

# Monte Carlo Methods

## CS60077: Reinforcement Learning

Abir Das

IIT Kharagpur

Oct 05 and 06, 2020

# Agenda

- § Understand how to evaluate policies in model-free setting using Monte Carlo methods
- § Understand Monte Carlo methods in model-free setting for control of Reinforcement Learning problems

# Resources

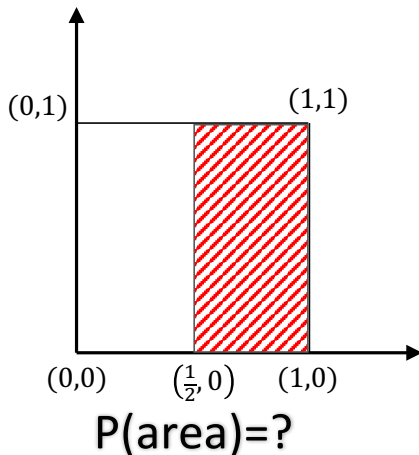
- § Reinforcement Learning by David Silver [[Link](#)]
- § Reinforcement Learning by Balaraman Ravindran [[Link](#)]
- § Monte Carlo Simulation by Nando de Freitas [[Link](#)]
- § SB: Chapter 5

# Model Free Setting

- § Like the previous few lectures, here also we will deal with prediction and control problems but this time it will be in a **model-free** setting
- § In model-free setting we do not have the full knowledge of the MDP
- § **Model-free prediction**: Estimate the value function of an unknown MDP
- § **Model-free control**: Optimise the value function of an unknown MDP
- § Model-free methods require only *experience* - sample sequences of states, actions, and rewards  $(S_1, A_1, R_2, \dots)$  from **actual** or **simulated** interaction with an environment.
- § Actual experience requires no knowledge of the environment's dynamics.
- § Simulated experience 'requires' models to generate samples only. No knowledge of the complete probability distributions of state transitions is required. In many cases this is easy to do.

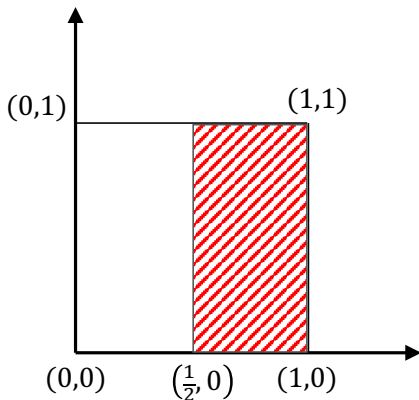
# Monte Carlo

- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



# Monte Carlo

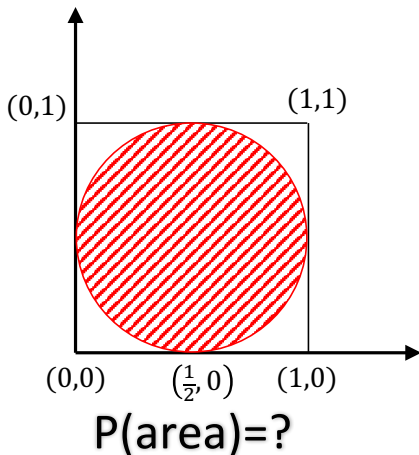
- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



$$P(\text{area}) = \frac{1}{2}$$

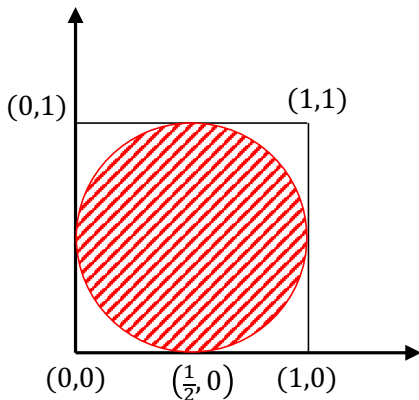
# Monte Carlo

- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



# Monte Carlo

- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

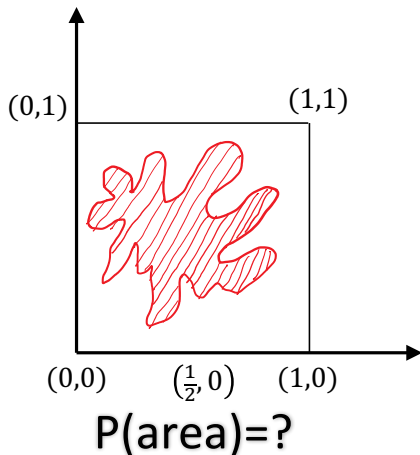


$$P(\text{area}) = \pi(1/2)^2$$



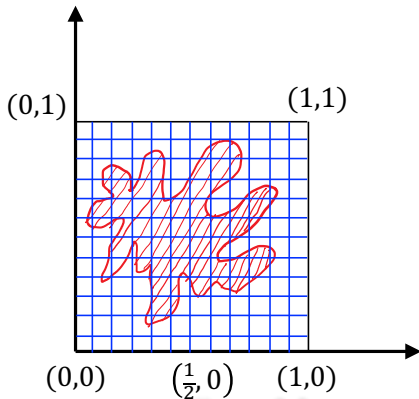
# Monte Carlo

- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



# Monte Carlo

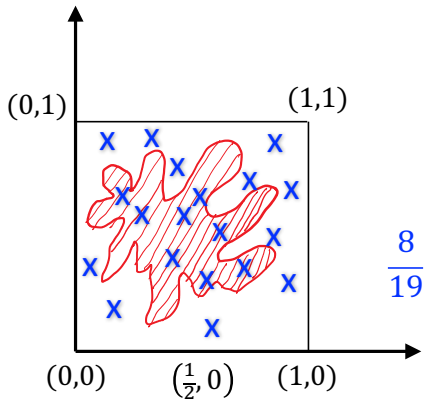
- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



$$P(\text{area}) = \frac{\# \text{ red boxes}}{\# \text{ blue boxes}}$$

# Monte Carlo

- § What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?



$$P(\text{area}) = \frac{\# \text{ darts in red area}}{\# \text{ darts}}$$

# History of Monte Carlo

## § The bomb and ENIAC



Image taken from: [www.livescience.com](http://www.livescience.com)

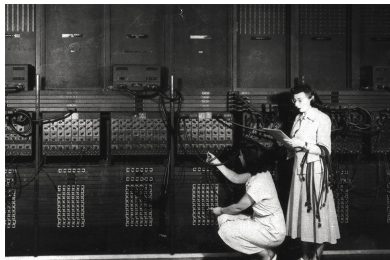


Image taken from: [www.digitaltrends.com](http://www.digitaltrends.com)

# Monte Carlo for Expectation Calculation

§ Lets say we want to compute  $\mathbb{E}[f(x)] = \int f(x)p(x)dx$

§ Draw i.i.d. samples  $\{x^{(i)}\}_{i=1}^N$  from the probability density  $p(x)$

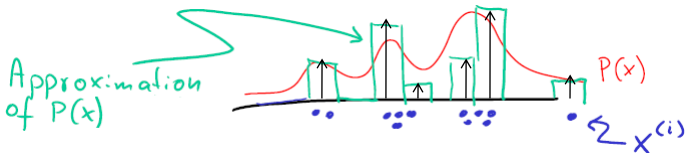


Image taken from: Nando de Freitas: MLSS 08

§ Approximate  $p(x) \approx \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(x)$  [ $\delta_{x^{(i)}}(x)$  is impulse at  $x^{(i)}$  on  $x$  axis]

§  $\mathbb{E}[f(x)] = \int f(x)p(x)dx \approx \int f(x) \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(x) dx =$

$$\frac{1}{N} \sum_{i=1}^N \underbrace{\int f(x) \delta_{x^{(i)}}(x) dx}_{f(x^{(i)})} = \frac{1}{N} \sum_{i=1}^N f(x^{(i)})$$

# Monte Carlo Policy Evaluation

§ Learn  $v_\pi$  from episodes of experience under policy  $\pi$

$$S_1, A_1, R_2, S_2, A_2, R_3, \dots, S_k, A_k, R_k \sim \pi$$

§ Recall that the *return* is the total discounted reward:

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

§ Recall that the value function is the expected return:

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

§ Monte-Carlo policy evaluation uses empirical mean return instead of expected return

# First Visit Monte Carlo Policy Evaluation

- § To evaluate state  $s$  i.e. to learn  $v_\pi(s)$
- § The **first** time-step  $t$  that state  $s$  is visited in an episode,
- § Increment counter  $N(s) \leftarrow N(s) + 1$
- § Increment total return  $S(s) \leftarrow S(s) + G_t$
- § Value is estimated by mean return  $V(s) = S(s)/N(s)$
- § By law of large number,  $V(s) \rightarrow v_\pi(s)$  as  $N(s) \rightarrow \infty$

# Every Visit Monte Carlo Policy Evaluation

- § To evaluate state  $s$  i.e. to learn  $v_\pi(s)$
- § **Every** time-step  $t$  that state  $s$  is visited in an episode,
- § Increment counter  $N(s) \leftarrow N(s) + 1$
- § Increment total return  $S(s) \leftarrow S(s) + G_t$
- § Value is estimated by mean return  $V(s) = S(s)/N(s)$
- § By law of large number,  $V(s) \rightarrow v_\pi(s)$  as  $N(s) \rightarrow \infty$



# Blackjack Example

- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action **stick**: Stop receiving cards (and terminate)
- Action **twist**: Take another card (no replacement)
- Reward for **stick**:
  - +1 if sum of cards  $>$  sum of dealer cards
  - 0 if sum of cards = sum of dealer cards
  - -1 if sum of cards  $<$  sum of dealer cards
- Reward for **twist**:
  - -1 if sum of cards  $>$  21 (and terminate)
  - 0 otherwise
- Transitions: automatically **twist** if sum of cards  $<$  12

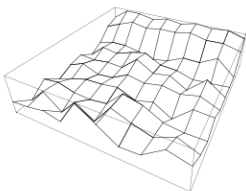


Slide courtesy: David Silver [Deepmind]

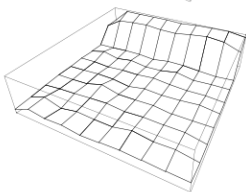
# Blackjack Example

After 10,000 episodes

Usable  
ace

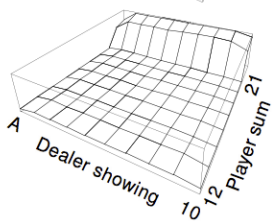
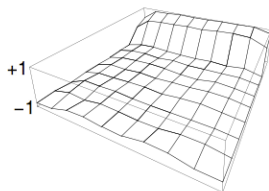


No  
usable  
ace



After 500,000 episodes

+1  
-1

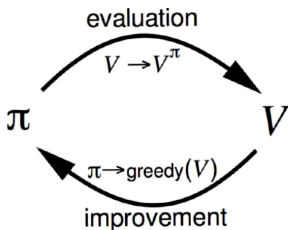


Policy: **stick** if sum of cards  $\geq 20$ , otherwise **twist**

Slide courtesy: David Silver [Deepmind]

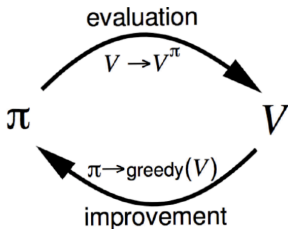
# Monte Carlo Control

- § We will now, see how Monte Carlo estimation can be used in *control*.
- § This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



# Monte Carlo Control

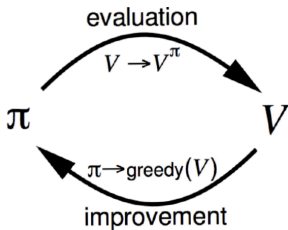
- § We will now, see how Monte Carlo estimation can be used in *control*.
- § This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- § Policy evaluation is done as Monte Carlo evaluation
- § Then, we can do greedy policy improvement.

# Monte Carlo Control

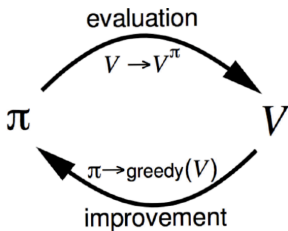
- § We will now, see how Monte Carlo estimation can be used in *control*.
- § This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- § Policy evaluation is done as Monte Carlo evaluation
- § Then, we can do greedy policy improvement.
- § **What is the problem!!**

# Monte Carlo Control

- § We will now, see how Monte Carlo estimation can be used in *control*.
- § This is mostly like the generalized policy iteration (GPI) where one maintains both an approximate policy and an approximate value function.



- § Policy evaluation is done as Monte Carlo evaluation
- § Then, we can do greedy policy improvement.
- § **What is the problem!!**

$$\pi'(s) \doteq \arg \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_\pi(s') \right\}$$

# Monte Carlo Control

§ Greedy policy improvement over  $v(s)$  requires model of MDP

$$\pi'(s) \doteq \arg \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_{\pi}(s') \right\}$$

# Monte Carlo Control

§ Greedy policy improvement over  $v(s)$  requires model of MDP

$$\pi'(s) \doteq \arg \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_{\pi}(s') \right\}$$

§ Greedy policy improvement over  $q(s, a)$  is model-free

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



# Monte Carlo Control

§ Greedy policy improvement over  $v(s)$  requires model of MDP

$$\pi'(s) \doteq \arg \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_{\pi}(s') \right\}$$

§ Greedy policy improvement over  $q(s, a)$  is model-free

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

§ How can we do Monte Carlo policy evaluation for  $q(s, a)$ ?

# Monte Carlo Control

§ Greedy policy improvement over  $v(s)$  requires model of MDP

$$\pi'(s) \doteq \arg \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_{\pi}(s') \right\}$$

§ Greedy policy improvement over  $q(s, a)$  is model-free

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

§ How can we do Monte Carlo policy evaluation for  $q(s, a)$ ?

§ Essentially the same as Monte Carlo evaluation for state values. Start at a state  $s$ , pick an action  $a$  and then follow the policy.

§ After few such episodes average the returns to get an estimate of  $q(s, a)$ .

# Monte Carlo Control

- § What are some concerns?
- § First visit/Every visit!!
- § Suppose you start at a state  $s$  and take action  $a$ . You reach at state  $s_1$  and then following the policy  $\pi$  at  $s$ , you take the action  $a_1 = \pi(s_1)$ . Can you take the rest of the trajectory as a sample to estimate  $q(s_1, a_1)$ ?
- § Practically you can, but convergence can not be guaranteed. The reason is that this strategy draws a disproportionately large number of actions corresponding to  $\pi$ . So, each sample is considered only for the starting  $s$  and  $a$ .

# Monte Carlo Control

- § What are some concerns?
- § First visit/Every visit!!
- § Suppose you start at a state  $s$  and take action  $a$ . You reach at state  $s_1$  and then following the policy  $\pi$  at  $s$ , you take the action  $a_1 = \pi(s_1)$ . Can you take the rest of the trajectory as a sample to estimate  $q(s_1, a_1)$ ?
- § Practically you can, but convergence can not be guaranteed. The reason is that this strategy draws a disproportionately large number of actions corresponding to  $\pi$ . So, each sample is considered only for the starting  $s$  and  $a$ .
- § How to make sure we have  $q(s, a)$  estimates for all  $s$  and  $a$ ? Especially because of the above the '*exploring starts*' becomes important.

# Monte Carlo Control

- § Many state-action pairs may never be visited.
- § For deterministic policy, with no returns to average, the Monte Carlo estimates of many actions will not improve with experience.
- § This is the general problem of maintaining exploration.
- § One way to do this is by specifying that the episodes start in a state-action pair, and that every pair has a nonzero probability of being selected as the start.
- § This assumption is called '*exploring starts*'

# Monte Carlo Control

- § Many state-action pairs may never be visited.
- § For deterministic policy, with no returns to average, the Monte Carlo estimates of many actions will not improve with experience.
- § This is the general problem of maintaining exploration.
- § One way to do this is by specifying that the episodes start in a state-action pair, and that every pair has a nonzero probability of being selected as the start.
- § This assumption is called '*exploring starts*'
- § Monte Carlo Exploration Starts is an 'on policy' method. On policy methods evaluate or improve the policy by drawing samples from the same policy.
- § Off-policy methods evaluate or improve a policy different from that used to generate the samples.

# Monte Carlo Control

- § Before going to off-policy methods let us look into an on policy Monte Carlo control method that does not use *exploring starts*.
- § The assumption of exploring starts is sometimes useful, but it cannot be relied upon in general, particularly when learning directly from actual interaction with an environment.
- § The easiest alternative is to consider stochastic policies with a nonzero probability of selecting all actions in each state.

# Monte Carlo Control

- § Before going to off-policy methods let us look into an on policy Monte Carlo control method that does not use *exploring starts*.
- § The assumption of exploring starts is sometimes useful, but it cannot be relied upon in general, particularly when learning directly from actual interaction with an environment.
- § The easiest alternative is to consider stochastic policies with a nonzero probability of selecting all actions in each state.
- § Instead of getting a greedy policy in the policy improvement step, an  $\epsilon$ -greedy policy is obtained.
- § It means most of the time, the action corresponding to maximum estimated action value is chosen, but sometimes (with probability  $\epsilon$ ) an action at random is chosen.
- § Probability of choosing nongreedy actions is  $\frac{\epsilon}{|\mathcal{A}(s)|}$  whereas remaining bulk of the probability,  $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$ , is given to the greedy action.



# Monte Carlo Control

- §  $\epsilon$ -greedy policy is an example of a bigger class of policies known as  $\epsilon$ -soft policies where  $\pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$  for all states and actions, for some  $\epsilon > 0$ .
- § Among  $\epsilon$ -soft policies,  $\epsilon$ -greedy policy is, in some sense, closest to greedy.
- § By using  $\epsilon$ -greedy policy improvement strategy, we achieve the best policy among  $\epsilon$ -soft policies, but we eliminate the assumption of 'exploring starts'.

# Off-policy Methods

- § All methods trying to learn control face a dilemma.
- ▶ They seek to learn action values conditional on subsequent optimal behavior.
  - ▶ But they need to behave non-optimally in order to explore all actions (to find the optimal actions).

# Off-policy Methods

- § All methods trying to learn control face a dilemma.
  - ▶ They seek to learn action values conditional on subsequent optimal behavior.
  - ▶ But they need to behave non-optimally in order to explore all actions (to find the optimal actions).
- § The on-policy approach is actually a compromise, it learns action values not for the optimal policy, but for a near-optimal policy that still explores.

# Off-policy Methods

- § All methods trying to learn control face a dilemma.
  - ▶ They seek to learn action values conditional on subsequent optimal behavior.
  - ▶ But they need to behave non-optimally in order to explore all actions (to find the optimal actions).
- § The on-policy approach is actually a compromise, it learns action values not for the optimal policy, but for a near-optimal policy that still explores.
- § Off-policy methods address this by using two policies for two different purposes.
  - ▶ one that is learned about and that becomes the optimal policy - **target policy**.
  - ▶ one that is more exploratory and is used to generate behavior - **behavior policy**.

# Off-policy Prediction

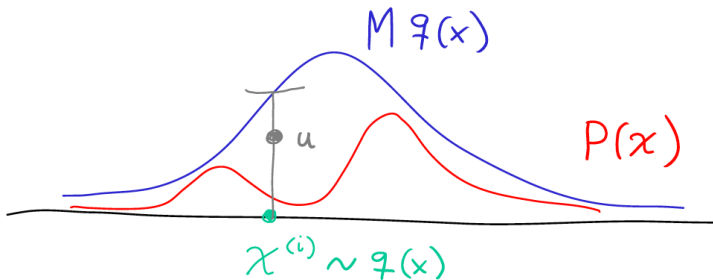
- § Estimate  $v_\pi$  or  $q_\pi$  of the target policy  $\pi$ , but we have episodes from another policy  $\mu$ , the behavior policy.
- § Almost all off-policy methods utilize concepts from sampling theory for such operations.

# Rejection Sampling

set  $i = 1$

Repeat until  $i = N$

- 1 Sample  $x^{(i)} \sim q(x)$  and  $u \sim \mathcal{U}_{(0,1)}$
- 2 If  $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$ , then accept  $x^{(i)}$  and increment counter  $i$  by 1.  
Otherwise, reject.



# Importance Sampling

§ What is bad about rejection sampling?

# Importance Sampling

- § What is bad about rejection sampling?
- § Many wasted samples! why?



# Importance Sampling

- § What is bad about rejection sampling?
- § Many wasted samples! why?
- § Importance sampling is a classical way to address this. You keep all the samples from the proposal/behavior distribution, you just weigh them.

# Importance Sampling

- § What is bad about rejection sampling?
- § Many wasted samples! **why?**
- § Importance sampling is a classical way to address this. You keep all the samples from the proposal/behavior distribution, you just weigh them.
- § Lets say we want to compute  $\mathbb{E}_{x \sim p(\cdot)}[f(x)] = \int f(x)p(x)dx$

$$\begin{aligned}\mathbb{E}_{x \sim p(\cdot)}[f(x)] &= \int f(x)p(x)dx = \int f(x) \frac{p(x)}{q(x)} q(x)dx \\ &= \mathbb{E}_{x \sim q(\cdot)} \left[ f(x) \frac{p(x)}{q(x)} \right] \\ &\approx \frac{1}{N} \sum_{x^{(i)} \sim q(\cdot), i=1}^N f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}\end{aligned}$$

- §  $\frac{p(x^{(i)})}{q(x^{(i)})}$  is called the *importance weight*.

# Normalized Importance Sampling

To avoid numerical instability, the denominator is changed in the following way

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

# MC Control with Importance Sampling

- § What are the samples  $x^{(i)}$ ? What are the  $p(\cdot)$  and  $q(\cdot)$  in our case? and what is  $f(x^{(i)})$ ?

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

# MC Control with Importance Sampling

- § What are the samples  $x^{(i)}$ ? What are the  $p(\cdot)$  and  $q(\cdot)$  in our case? and what is  $f(x^{(i)})$ ?

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

- §  $x^{(i)}$  are the trajectories.

# MC Control with Importance Sampling

- § What are the samples  $x^{(i)}$ ? What are the  $p(\cdot)$  and  $q(\cdot)$  in our case? and what is  $f(x^{(i)})$ ?

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

- §  $x^{(i)}$  are the trajectories.
- §  $p(x^{(i)})$  is the probability of the trajectory  $x^{(i)}$  given that the trajectory follows the target policy.

# MC Control with Importance Sampling

- § What are the samples  $x^{(i)}$ ? What are the  $p(\cdot)$  and  $q(\cdot)$  in our case? and what is  $f(x^{(i)})$ ?

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

- §  $x^{(i)}$  are the trajectories.
- §  $p(x^{(i)})$  is the probability of the trajectory  $x^{(i)}$  given that the trajectory follows the target policy.
- §  $q(x^{(i)})$  is the probability of the trajectory  $x^{(i)}$  given that the trajectory follows the behavior policy.

# MC Control with Importance Sampling

- § What are the samples  $x^{(i)}$ ? What are the  $p(\cdot)$  and  $q(\cdot)$  in our case? and what is  $f(x^{(i)})$ ?

$$\mathbb{E}_{x \sim p(\cdot)}[f(x)] \approx \frac{\sum_{x^{(i)} \sim q(\cdot)} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim q(\cdot)} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

- §  $x^{(i)}$  are the trajectories.
- §  $p(x^{(i)})$  is the probability of the trajectory  $x^{(i)}$  given that the trajectory follows the target policy.
- §  $q(x^{(i)})$  is the probability of the trajectory  $x^{(i)}$  given that the trajectory follows the behavior policy.
- §  $f(x^{(i)})$  is the return.



# MC Control with Importance Sampling

§ How is a trajectory represented?

# MC Control with Importance Sampling

- § How is a trajectory represented?
- § Refresher from the very first lecture.

- **Goal in RL Problem:**- to maximize the total reward “in expectation” over the long run.
- $\tau \stackrel{\text{def}}{=} (s_1, a_1, s_2, a_2, \dots), p(\tau) = p(s_1) \prod_t p(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- $\max \mathbb{E}_{\tau \sim p(\tau)} [\sum_t R(s_t, a_t)]$

# MC Control with Importance Sampling

- § How is a trajectory represented?
- § Refresher from the very first lecture.

- **Goal in RL Problem**:- to maximize the total reward “in expectation” over the long run.
- $\tau \stackrel{\text{def}}{=} (s_1, a_1, s_2, a_2, \dots), p(\tau) = p(s_1) \prod_t p(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- $\max \mathbb{E}_{\tau \sim p(\tau)} [\sum_t R(s_t, a_t)]$

- § Let some trajectory  $x^{(i)}$  be  $(s_1, a_1, s_2, a_2, \dots)$
- §  $p(x^{(i)}) = p(s_1) \pi(a_1 | s_1) p(s_2 | s_1, a_1) \pi(a_2 | s_2) p(s_3 | s_2, a_2) \dots$
- §  $q(x^{(i)}) = p(s_1) \mu(a_1 | s_1) p(s_2 | s_1, a_1) \mu(a_2 | s_2) p(s_3 | s_2, a_2) \dots$

# MC Control with Importance Sampling

- § How is a trajectory represented?
- § Refresher from the very first lecture.

- **Goal in RL Problem:**- to maximize the total reward “in expectation” over the long run.
- $\tau \stackrel{\text{def}}{=} (s_1, a_1, s_2, a_2, \dots), p(\tau) = p(s_1) \prod_t p(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- $\max \mathbb{E}_{\tau \sim p(\tau)} [\sum_t R(s_t, a_t)]$

§ Let some trajectory  $x^{(i)}$  be  $(s_1, a_1, s_2, a_2, \dots)$

§  $p(x^{(i)}) = p(s_1) \pi(a_1 | s_1) p(s_2 | s_1, a_1) \pi(a_2 | s_2) p(s_3 | s_2, a_2) \dots$

§  $q(x^{(i)}) = p(s_1) \mu(a_1 | s_1) p(s_2 | s_1, a_1) \mu(a_2 | s_2) p(s_3 | s_2, a_2) \dots$

§  $\frac{p(x^{(i)})}{q(x^{(i)})} = \frac{\cancel{p(s_1)} \pi(a_1 | s_1) \cancel{p(s_2 | s_1, a_1)} \pi(a_2 | s_2) \cancel{p(s_3 | s_2, a_2)} \dots}{\cancel{p(s_1)} \mu(a_1 | s_1) \cancel{p(s_2 | s_1, a_1)} \mu(a_2 | s_2) \cancel{p(s_3 | s_2, a_2)} \dots} = \frac{\pi(a_1 | s_1) \pi(a_2 | s_2) \dots}{\mu(a_1 | s_1) \mu(a_2 | s_2) \dots} =$

$$\prod_{t=1}^{T_i} \frac{\pi(a_t | s_t)}{\mu(a_t | s_t)}$$

# MC Control with Importance Sampling

$$\mathbb{E}_{x \sim \pi}[f(x)] \approx \frac{\sum_{x^{(i)} \sim \mu} f(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}}{\sum_{x^{(i)} \sim \mu} \frac{p(x^{(i)})}{q(x^{(i)})}}$$

$$\begin{aligned} v_{\pi}(s) &= \mathbb{E}[G | S_1 = s] \\ &\approx \frac{\sum_{i=1}^N G^{(i)} \prod_{t=1}^{T_i} \frac{\pi(a_t^{(i)} | s_t^{(i)})}{\mu(a_t^{(i)} | s_t^{(i)})}}{\sum_{i=1}^N \prod_{t=1}^{T_i} \frac{\pi(a_t^{(i)} | s_t^{(i)})}{\mu(a_t^{(i)} | s_t^{(i)})}} \end{aligned}$$

§ This was the evaluation step then do the greedy policy improvement.