

# 지능 시스템

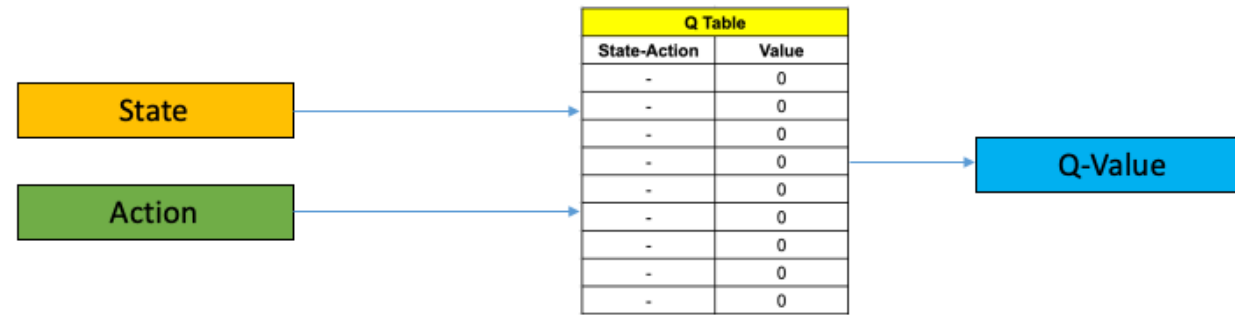
# Intelligent Systems

---

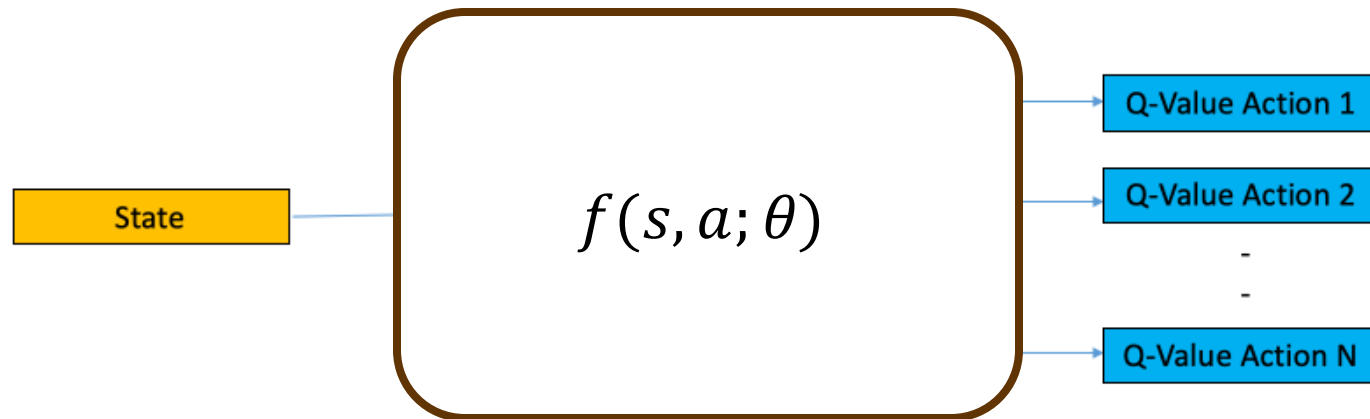
Lecture 8 – Policy Gradient

# From Last Lecture

---

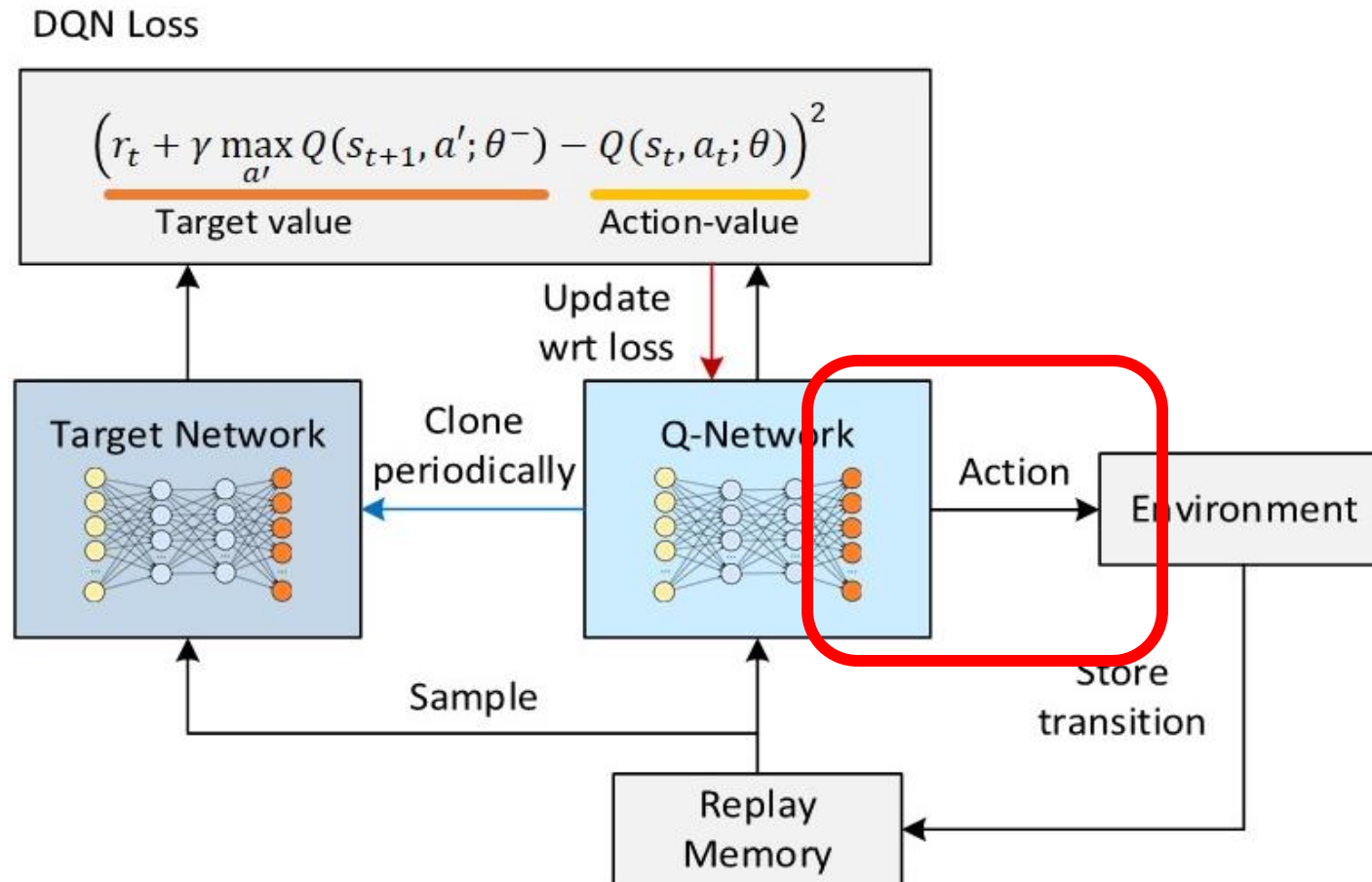


Q Learning



Deep Q Learning

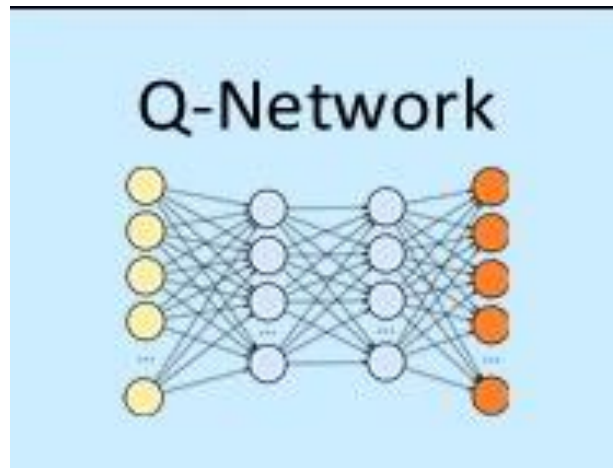
# From Last Lecture



# How is the action selected?

---

State,  $s$



$$Q(s, a_1)$$

$$Q(s, a_2)$$

$$Q(s, a_3)$$

$$Q(s, a_4)$$

$$Q(s, a_5)$$

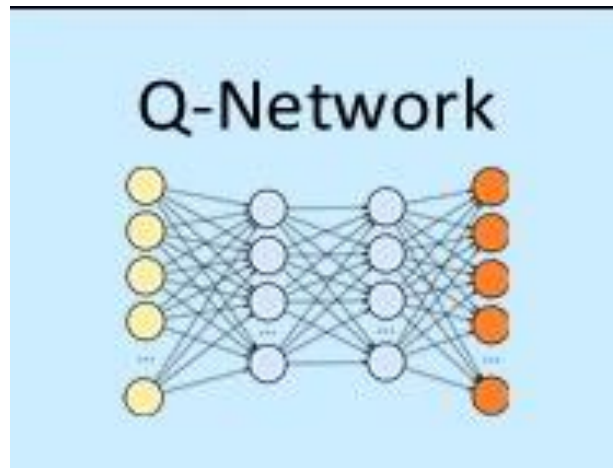
In DQN, the action is chosen based on the Q-values.

The action with the largest Q-value is chosen based on the epsilon-greedy algorithm

# How is the action selected?

---

State,  $s$



$$Q(s, a_1)$$

$$Q(s, a_2)$$

$$Q(s, a_3)$$

$$Q(s, a_4)$$

$$Q(s, a_5)$$

Condition?

- Action must be discrete
- The one best action exists (assume  $Q$  is not optimal)

# Example: Stochastic Policy

---

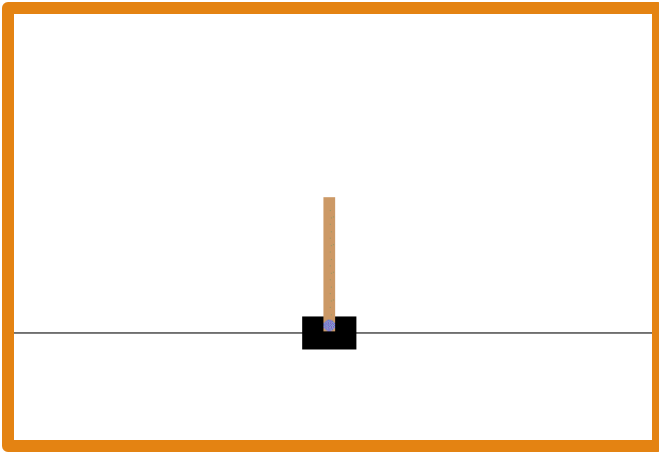
## Rock-Paper-Scissors

- No good one action, all actions are equally good
- A deterministic optimal policy does not exist.
- A uniform random policy is optimal



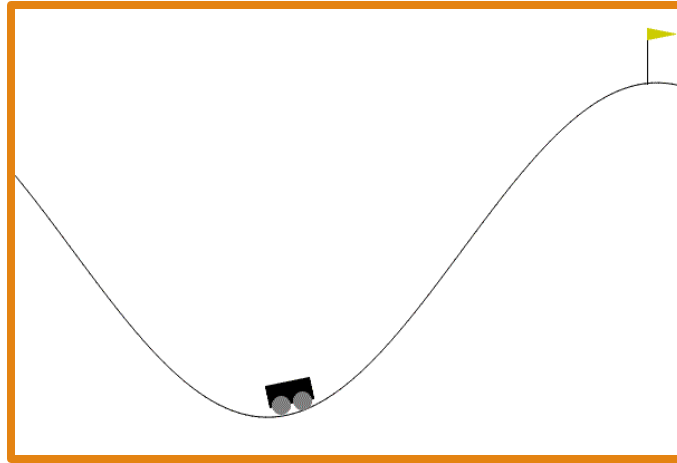
# Example: Discrete Action

---



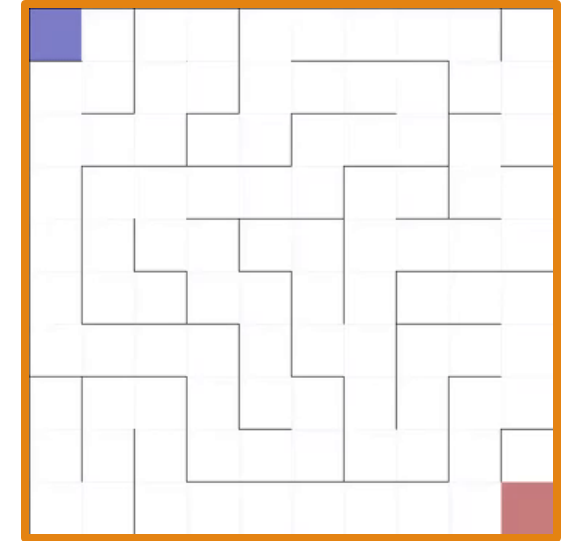
Cartpole:

- Move Left
- Move Right



Mountain Car:

- Move Left
- Move Right

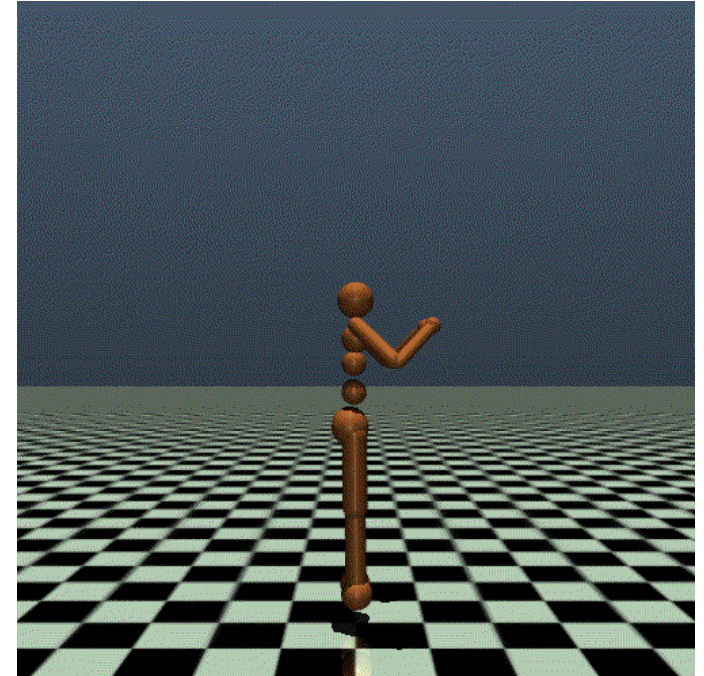
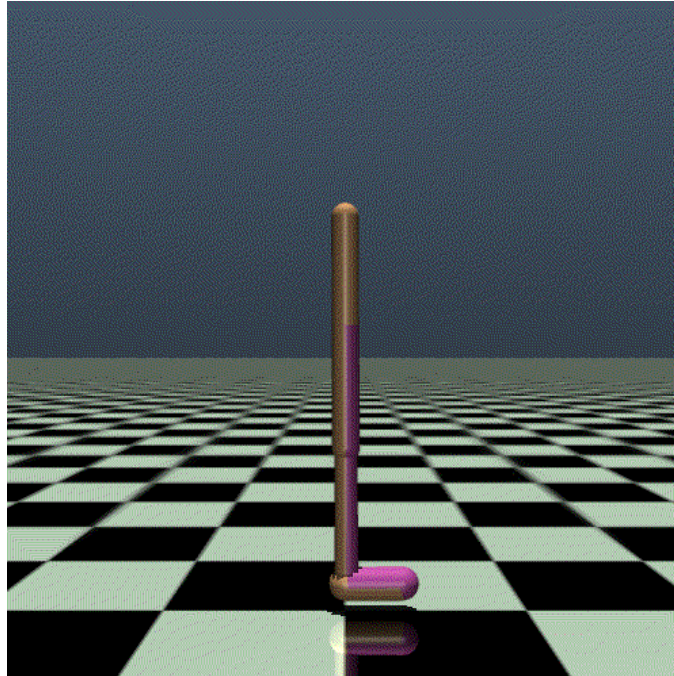


Maze:

- More E/W/N/S

# Example: Continuous Action

---



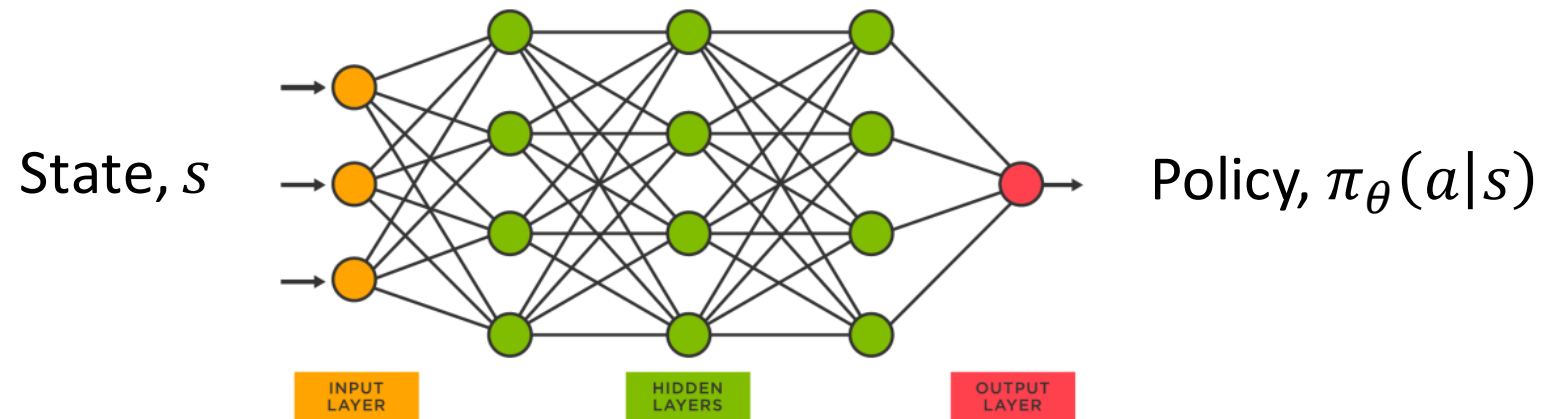
- Some environments have continuous action space.
- Specific values must be chosen in the continuous space.



# Policy-based Methods

---

- In policy-based methods, we directly define the function approximator for the policy itself, rather than values.
- We define the policy as:
  - $\pi_{\theta}(s) \rightarrow$  Probability distribution of all possible actions, given state.
  - $\pi_{\theta}(a|s) \rightarrow$  A single probability of a single action, given state.



# Objective Function

---

Recall that the objective in RL is to maximize the return;

$$\sum_{t=0}^H \gamma^t r(s_t, a_t)$$

Over all possible trajectories,

$$\tau = (s_0, a_0, \dots, s_H)$$

Therefore, we write our objective function as;

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \gamma^t r(s_t, a_t) \right] = \int_{\tau} p_{\theta}(\tau) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau$$

Hence, we must find the neural network parameters,  $\theta$ , by maximizing the above objective function;

$$\theta^* = \underset{\theta}{\operatorname{argmax}} J(\theta)$$

# Policy Gradient

---

To maximize  $J(\theta)$ , we need to compute its gradient;

$$\begin{aligned}\frac{dJ(\theta)}{d\theta} &= \nabla_{\theta} J(\theta) = \nabla_{\theta} \int_{\tau} p_{\theta}(\tau) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau \\ &= \int_{\tau} \nabla_{\theta} p_{\theta}(\tau) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau\end{aligned}$$

Here, we use a simple mathematical trick!

$$\nabla_{\theta} p_{\theta}(\tau) = \frac{p_{\theta}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta} p_{\theta}(\tau) = p_{\theta}(\tau) \frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} = p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)$$

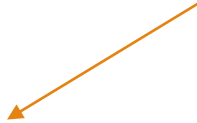
Therefore;

$$\nabla_{\theta} J(\theta) = \int_{\tau} p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau$$

# Policy Gradient

Now consider

Note that  $p_\theta(\tau)$  is the probability of trajectory;  $\nabla_\theta \log p_\theta(\tau)$

$$p_\theta(\tau) = p(s_0)p(a_0) \dots p(s_H) = p(s_0) \prod_{t=0}^H \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$$


Environment's Transitional probability

Therefore;

$$\begin{aligned} \nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log \left( p(s_0) \prod_{t=0}^H \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t) \right) \\ &= \nabla_\theta \left( \log p(s_0) + \sum_{t=0}^H \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t) \right) \end{aligned}$$

Note that only the policy terms are functions of  $\theta$ , therefore;

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^H \nabla_\theta \log \pi_\theta(a_t|s_t)$$

# Policy Gradient

---

Therefore,

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int_{\tau} p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau \\ &= \int_{\tau} p_{\theta}(\tau) \sum_{t=0}^H \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t=0}^H \gamma^t r(s_t, a_t) d\tau \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \left( \sum_{t=0}^H \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right) \left( \sum_{t=0}^H \gamma^t r(s_t, a_t) \right) \right]\end{aligned}$$

Is our final expression of the objective's gradient.

Note that in the gradient term,  $p(s_{t+1} | s_t, a_t)$ , is disappeared, which indicates the algorithm being **model-free**.

# Policy Gradient

---

Add the effect of causality;

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \left( \sum_{t=0}^H \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right) \left( \sum_{t=0}^H \gamma^t r(s_t, a_t) \right) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^k r(s_t, a_t) \right) \right) \right]\end{aligned}$$

Now we incorporated the causality in the expression.

Where  $k$  indicates the earliest timestep from the present timestep.

$$k \geq t$$

# Policy Gradient

---

Continue to manipulate the expression;

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^k r(s_t, a_t) \right) \right) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^t \gamma^{k-t} r(s_t, a_t) \right) \right) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^{k-t} r(s_t, a_t) \right) \right) \right]\end{aligned}$$

# Policy Gradient

---

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^{k-t} r(s_k, a_k) \right) \right) \right]$$

The term,  $\gamma^t$  has some problems;

- For future timesteps, it makes the gradient very small, hence stopping the parameter updates.
- But if we set it to 1, we cannot apply the algorithm to infinite timesteps in environments.

So, we amend our objective function as such, simply by just removing the discounting term,

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H r(s_k, a_k) \right) \right) \right]$$



# Policy Update

---

We now perform gradient ascent algorithm as follows;

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Objective function

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \gamma^t r(s_t, a_t) \right]$$

Assumption

Stochastic policy

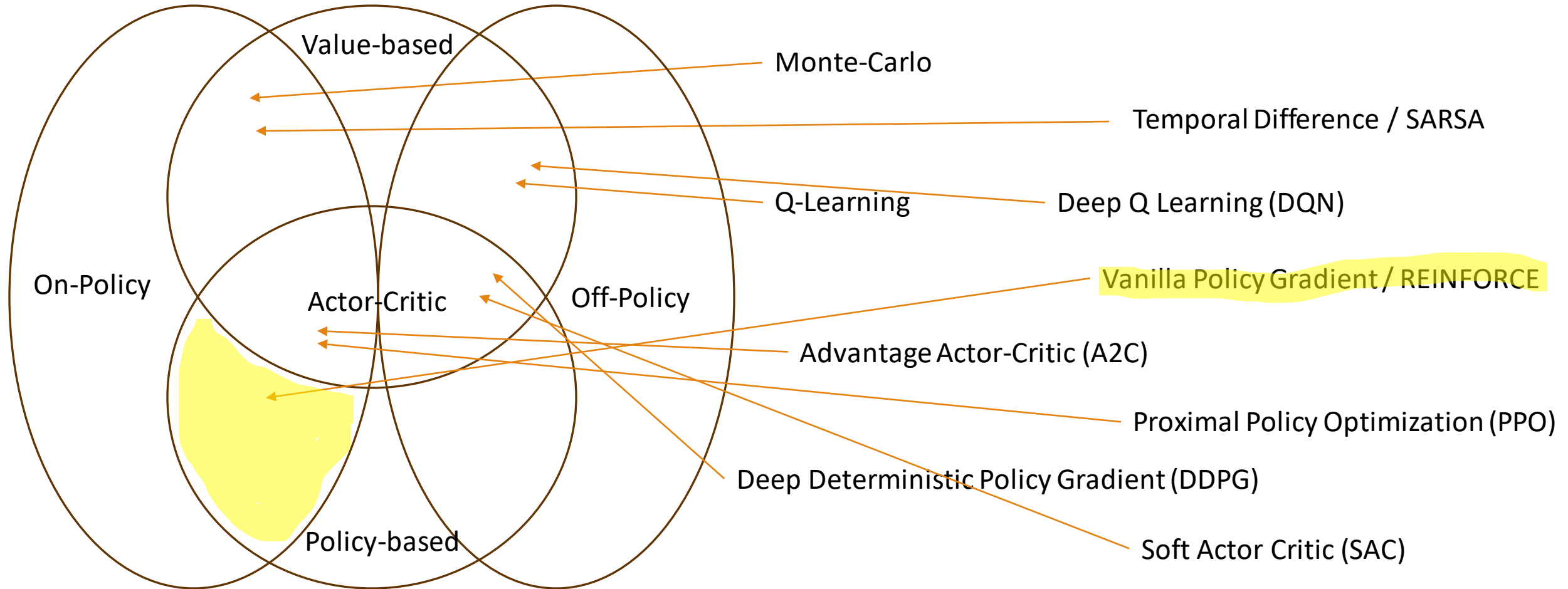
Gradient

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{k=t}^H \gamma^{k-t} r(s_k, a_k) \right) \right) \right]$$

Update

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

# RL Algorithms and Types



# REINFORCE algorithm

---

We start by collecting multiple trajectories;

$$\tau^{(m)} = \{s_0^{(m)}, a_0^{(m)}, s_1^{(m)}, a_1^{(m)}, \dots, s_H^{(m)}\}$$

We approximate the expectation;

$$\mathbb{E}_{\tau \sim p_{\theta}(\tau)}[\cdot] \approx \frac{1}{M} \sum_{m=1}^M [\cdot]$$

Therefore,

$$\begin{aligned} \nabla_{\theta} J(\theta) &\approx \frac{1}{M} \sum_{m=1}^M \left[ \sum_{t=0}^H \left( \nabla_{\theta} \log \pi_{\theta} \left( a_t^{(m)} \middle| s_t^{(m)} \right) \left( \sum_{k=t}^H \gamma^{k-t} r(s_k^{(m)}, a_k^{(m)}) \right) \right) \right] \\ &= \nabla_{\theta} \frac{1}{M} \sum_{m=1}^M \left[ \sum_{t=0}^H \left( \log \pi_{\theta} \left( a_t^{(m)} \middle| s_t^{(m)} \right) G_t^{(m)} \right) \right] \end{aligned}$$

# REINFORCE algorithm

---

From previous lectures on Neural Networks, recall that;

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}}$$

Where  $\mathbf{w}$  is the network parameter and  $L$  is the loss function, which usually takes the **sum of minibatch**.

Also recall that in the case of REINFORCE:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Where  $\theta$  is the network parameter.

But, how about loss?

What can we use as a loss function in our REINFORCE algorithm?

Hint:

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{M} \sum_{m=1}^M \left[ \sum_{t=0}^H \left( \log \pi_{\theta} \left( a_t^{(m)} | s_t^{(m)} \right) G_t^{(m)} \right) \right]$$

# REINFORCE algorithm

---

The loss function:

$$loss^{(m)} = - \sum_{t=0}^H \left( \log \pi_{\theta} \left( a_t^{(m)} \middle| s_t^{(m)} \right) G_t^{(m)} \right)$$

Then;

$$loss = \frac{1}{M} \sum_{m=1}^M [loss^{(m)}]$$

Overall:

$$\begin{aligned} \theta &\leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \\ &\approx \theta - \alpha \nabla_{\theta} \frac{1}{M} \sum_{m=1}^M \left[ \sum_{t=0}^H \left( \log \pi_{\theta} \left( a_t^{(m)} \middle| s_t^{(m)} \right) G_t^{(m)} \right) \right] \end{aligned}$$

# REINFORCE algorithm

---

## REINFORCE Algorithm

1. Start with an **Initial Policy**,  $\pi_0$
2. Collect multiple **trajectories**;

$$\tau^{(m)} = \{s_0^{(m)}, a_0^{(m)}, s_1^{(m)}, a_1^{(m)}, \dots, s_H^{(m)}\}$$

3. Compute the **returns** for all trajectories and timesteps,

$$\sum_{k=t}^H \gamma^{k-t} r(s_t, a_t)$$

4. Compute the loss;

$$loss^{(m)} = - \sum_{t=0}^H \left( \log \pi_{\theta} \left( a_t^{(m)} \middle| s_t^{(m)} \right) G_t^{(m)} \right)$$

5. Update the parameters(policy)

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

# REINFORCE algorithm

---

REINFORCE algorithm is an **on-policy** algorithm.

Unlike DQN, the sample data are **not used again**.

