarrow{I}$ **Improvement**

$$\pi_{k+1}(s) \doteq \operatorname*{argmax}_a q_{\pi_k}(s,a)$$

$\xrightarrow{E}$ **Evaluation**

Monte Carlo Prediction



Monte Carlo Control

Pseudocode: Monte Carlo 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 $Returns(S_t, A_t)$

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

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Exploring Start

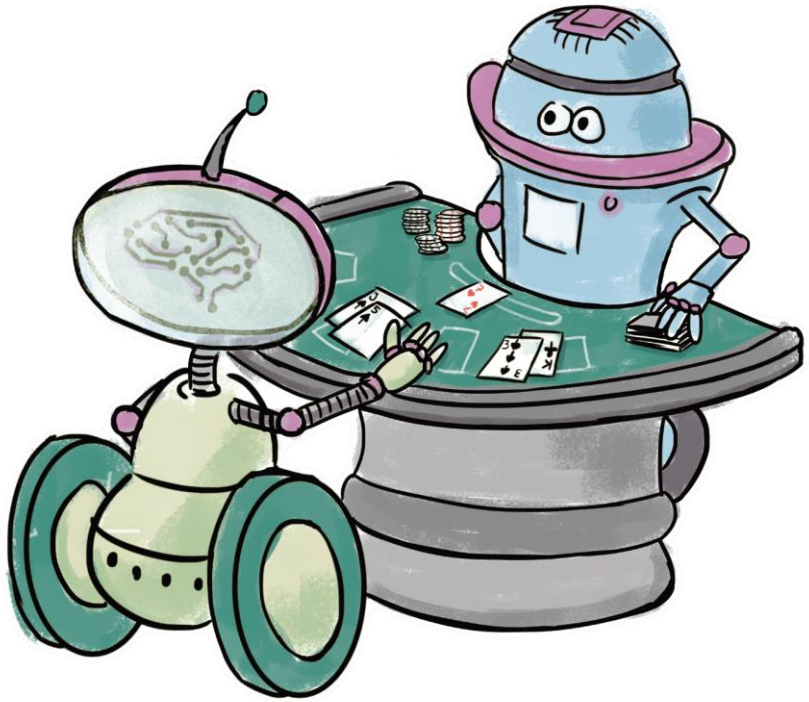
Greedy Policy

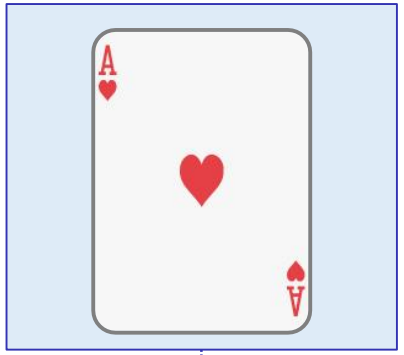
Policy Evaluation

Monte Carlo for Control

Example: Monte Carlo Control for Blackjack

Face Cards = 10

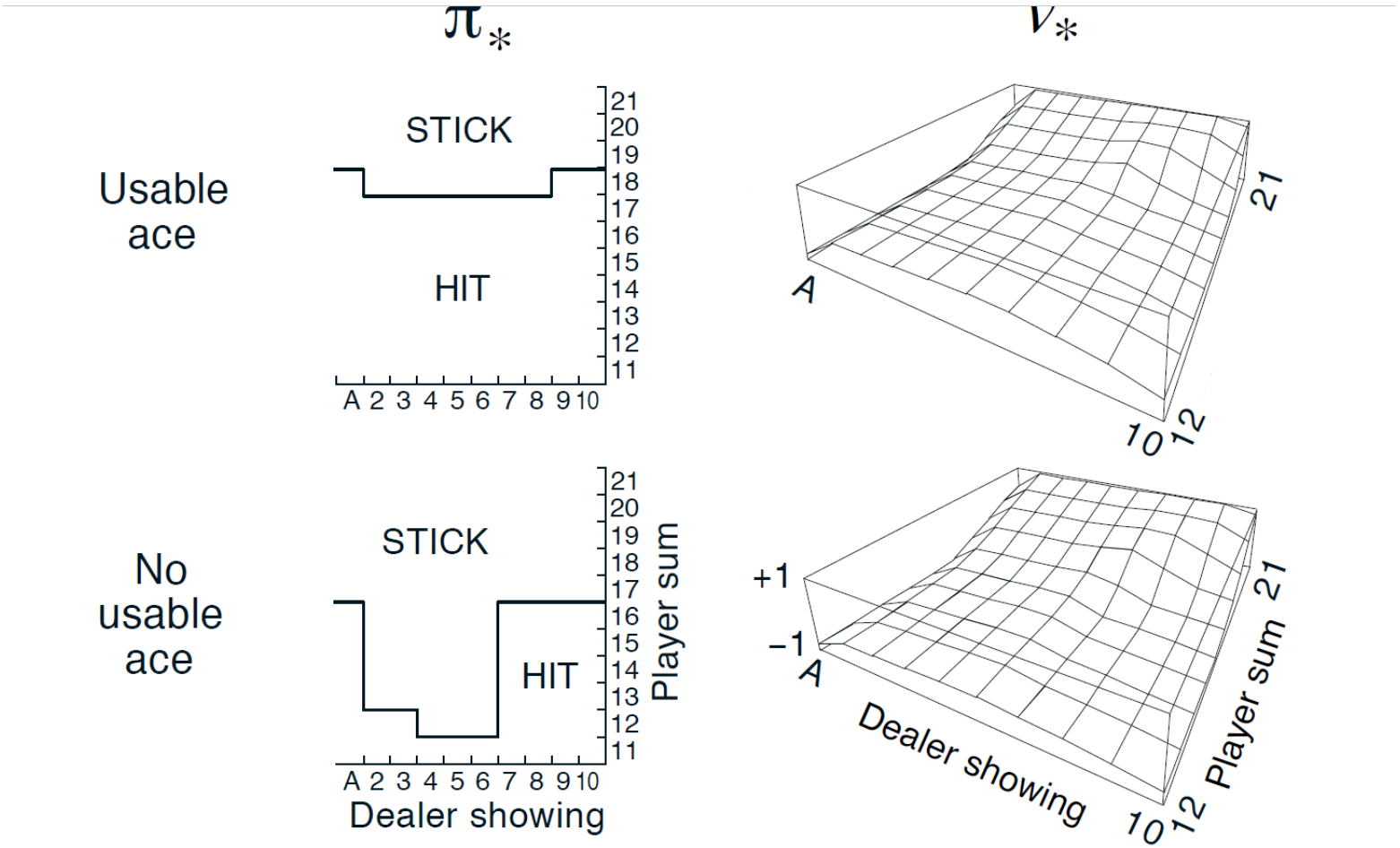


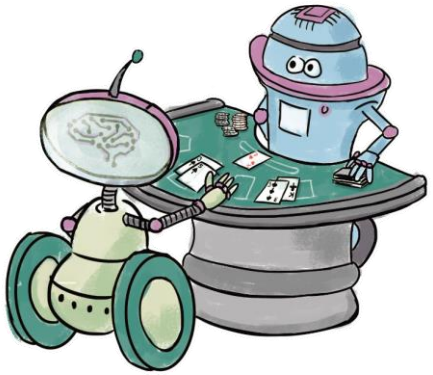


Ace = 1 or 11

Monte Carlo for Control

Example: Monte Carlo Control for Blackjack





Monte Carlo for Control

Policy Improvement: Guarantee of convergence

Policy Improvement theorem

- Let π and π' be any pair of deterministic policies such that

$$q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s)), \text{ for all } s \in \mathcal{S},$$

New Value — Old value

- For any action-value function q, the corresponding greedy policy is the one that, for each $s \in \mathcal{S}$, deterministically chooses an action with maximal action-value

$$\pi_{k+1}(s) \doteq \operatorname*{argmax}_a q_{\pi_k(s,a)}$$

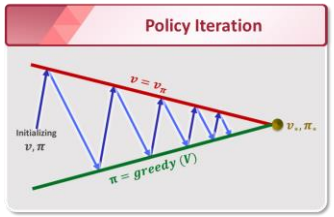
$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \operatorname*{argmax}_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s) \end{aligned}$$

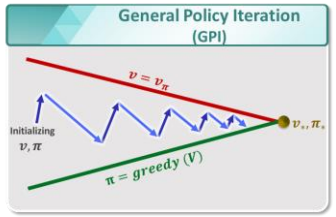
Monte Carlo for Control

Convergence Assumptions (1)

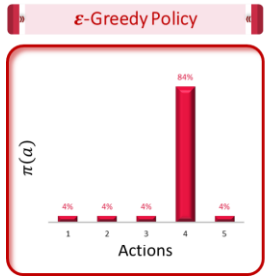
- **The policy improvement shown in the previous slide was based on two fundamental assumptions:**

1. **Infinite number of episodes:** $\lim_{k\to\infty} Q_{\pi_k}(s,a) = q_{\pi_k}(s,a)$





2. **Episodes have exploring starts**



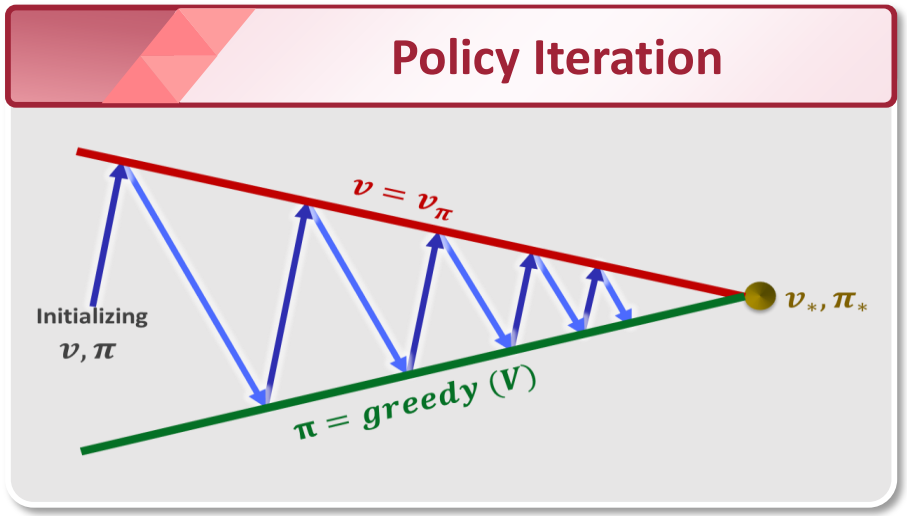
Monte Carlo for Control

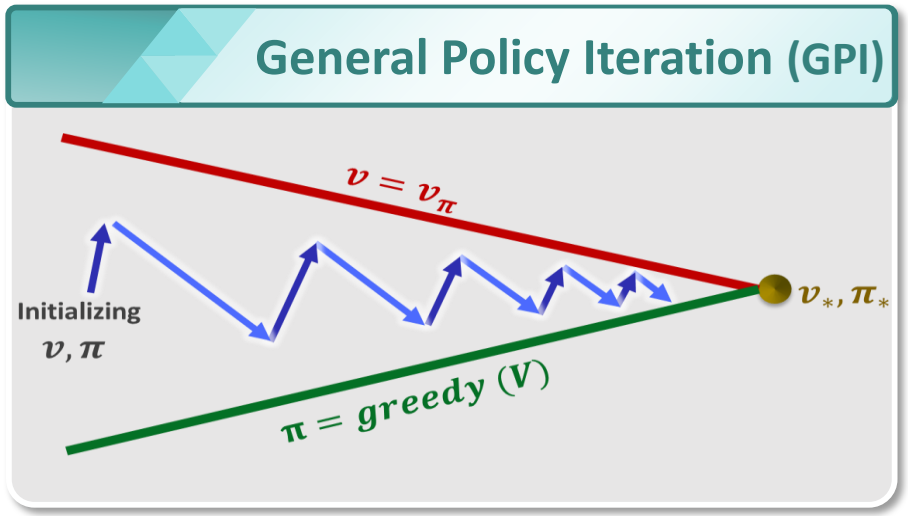
Convergence Assumptions (2): General Policy Iteration (GPI)

1 **Infinite number of episodes:** $\lim_{k\to\infty} Q_{\pi_k}(s,a) = q_{\pi_k}(s,a)$

- **First Approach:** Approximating q_{π_k}
  - Measurements and assumptions are made to obtain bounds on the magnitude and probability of error in the estimates.

- **Second Approach:** General Policy Iteration (GPI)





Monte Carlo for Control

Convergence Assumptions (3): Maintaining exploration (revisited)

② Episodes have exploring starts

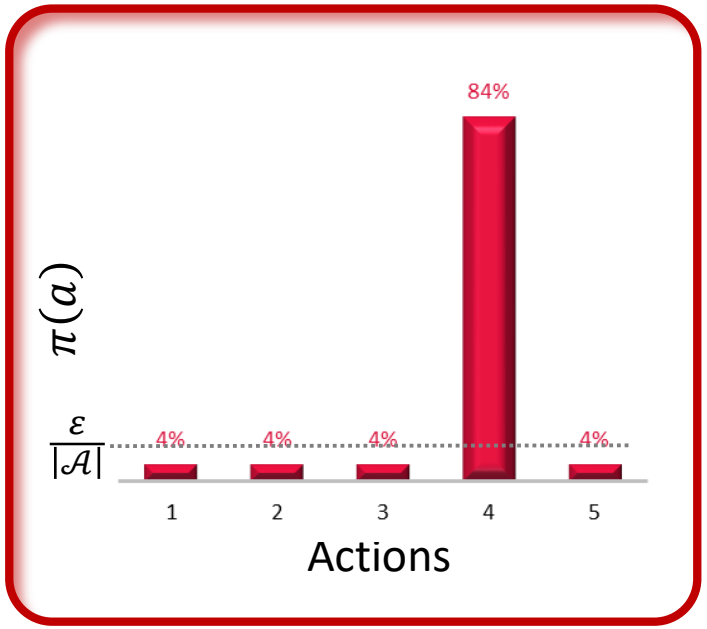
- **In Exploring starts exploring, all the state action pairs have non-zero probability of being the starting pair**
  - **Problem:** what if we have always a single start point for the environment, i.e in Chess game
    - In this case, exploring starts is does not maintain the exploration.
  - **Solution:** Simplest idea for ensuring continual exploration all actions are tried with non-zero probability
    - $1-\varepsilon$: choose the action which maximizes the action value function and with probability
    - ε: choose an action at random

$$\pi(a|s) = \begin{cases} \dfrac{\varepsilon}{|A(s)|} & \text{for the non} - \text{greedy action} \\ 1 - \varepsilon + \dfrac{\varepsilon}{|A(s)|} & \text{for the greedy action} \end{cases}$$

Monte Carlo for Control

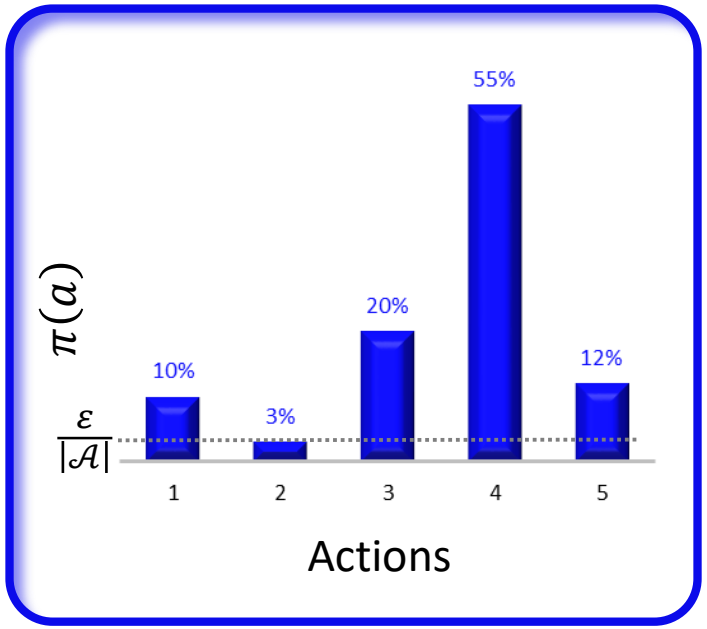
ε-soft policies

ε-Greedy Policy



$\subset$

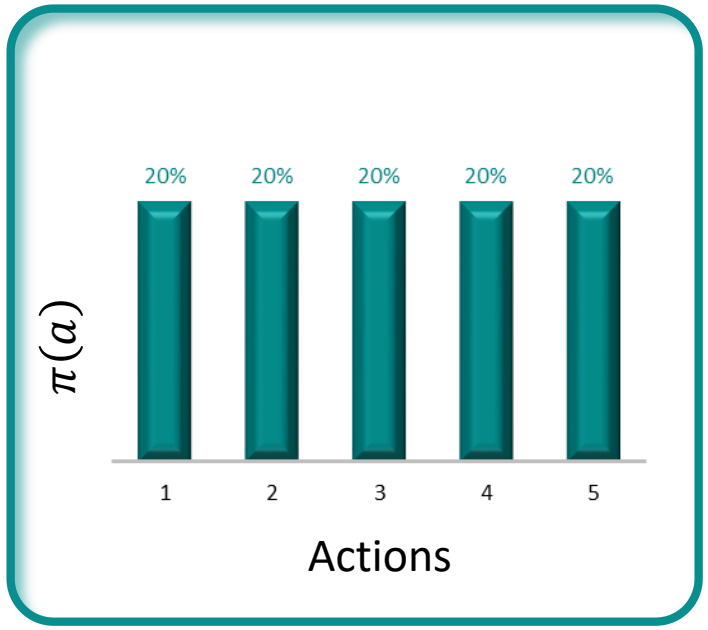
ε-Soft Policy



Monte Carlo for Control

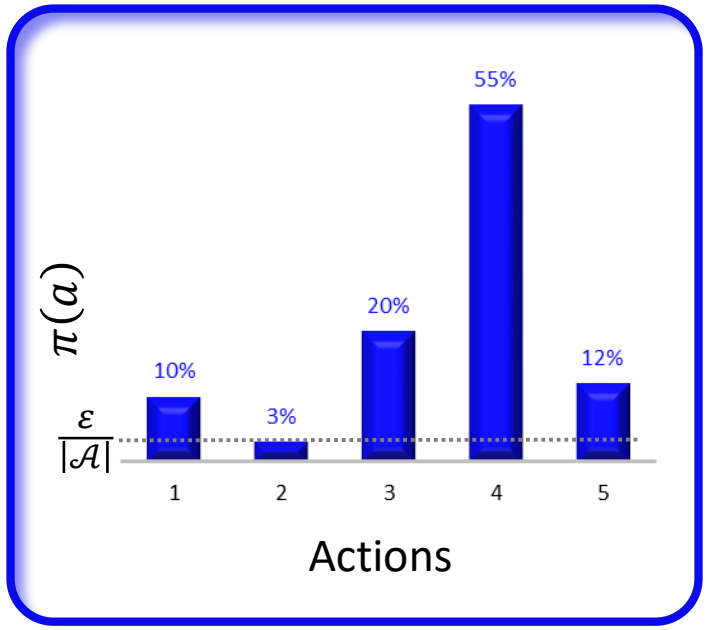
ε-soft policies

Uniform Policy



$\subset$

ε-Soft Policy

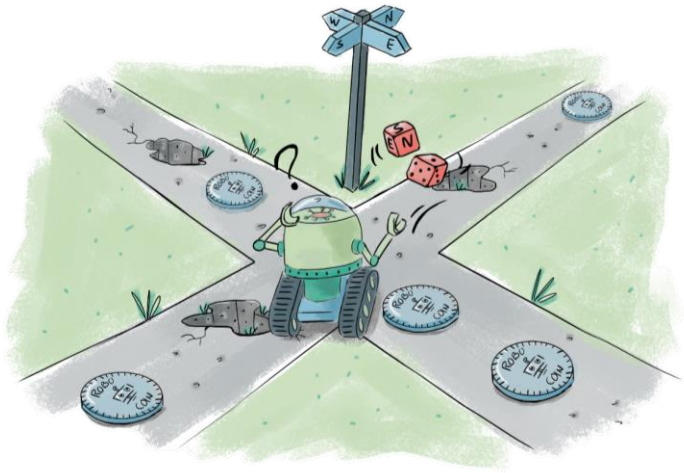


Monte Carlo Control

Example: The Small Gridworld Example (control)

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

States



ExceLab 5-2

	-10	-10	-10
-1	-1	-1	-1
-10	-10	-10	-1
-1	-1	-1	-1

Reward

Monte Carlo Control

ε-soft policies

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

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 $Returns(S_t, A_t)$

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

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

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

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

No- Exploring Start

Policy Evaluation

ϵ-Greedy Policy

Some other Strategies

1: softmax

- Selects actions using a Boltzmann distribution:

$$\pi(a|s) = Pr\{A_t = a|S_t = s\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}}$$

  - τ is a positive parameter called temperatur

2: ε-first or ε-decreasing

- Given $m \in \{1, \dots, {}^{T}/_{K}\}$
  - Draw each arm m times
  - Compute the empirical best arm

$$\hat{a} = argmax_a \, \hat{\mu}_a(K_m)$$

  - Keep playing this arm until round T

$$A_{t+1} = \hat{a} \text{ for } t \geq Km$$

  - Explore-then-Commit [O. Caelen and G. Bontempi, 2007]

3: ε-PCSgreedy

- PCS: Probability of Correct Selection
  - Optimal exploration on the basis of PCS
  - The idea is that an effective exploration step should lead to the largest increase of the probability PCS of correctly selecting the best arm.
    - [J. Vermorel and M. Mehryar, 2005]

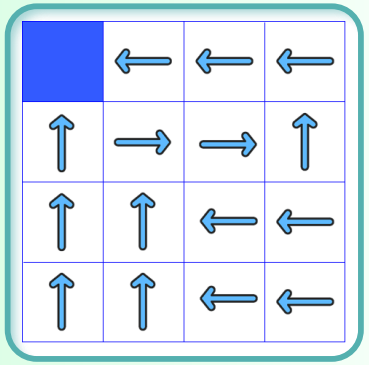
4: ε -greedy VDBE-Boltzmann

- Value-Difference Based Exploration (VDBE)
- Extends the ε-greedy method by controlling a state-dependent exploration probability
- $\varepsilon(s)$, in dependence of the value-function error instead of manual tuning
  - [M. Tokic,, 2010]

Monte Carlo Control

Discussion: ε-soft policies vs Exploring Starts:

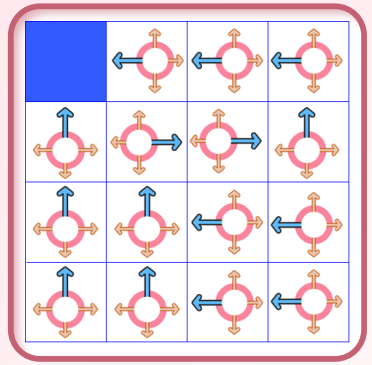
Optimal Policy π_*



- **Exploring starts can be used to find the optimal policy**
- **The idea of exploring starts is not terribly efficient**
  - it does not explore very well the space of possible episodes
  - some start states may rarely be encountered in practice

$>$

Optimal ε-soft Policy



- **A more effective way to ensure exploration**
- **It's impossible to converge to a deterministic optimal policy**
  - The policy always gives at least Epsilon probability to each action
  - Soft policies can only be used to find the optimal Epsilon soft policy (no determinist Optimal Policy)

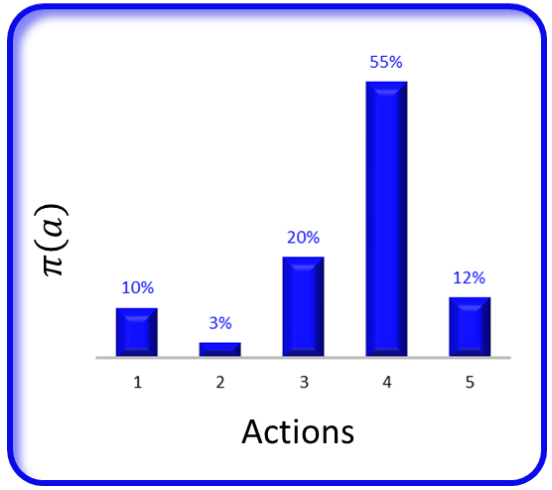
Off-policy Prediction

Off-policy Prediction

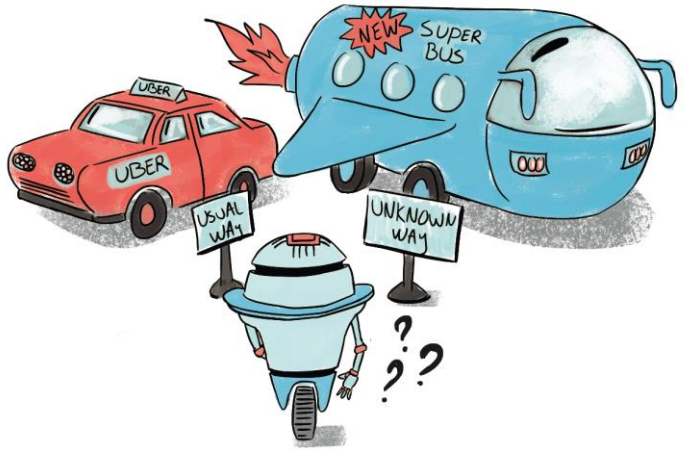
Why does off-policy learning matter?

- **So far we were using (almost) the same policy to (i) select actions, and (ii) estimate the value function**

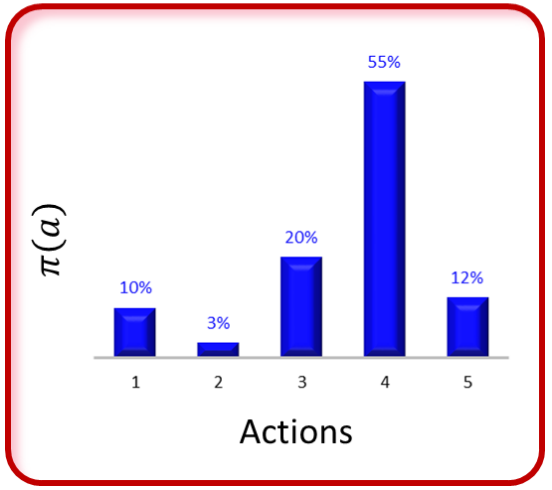
Action Selection



Behavior Policy



Value Estimation



Target Policy

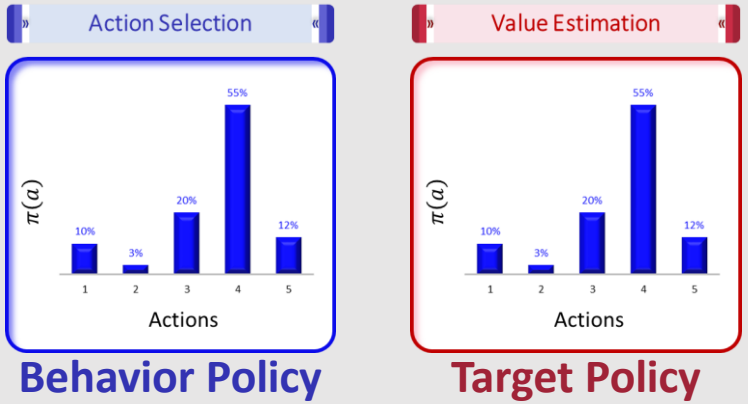
- **Problem:** What happens to the Exploration?

Off-policy Prediction

On-Policy vs Off-Policy

On-Policy Approach

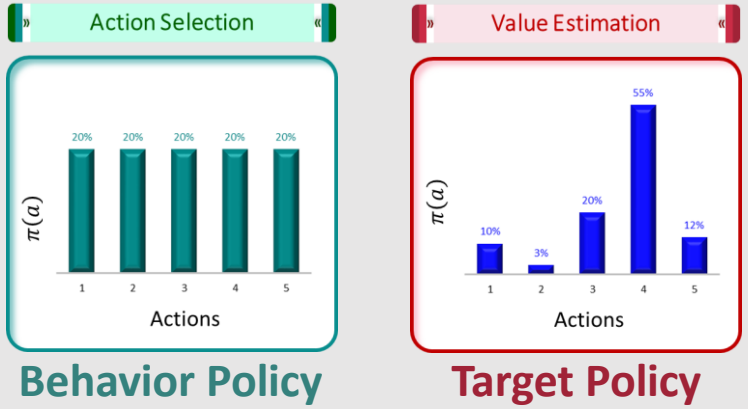
- **Estimate the value of data-generating policy**
  - Learns and acts based on the same policy



  - It learns action values not for the optimal policy, but for a near-optimal policy that still explores

Off-Policy Approach

- **Estimate the value of another policy**
  - Learns about the optimal policy while behaving according to an exploratory policy



  - In this case we say that learning is from data "off" the target policy

Off-policy Prediction

Off-Policy Approach

- **Performance**
  - Since the data is due to a different policy, off-policy methods are often of greater variance and are slower to converge

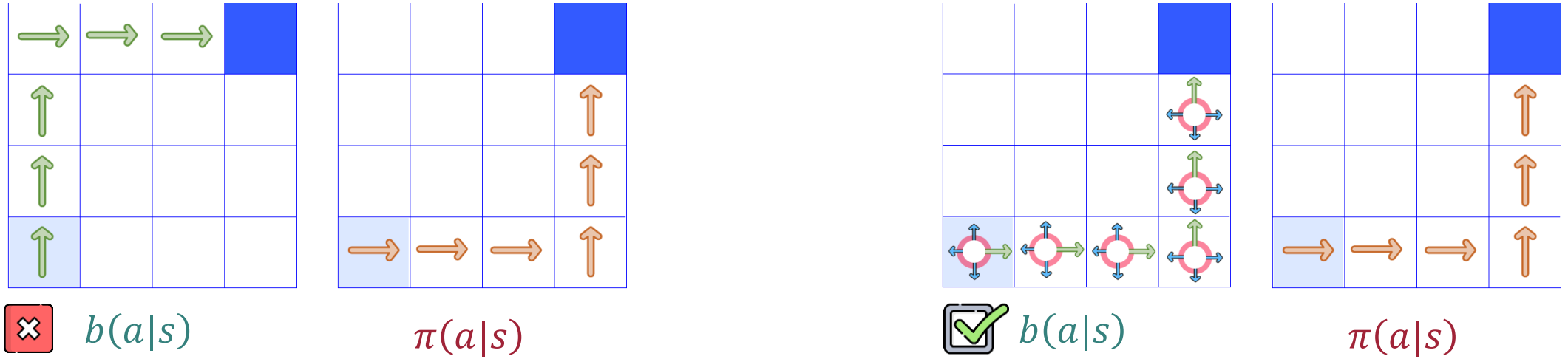
- **Why is it important?**
  - Off-policy methods are more powerful and general
    - Learn about optimal policy while following exploratory policy
    - They include on-policy methods as the special case in which the target and behavior policies are the same.
  - Learn from observing humans or other agents
    - Re-use experience generated from old policies
    - They can often be applied to learn from data generated by a conventional non-learning controller, or from a human expert

Off-policy Prediction

Assumption of Coverage

- **Assumption of Coverage:**
  - In order to use episodes from b to estimate values for π, we require that every action taken under π is also taken, at least occasionally, under b.

$$\pi(a|s) > 0 \text{ implies } b(a|s) > 0.$$



  - In general, we only require coverage, i.e., that b generates behavior that covers, or includes, π

- **In off-Policy Approach:**
  - We Sample based on the Behavior Policy

$$\boldsymbol{Sample}: x \sim b$$

  - But we Estimate based on Target Policy

$$\boldsymbol{Estimate}: \boldsymbol{E}_{\pi}[X]$$

- **How we can estimate values in Off-Policy Approach?**

$$\mathbb{E}_{b}[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma V}(\boldsymbol{S_{t+1}})] = \boldsymbol{V}_{b}(\boldsymbol{S_t}) \neq \boldsymbol{V}_{\pi}(\boldsymbol{S_t})$$

Off-policy Prediction

Derivation of Importance Sampling (1)

- **Importance Sampling (IS) helps to remove this pathology**
  - It is a general technique for estimating expected values under one distribution given samples from another.
  - Importance sampling is typically presented as a method for reducing the variance of the estimate of an expectation by carefully choosing a sampling distribution

- **IS application in Off-Policy**

$$\boldsymbol{E}_{\pi}[X] \doteq \sum_{x \in X} x\pi(x)$$

$$= \sum_{x \in X} x\pi(x)\frac{b(x)}{b(x)}$$

$$= \sum_{x \in X} x\frac{\pi(x)}{b(x)}b(x)$$

$$\rho(x) = \frac{\pi(x)}{b(x)}$$

importance-sampling ratio

$$\boldsymbol{E}_{\pi}[X] = \sum_{x \in X} x\rho(x)\, b(x)$$

$$= \boldsymbol{E}_{b}[X\rho(X)]$$

Consider $x\rho(x)$ as a Random Variable

Off-policy Prediction

Derivation of Importance Sampling (2)

$$\boldsymbol{E}_{\pi}[X] = \sum_{x \in X} x\rho(x)\, b(x)$$

$$\boldsymbol{E}[X] \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

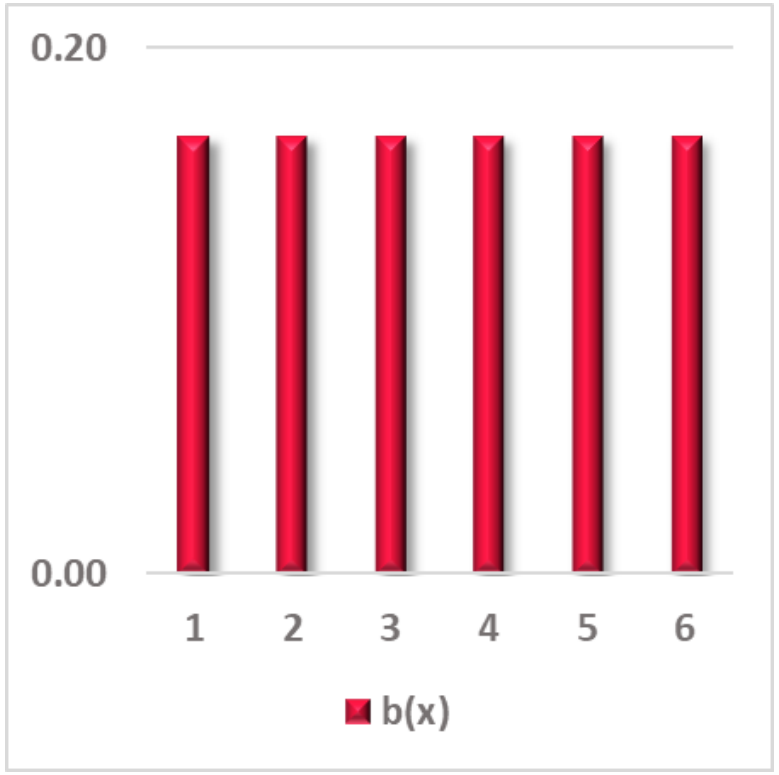
$$\boldsymbol{E}_{\pi}[X] \approx \frac{1}{n}\sum_{i=1}^{n} x_i\rho(x_i)$$

Where $x_i \sim b$

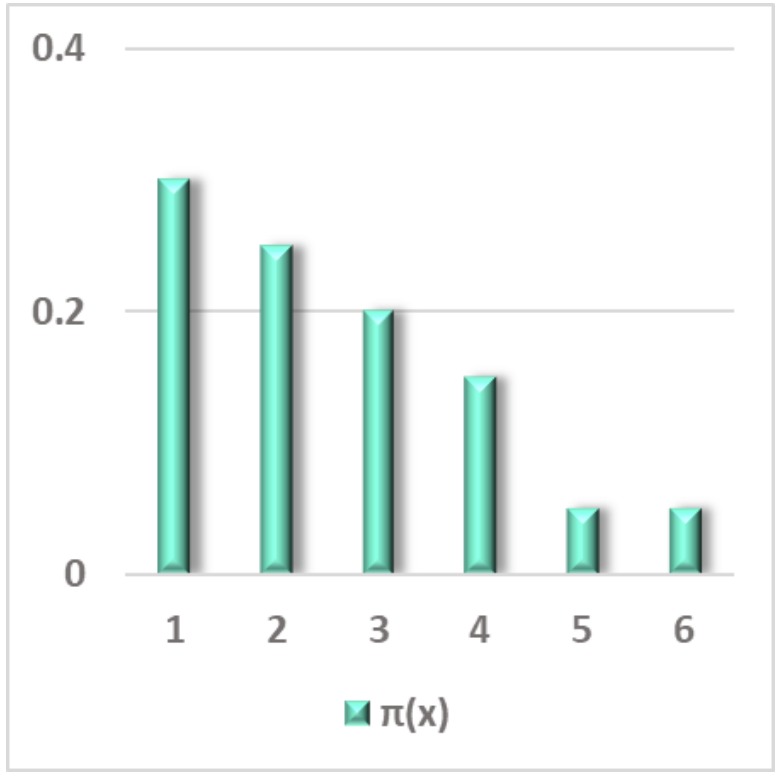
Sample from b
Estimate based on π

Off-policy Prediction

Importance Sampling: Die Roll Example



x	1	2	3	4	5	6	→	Mu
b(x)	1/6	1/6	1/6	1/6	1/6	1/6		3.5



x	1	2	3	4	5	6	→	Mu
π(x)	0.3	0.25	0.2	0.15	0.05	0.05		2.55

ExceLab (5-c)

Off-policy Monte Carlo Prediction

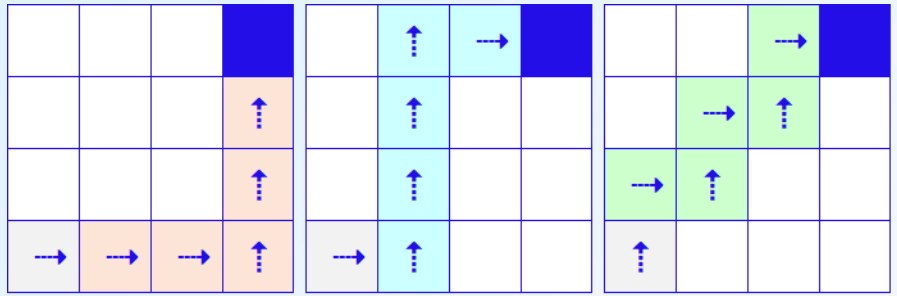
Off-policy Monte Carlo Prediction

On-Policy vs Off-Policy

On-Policy Approach

- **Estimate the value of data-generating policy**

$Sample: x \sim \pi$, and $Estimate: \boldsymbol{E}_{\pi}[X]$

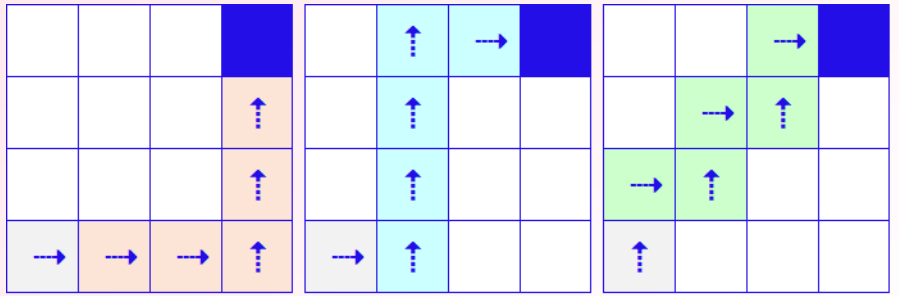


$$\boldsymbol{V}_{\pi}(\boldsymbol{s}) \approx \boldsymbol{average}(\boldsymbol{Returns}[0], \boldsymbol{Returns}[1], \boldsymbol{Returns}[2])$$

Off-Policy Approach

- **Estimate the value of another policy**

$Sample: x \sim b$, and $Estimate: \boldsymbol{E}_{\pi}[X]$



$$\boldsymbol{V}_{\pi}(\boldsymbol{s}) \approx average(\rho_0 Returns[0], \rho_1 Returns[1], \rho_2 Returns[2])$$

$$\rho = \frac{\mathbb{P}(\text{trajectory under } \pi)}{\mathbb{P}(\text{trajectory under } b)}$$

Off-policy Monte Carlo Prediction

Two Approaches: OIS vs WIS

- **Recall:** The goal in RL is to estimate the expected returns (values) under the target policy

$$V_\pi(s) \doteq \boldsymbol{E_\pi}[G_t \mid S_t = s]$$
$$V_\pi(s) \doteq \boldsymbol{E_b}[\rho_{t:T-1} G_t \mid S_t = s]$$

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

$$\boldsymbol{E}_\pi[X] \approx \frac{1}{n}\sum_{i=1}^{n} x_i \rho(x_i)$$

Where $x_i \sim b$

Ordinary Importance Sampling (OIS)

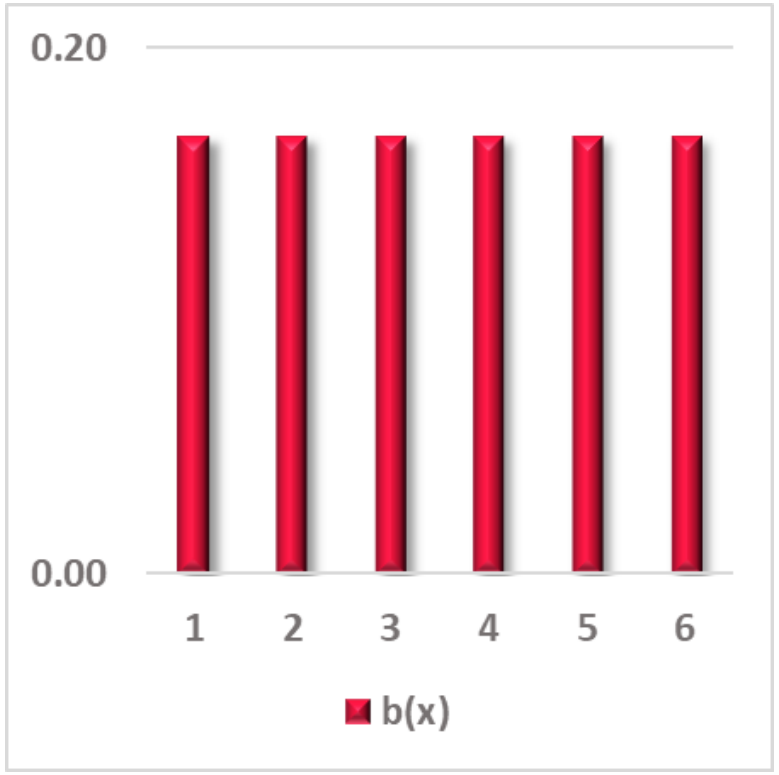
$$\boldsymbol{V(s)} \doteq \frac{\sum_{\boldsymbol{t \in \mathcal{T}(s)}} \rho_{t:T-1} G_t}{|\boldsymbol{\mathcal{T}(s)}|}$$

weighted Importance Sampling (WIS)

$$\boldsymbol{V(s)} \doteq \frac{\sum_{\boldsymbol{t \in \mathcal{T}(s)}} \rho_{t:T-1} G_t}{\sum_{\boldsymbol{t \in \mathcal{T}(s)}} \rho_{t:T-1}}$$

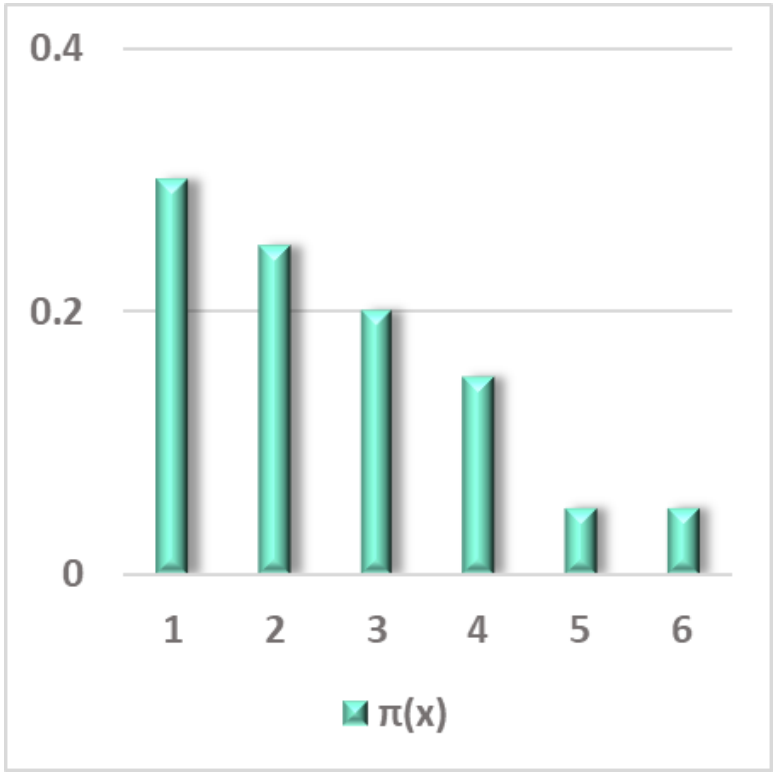
Off-policy Monte Carlo Prediction

Importance Sampling: Die Roll Example



x	1	2	3	4	5	6	→	Mu
b(x)	1/6	1/6	1/6	1/6	1/6	1/6		3.5

ExceLab (5-c)



x	1	2	3	4	5	6	→	Mu
π(x)	0.3	0.25	0.2	0.15	0.05	0.05		2.55

Off-policy Monte Carlo Prediction

Compare OIS vs WIS

Ordinary Importance Sampling (OIS)

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T-1} G_t}{|\mathcal{T}(s)|}$$

- **First-visit methods:**
  - **Unbiased Estimation**
  - **Unbounded Variance:**
    - the variance of OIS is in general unbounded because the variance of the ratios can be unbounded
    - The estimation can be extreme

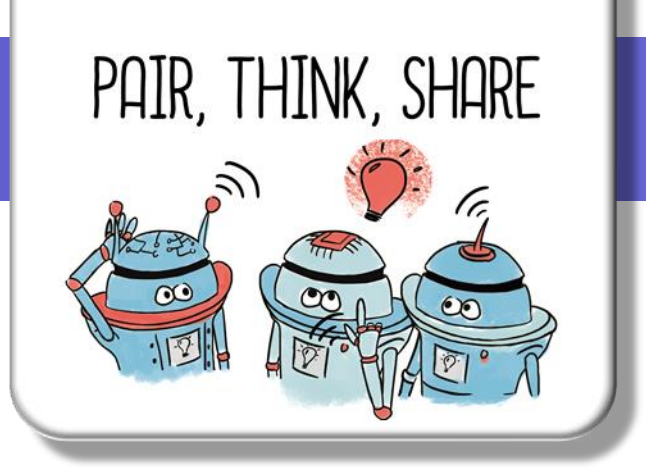
weighted Importance Sampling (WIS)

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T-1}}$$

- **First-visit methods:**
  - **Biased Estimation** (bias falls asymptotically to zero)
    - ratio $\rho\_(t{:}T{-}1)$ for the single return cancels in the numerator and denominator
  - **Bounded Variance**
    - largest weight on any single return is one
    - variance of the WIS estimator converges to zero

- **Every-visit methods**
  - **Both OIS and WIS are Biased Estimation**
    - bias falls asymptotically to zero as the number of samples increases
  - **In practice, every-visit methods are often preferred**
    - **(1)** no need to keep track of states' visit, **(2)** they are much easier to extend to approximations.

- **In practice, the weighted estimator usually has dramatically lower variance and is strongly preferred.**

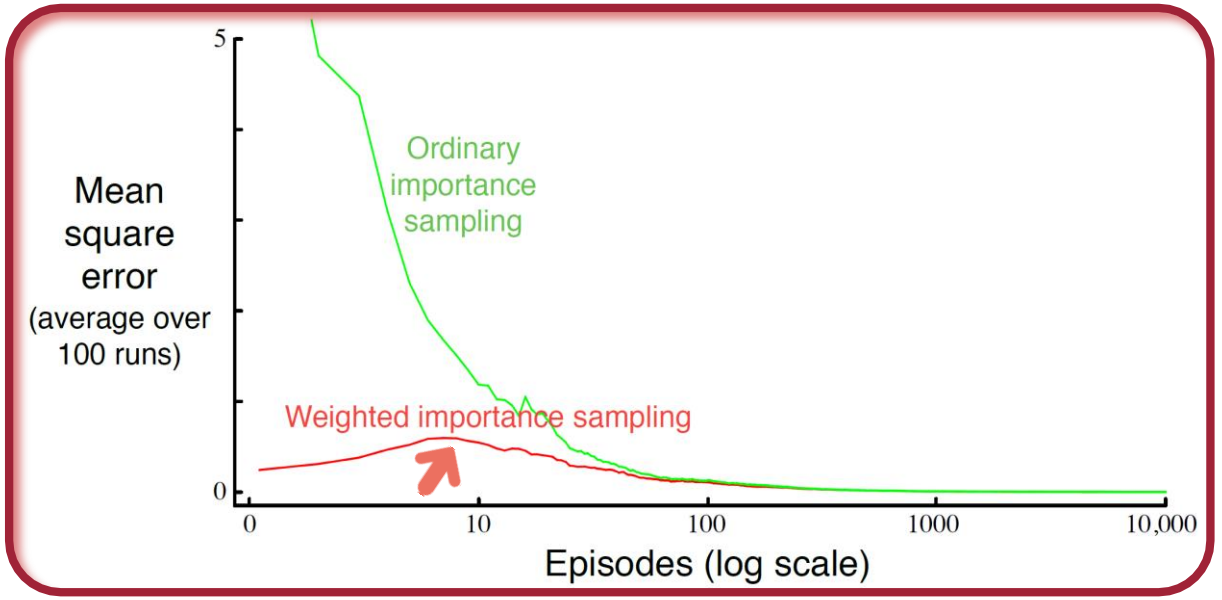


Off-Policy Estimation: Blackjack State Value

- The figure below shows the estimation of a single blackjack state value using off-policy data
- **State:** (Usable Ace, Player 13, Dealer 2)

Estimation:

- **Behavior Policy:** Random Hit or Stick
- **Target Policy:** stick only on a sum of 20 or 21
- **State Value:** −0.27726
- **Settings:** 1000 episodes, 100 ind. runs

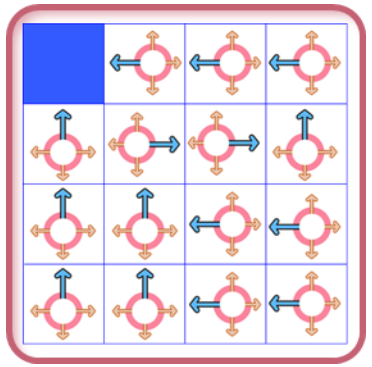


a **Which method performs better? OIS or WIS**

b **For WIS, why error first increased and then decreased?**

Off-policy Monte Carlo Prediction

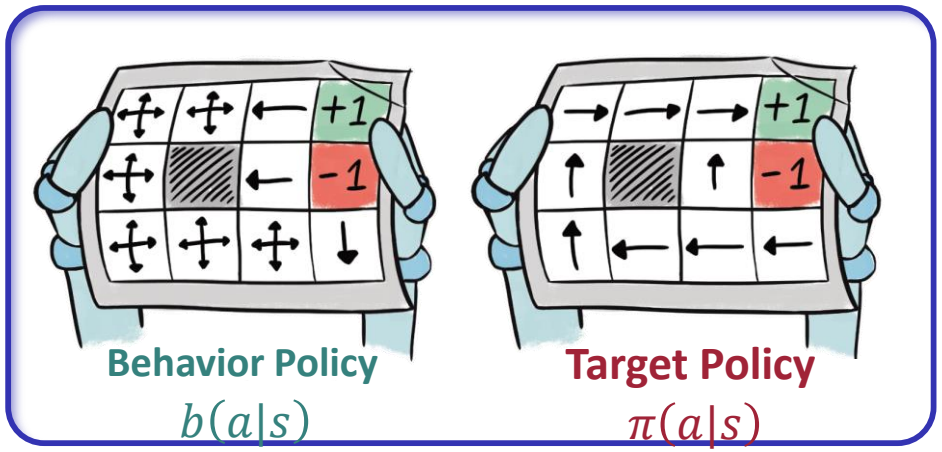
Why Off-Policy



suboptimal

Epsilon' soft policies

- Help with the continual exploration; However,
- They are **suboptimal** for both acting and learning



Off-policy

- Decouples behavior from the target Policy
- Parallel learning and facilitating exploration
- learns an optimal policy while maintaining exploration
  - Behavior policy takes the advantage of exploration, while Target policy is determined based on a deterministic approach

Incremental Implementation

Incremental Implementation

- **Monte Carlo prediction methods can be implemented incrementally**

$$\boldsymbol{P}(trajectory\ under\ \pi) =$$

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

$$\boldsymbol{P}(trajectory\ under\ b) = \prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)$$

Incremental Implementation

- **Monte Carlo prediction methods can be implemented incrementally**

$$\rho_{t:T-1} \doteq \frac{\boldsymbol{P}(trajectory\ under\ \pi)}{\boldsymbol{P}(trajectory\ under\ b)}$$

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

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

Incremental Implementation

Computing $\rho_{t:T-1}$ incrementally

- **Monte Carlo prediction methods can be implemented incrementally**

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

$$= \rho_t \rho_{t+1} \rho_{t+2} \cdots \rho_{T-2} \rho_{T-1}$$

$$\boldsymbol{W}_1 \leftarrow \rho_{T-1}$$

$$\boldsymbol{W}_2 \leftarrow \rho_{T-1}\rho_{T-2}$$

$$\boldsymbol{W}_3 \leftarrow \rho_{T-1}\rho_{T-2}\rho_{T-3}$$

$$\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t \rho_t$$

Incremental Implementation

Pseudocode: Off-policy Monte Carlo Prediction

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Behavior Policy

New term for Action Values

$W_t \rho_t$

Incremental Implementation

Pseudocode: Off-policy Monte Carlo Control

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Behavior Policy

Incremental Value Updates

Greedy Target Policy

deterministic policy

Summary

- **Monte Carlo methods are sample-based methods.**
  - Sample and average returns for each state-action pair and average rewards for each action.
  - Monte Carlo methods sample and average returns for each state-action pair and average rewards for each action.
  - They can be used when the model is unavailable or hard to formulate

- **They are typically applied to episodic tasks.**
  - The policy is updated only at the end of an episode.

- **Maintains exploration**
  - All state-action pairs must be visited
    - Exploring Starts
    - Stochastic Policies

Summary

- **On-policy methods vs Off-policy methods**
  - **On-policy methods** which attempt to evaluate or improve the policy that is used to make decisions
  - **Off-policy methods** which evaluate or improve a policy different from that used to generate the data.
    - **Target policy:** the policy being learned (a greedy policy with respect to Q)
    - **Behavior policy:** the policy that generates behavior

https://dilbert.com/search_results?terms=Dilbert+randomness

Summary

Pros

- Can learn V and Q directly from interaction with environment (using episodes!)
- No need for full models (using episodes!)
- No need to learn about ALL states (using episodes!)
- Estimates for each state are independent of both *(i) the number of states* and *(ii) the value of state*

Cons

- MC only works for episodic (terminating) environments
- MC learns from complete episodes, so no bootstrapping
- MC must wait until the end of an episode before
- return is known

- **Upcoming Lecture:** Temporal-Difference
  - TD works in continuing (nonterminating) environments, and can learn online after every step
  - TD can learn from incomplete sequences

References

- [1] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
- [2] Scratchapixel, “Monte Carlo Methods in Practice,” Scratchapixel. [Online]. Available: https://www.scratchapixel.com//lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-in-practice. [Accessed: 09-Feb-2020].
- [3] N. Metropolis and S. Ulam, “The monte carlo method,” Journal of the American statistical association, vol. 44, no. 247, pp. 335–341, 1949.
- [4] J. P. Hanna, S. Niekum, and P. Stone, “Importance sampling policy evaluation with an estimated behavior policy,” arXiv preprint arXiv:1806.01347, 2018.
- [5] C. R. Shelton, “Importance sampling for reinforcement learning with multiple objectives,” 2001.