Deep Reinforcement Learning

Lecture 4 — Function Approximation

國立清華大學
NATIONAL TSING HUA UNIVERSITY

National Tsing Hua University
Department of Computer Science

Prof. Chun-Yi Lee

Outline

- **Function Approximation**
- Value Based Methods
- Policy Gradients Methods
- Actor Critic Methods

Function Approximation

Large state space MDP problem

- Large State Reinforcement Learning
  - Shogi : 10^{71} states
  - GO : 10^{170} states
  - Robotic Arm : Continuous state space

1. Hard to represent all possible states
2. Learn value for each state is slow

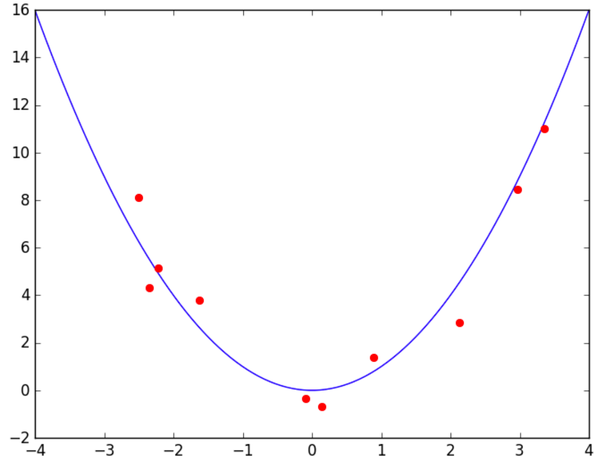
Function Approximation

Large state space MDP problem

- Estimate value function or policy with function approximation

$$V_\theta(s) = V(s, \theta) \approx V_\pi(s)$$
$$Q_\theta(s, a) = Q(s, a, \theta) \approx Q_\pi(s, a)$$
$$\pi_\theta(a, s) = \pi(a \mid s, \theta) \approx \pi(s \mid a)$$

- Generalize from seen states to unseen states
- Less memory used in large state MDPs



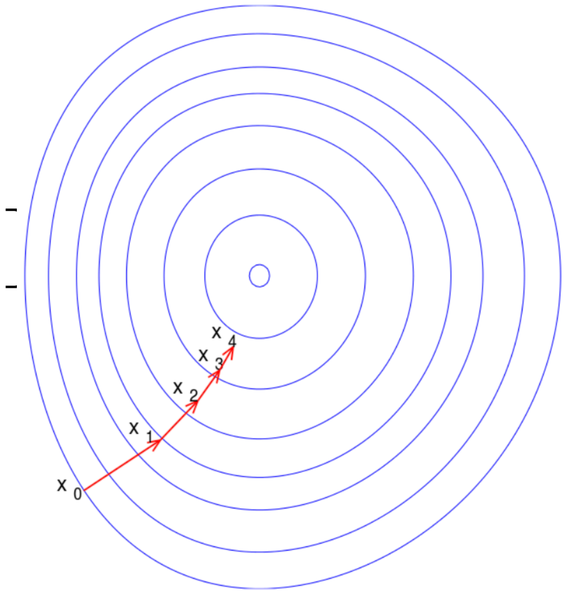
Function Approximation

- Many function approximators :
    - Linear combinations of features
    - Neural network
    - Decision tree
    - Nearest neighbors
    - Fourier/ wavelet bases
- Focus on differentiable function approximator (Why differentiable?)

Function Approximation

Gradient Descent

- Consider $J(\theta)$ that is a differentiable function of a parameter θ
- Find a set of parameter θ that minimize $J(\theta)$
- The gradient of $J(\theta)$
  - $\nabla_\theta J(\theta) = [\frac{\partial J(\theta)}{\theta_1}, \frac{\partial J(\theta)}{\theta_2}, \cdots$
- Adjust parameter by $\theta = \theta - \alpha \nabla_\theta J(\theta)$

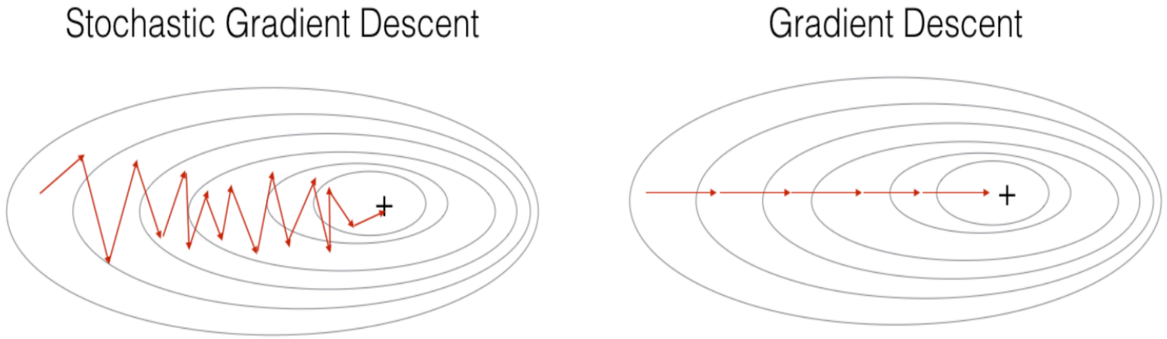


Function Approximation

Stochastic Gradient Descent (SGD)

- **Gradient Descent** uses entire dataset to calculate gradients at once
  - However, for large RL problems, it is impossible to collect all possible data

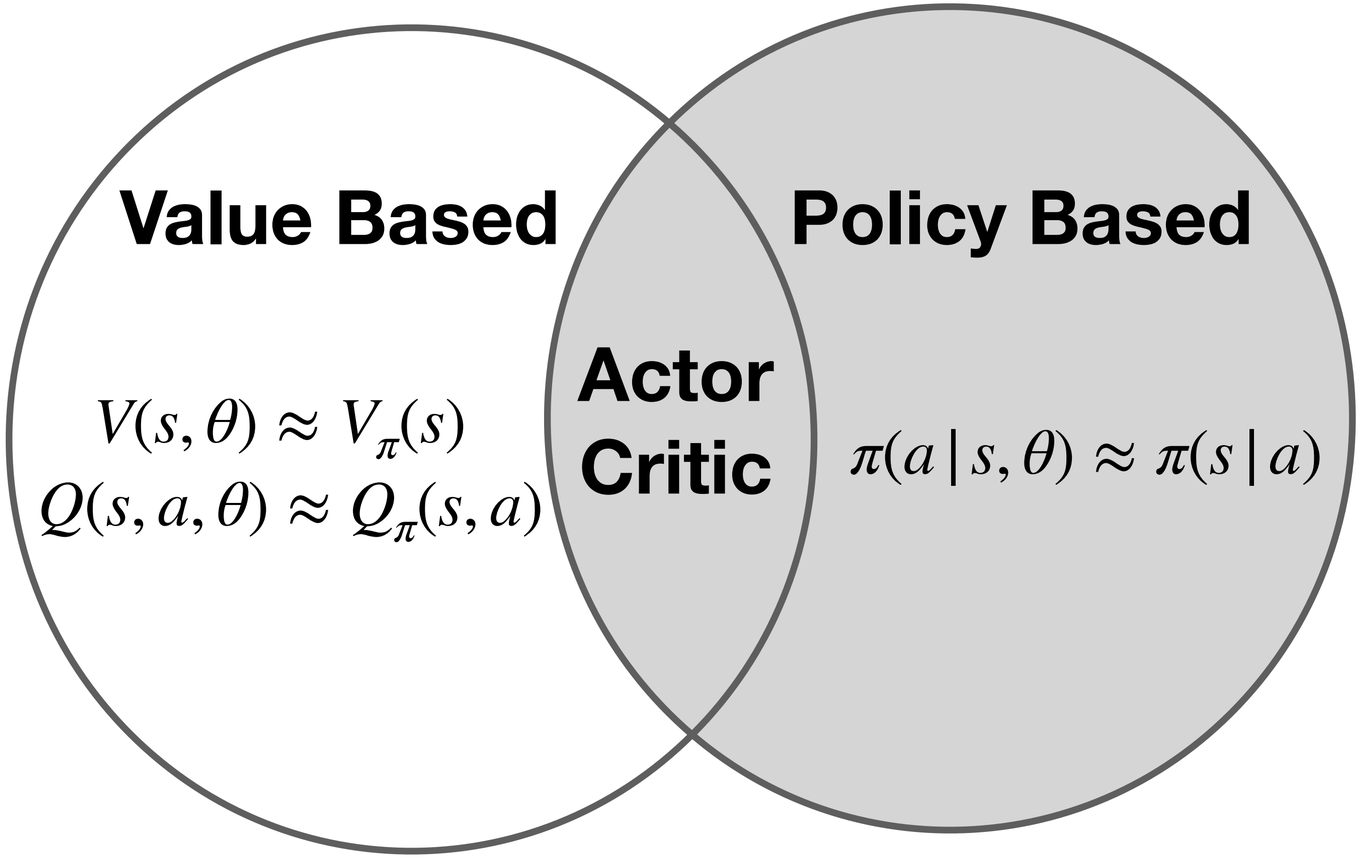
- **Stochastic Gradient Descent** randomly samples mini-batches of data to calculate gradients
  - Adjust parameters by $\theta = \theta - \alpha \nabla_\theta J(\theta)$ with random samples



Function Approximation

Different aspects of function approximation

- Value Based
    - Learn value functions
- Policy Based
    - Learn policies
- Actor Critic
    - Learn both



Outline

- Function Approximation
- **Value Based Methods**
- Policy Gradients Methods
- Actor Critic Methods

Valued Based Method

How to learn a value function?

Generate Sample

01

02

Learn Value Function

03

Improve Policy

Valued Based Method

How to learn a value function?

Learn value functions

How to fit the model?

Regression Problem

1. A lots of data
2. Ground truth labels

Fit the model

A model would predict the labels for any new input data

RL Problem

1. A lot of (S, A, R) pairs
2. Collect samples from an environment

Fit the model

When an unseen (S, A, R) pair occurs, the model can output the predicted value

Value Function Approximation

Value function update method

- Find a set of parameters θ that minimiz the loss between a true value function $V_\pi(s)$ and its approximation $V(s, \theta)$
- Generally, mean squared error (MSE) is used as a measure of loss : $J(\theta) = \mathbb{E}_\pi[(V_\pi(s) - V(s, \theta))^2]$
- SGD generates gradients as follows:
    - $\Delta\theta = \alpha[(V_\pi(s) - V(s, \theta))] \nabla_\theta V(s, \theta)$

Value Function Approximation

Value function update target

- Fit the approximator by $\Delta\theta = \alpha[(V_\pi(s) - V(s,\theta))]\nabla_\theta V(s,\theta)$,
  - However, what is $V_\pi(s)$?
- In practice, we substitute $V_\pi(s)$ by a new target
    - Use G_t from MC,
        - $\Delta\theta = \alpha[G_t - V(s,\theta)]\nabla_\theta V(s,\theta)$
    - Use $R_t + \gamma V(s_{t+1},\theta)$ from TD,
        - $\Delta\theta = \alpha[R_t + \gamma V(s_{t+1},\theta) - V(s,\theta)]\nabla_\theta V(s,\theta)$

Value Function Approximation

Policy Evaluation

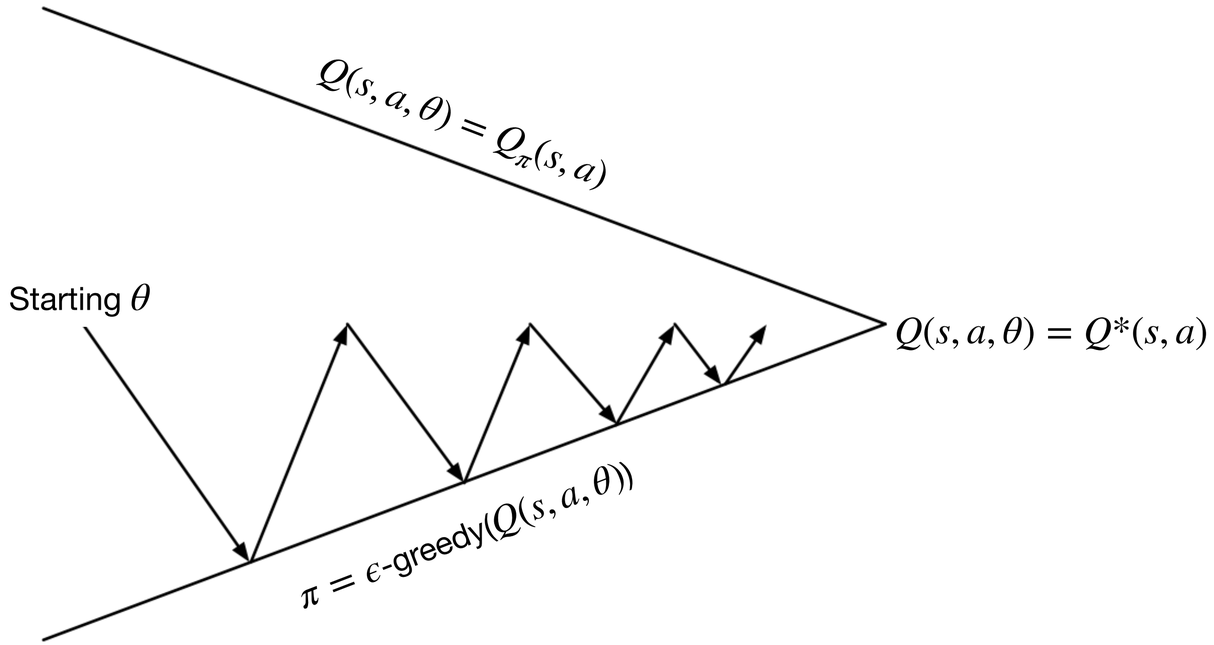
- Tabular method
    - Generate experiences from π
    - Use these experiences to derive tabular values

- Function approximation method
    - Generate experiences from π
    - Fit the function approximator (e.g., SGD) using the experiences

Value Function Approximation

Control using value function approximation

- Policy evaluation — Approximate $Q(s, a, \theta) \approx Q_\pi(s, a)$
- Policy improvement — **ϵ-greedy policy improvement**



Value Function Approximation

State Action value function

- We fit the approximator by
  - Use G_t from **MC**,
    - $\Delta\theta = \alpha[G_t - Q(s_t, a_t, \theta)] \nabla_\theta Q(s_t, a_t, \theta)$
  - Use $R_t + \gamma Q(s_{t+1}, a_{t+1}, \theta)$ from **SARSA**,
    - $\Delta\theta = \alpha[R_t + \gamma Q(s_{t+1}, a_{t+1}, \theta) - Q(s_t, a_t, \theta)] \nabla_\theta Q(s_t, a_t, \theta)$
  - Use $R_t + \gamma \max_a Q(s_{t+1}, a, \theta)$ from **Q-learning**,
    - $\Delta\theta = \alpha[R_t + \gamma \max_a Q(s_{t+1}, a, \theta) - Q(s_t, a_t, \theta)] \nabla_\theta Q(s_t, a_t, \theta)$

Value Function Approximation

State Action value function

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$

Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Repeat (for each episode):
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        If S' is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

Outline

- Function Approximation
- Value Based Methods
- **Policy Gradients Methods**
- Actor Critic Methods

Policy Based Method

How to improve a policy?

Generate Sample

Learn Value Function

No value function to learn

Improve Policy

Directly improve the policy

Policy Based Method

Policy improvement method - one-step MDP

Improve Policy

How to improve ?

Consider a simple case, one-step MDP:

—Terminating after one-step

—d(s) is the distribution of the starting states

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r]$$

$$= \sum d(s) \sum \pi(a \mid s, \theta) R_{s,a}$$

$$\nabla_\theta J(\theta) = \sum d(s) \sum \nabla_\theta \pi(a \mid s, \theta) R_{s,a}$$

$$= \sum d(s) \sum \pi(a \mid s, \theta) \nabla_\theta \log(\pi(a \mid s, \theta)) R_{s,a}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log(\pi(a \mid s, \theta)) r]$$

Likelihood ratios

$$\nabla_\theta \pi(a \mid s, \theta) = \pi(a \mid s, \theta) \frac{\nabla_\theta \pi(a \mid s, \theta)}{\pi(a \mid s, \theta)}$$

$$= \pi(a \mid s, \theta) \nabla_\theta \log(\pi(a \mid s, \theta))$$

Policy Based Method

Policy improvement method - from the perspective of trajectories

Consider the more general case:

The goal of RL

→ Maximize G_t at each timestep $\underline{t}$

→ Find a set of parameters θ that maximize $\mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$,

Trajectory $\tau : s_0, a_0, r_0, s_1, a_1, r_1 \ldots$

$r(\tau) : \sum_t^T \gamma^t r_t$ under τ

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$$

$$= \int p_{\pi_\theta}(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_{\pi_\theta}(\tau) r(\tau) d\tau$$

$$= \int p_{\pi_\theta}(\tau) \nabla_\theta \log(p_{\pi_\theta}(\tau)) r(\tau) d\tau$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log(p_{\pi_\theta}(\tau)) r(\tau)]$$

Policy Based Method

Policy improvement method - from the perspective of trajectories

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log(p_{\pi_\theta}(\tau)) r(\tau)]$$

$$\downarrow \quad \text{Expand } p_{\pi_\theta}(\tau)$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \{\log p(s_0) + \sum_{t=0}^{T} [\log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)]\} r(\tau)]$$

$$= \mathbb{E}_{\tau\sim\pi_\theta} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t=0}^{T} \gamma^t r_t$$

$$p_{\pi_\theta}(\tau) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

$$\downarrow \log \text{ on both sides}$$

$$\log(p_{\pi_\theta}(\tau)) = \log[p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)]$$

$$= \log p(s_0) + \sum_{t=0}^{T} [\log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)]$$

Policy Based Method

Policy improvement method - from the perspective of states

Consider the more general case:

The goal of RL

→ Maximize G_t at each timestep t

→ Find a parameter θ that maximize $V_\pi(s)$

Recall : $V_\pi(s) = \mathbb{E}[G_t \mid s_t = s]$

$$J(\theta) = V_\pi(s)$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta V_\pi(s) \\ &= \nabla \sum_a \pi(a \mid s) Q_\pi(s, a) \\ &= \sum_a \nabla \pi_\theta(a \mid s) Q_\pi(s, a) + \pi_\theta(a \mid s) \nabla Q_\pi(s, a)\end{aligned}$$

Policy Based Method

Policy improvement method - from the perspective of states

$$\nabla_\theta J(\theta) = \nabla_\theta V_\pi(s_0)$$

$$\nabla_\theta V_\pi(s) = \nabla \sum_a \pi(a \mid s) Q_\pi(s, a)$$

$$= \sum_a [\nabla \pi(a \mid s) Q_\pi(s, a) + \pi(a \mid s) \nabla Q_\pi(s, a)]$$

$$= \sum_a [\nabla \pi(a \mid s) Q_\pi(s, a) + \pi(a \mid s) \nabla \sum_{s'} p(s' \mid s, a)(r + V_\pi(s'))]$$

$$= \sum_a [\nabla \pi(a \mid s) Q_\pi(s, a) + \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \nabla V_\pi(s')]$$

Policy Based Method

Policy improvement method - from the perspective of states

$$= \sum_a [\nabla \pi(a|s) Q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla V_\pi(s')]$$

$$= \sum_a \{[\nabla \pi(a|s) Q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a)$$

$$[\sum_{a'} \nabla \pi(a'|s') Q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla V_\pi(s'')]\}$$

$$= \sum_{x \in S} \sum_{k}^{\infty} Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) Q_\pi(x,a)$$

$Pr(s \leftarrow x, k, \pi)$ is the transition probability from state s to state x in k steps

Policy Based Method

Policy improvement method - from the perspective of states

$$\nabla_\theta V_\pi(s) = \sum_{x \in S} \sum_{k}^{\infty} Pr(s \to x, k, \pi) \sum_{a} \nabla \pi(a \mid x) Q_\pi(x, a)$$

$$\nabla_\theta J(\theta) = \nabla_\theta V_\pi(s_0) = \sum_{x \in S} \sum_{k}^{\infty} Pr(s_0 \to x, k, \pi) \sum_{a} \nabla \pi(a \mid x) Q_\pi(x, a)$$

$$= \sum_{s \in S} d_\pi(s) \sum_{a} \nabla \pi(a \mid s) Q_\pi(s, a)$$

$$= \sum_{s \in S} d_\pi(s) \sum_{a} \pi(a \mid s) \frac{\nabla \pi(a \mid s)}{\pi(a \mid s)} Q_\pi(s, a)$$

$$= \mathbb{E}_{\pi_\theta}[\nabla \log \pi(a \mid s) Q_\pi(s, a)]$$

Policy Based Method

Policy improvement method - practical implementation

- True expected gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \log \pi(a \mid s) Q_\pi(s, a)]$$

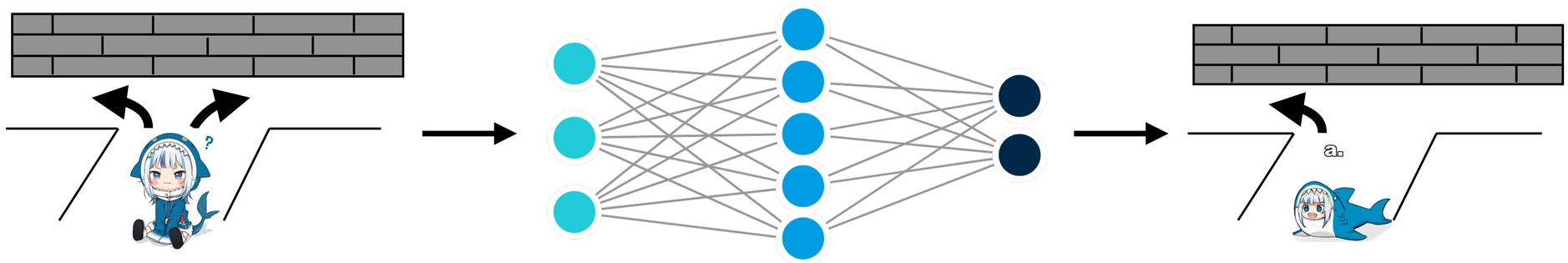
- Stochastic gradient ascent

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi(a_{i,t} \mid s_{i,t}) Q_\pi(s_{i,t}, a_{i,t})$$

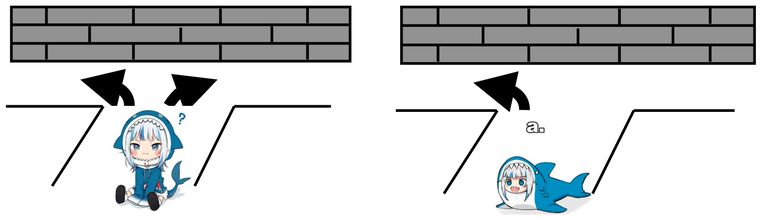
Policy Gradient

Comparison to Maximum Likelihood

If we train an agent with supervised learning:



Collect a lot of training data



(s_t, a_t)

Maximum Likelihood

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})$$

Policy Gradient

Comparison to Maximum Likelihood

Maximum Likelihood: $\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})$

Policy Gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) Q_\pi(s_{i,t}, a_{i,t})$

It looks like a weighted version of maximum likelihood

Policy Gradient

REINFORCE

- Using return G_t as an unbiased sample of Q_π

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^d$
Initialize policy parameter $\boldsymbol{\theta}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    For each step of the episode $t = 0, \ldots, T-1$:
        $G \leftarrow$ return from step t
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t,\boldsymbol{\theta})$

Policy Gradient

REINFORCE - advantages and disadvantages

- Advantage
    - Effective in high-dimensional or continuous action spaces
    - Can learn stochastic policies

- Disadvantage
    - High variance

Policy Gradient

Baselines

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_{\pi_\theta}(\tau)(r(\tau) - b)\, d\tau$$

Same as shifted reward

$$\mathbb{E}[\nabla_\theta p_{\pi_\theta}(\tau) b] = \int \nabla_\theta p_{\pi_\theta}(\tau) b d\tau = b \nabla_\theta \int p_{\pi_\theta}(\tau) d\tau = b \nabla_\theta 1 = 0$$

Does not change the expectation

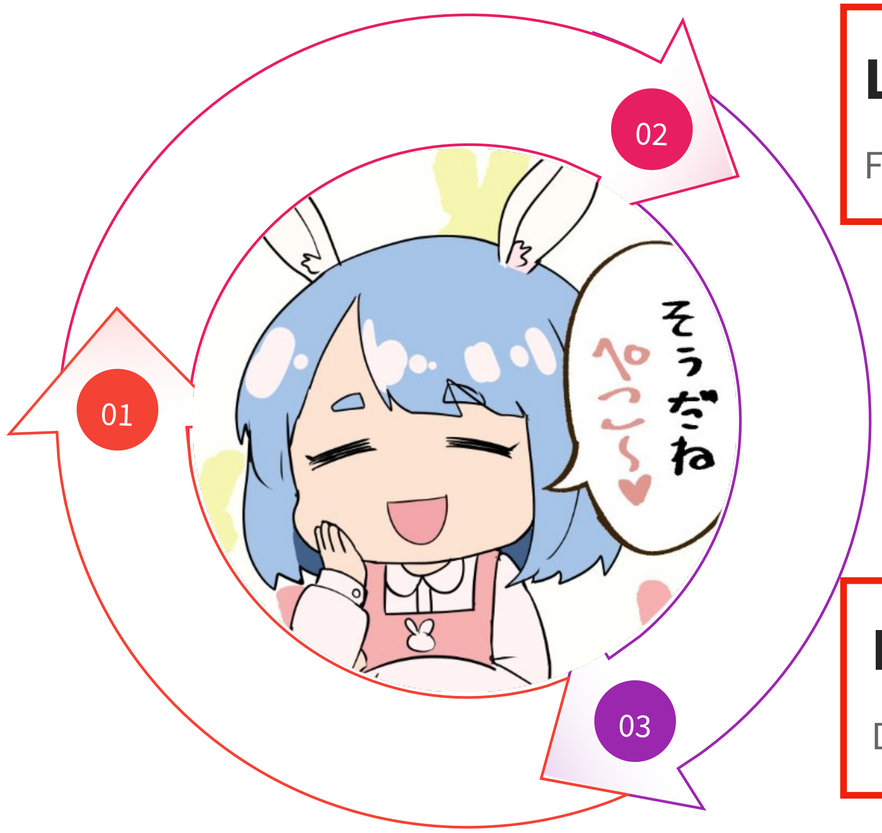
- A good baselines can reduce the variance of policy gradients
- We usually use state value function $V_{\pi_\theta}(s)$ as the baseline

Outline

- Function Approximation
- Value Based Methods
- Policy Gradients Methods
- **Actor Critic Methods**

Actor Critic Method

Generate Sample



Learn Value Function

Fit the value model by collection samples

Improve Policy

Directed improve the policy

Actor Critic

What is actor-critic

- Actor Critic method maintains two components
  - Critic : Action-Value function parameterized with w -> the estimated the Q-value.
  - Actor : Policy, updated through policy gradient in the direction suggested by the Critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \, Q_{\pi_\theta}(s_{i,t}, a_{i,t})$$

Actor Critic

What is actor-critic

- Actor Critic method maintains two components
  - Critic : Action-Value function parameterized with w -> the estimated the Q-value.
  - Actor : Policy, updated through policy gradient in the direction suggested by the Critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \, Q_w(s_{i,t}, a_{i,t})$$

Actor Critic

What is actor-critic

Action-Value Actor-Critic

- Simple actor-critic algorithm based on action-value critic
- Using linear value fn approx. $Q_w(s,a) = \phi(s,a)^\top w$

  Critic Updates w by linear TD(0)

  Actor Updates θ by policy gradient

function QAC

  Initialise s, θ

  Sample $a \sim \pi_\theta$

  for each step **do**

    Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$

    Sample action $a' \sim \pi_\theta(s', a')$

    $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$

    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$

    $w \leftarrow w + \beta \delta \phi(s, a)$

    $a \leftarrow a', s \leftarrow s'$

  end for

end function

Actor Critic

Compatible Function Approximation

- Approximating the policy gradient introduces bias

$$Q_w(s,a) \approx Q_{\pi_\theta}(s,a)$$

- A biased policy gradient may not find the right solution
- Luckily, if we choose a proper value function approximation method, we can avoid the bias problem
  - i.e., we can still follow the exact policy gradient

Actor Critic

Bias of the Q-function

- If the following two conditions are satisfied:
  - Value function approximator is compatible to the policy

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

  - Value function parameters *w* minimize the mean-squared error

$$\epsilon = \mathbb{E}_{\pi_\theta}[(Q_{\pi_\theta}(s,a) - Q_w(s,a))^2]$$

- Then the policy gradient is exact

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\nabla_\theta \log \pi_\theta(a,s) Q_w(s,a)]$$

Actor Critic

Compatible condition

- For example, if a policy is a Boltzmann distribution with linear features:

$$\pi(s,a) = \frac{e^{\theta^T \phi(s,a)}}{\sum_{a' \in A} e^{\theta^T \phi(s,a')}}$$

- We need to choose a function approximator satisfying the compatible condition:

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \sum_{a' \in A} \pi(s,a')\phi(s,a')$$

- Thus, the function approximator can be represented as follows:

$$Q_w(s,a) = w^T[\phi(s,a) - \sum_{a' \in A} \pi(s,a')\phi(s,a')]$$

Actor Critic

Proof

- If we choose a value function approximate that meets the compatible condition, we can minimize *w* using mean-squared error (MSE) ε. The gradient of ε is derived as:

$$\nabla_w \epsilon = 0$$

$$\mathbb{E}_{\pi_\theta}[(Q_{\pi_\theta}(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)] = 0$$

$$\mathbb{E}_{\pi_\theta}[(Q_{\pi_\theta}(s,a) - Q_w(s,a))\nabla_\theta \pi_\theta(s,a)] = 0$$

$$\mathbb{E}_{\pi_\theta}[Q_{\pi_\theta}(s,a)\nabla_\theta \pi_\theta(s,a)] = \mathbb{E}_{\pi_\theta}[Q_w(s,a)\nabla_\theta \pi_\theta(s,a)]$$

- Therefore, $Q_w(s,a)$ can be substituted directly into the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a,s) Q_w(s,a)]$$

Actor Critic

Baseline

- $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}\,|\,s_{i,t})[\underbrace{Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s)}_{A_{\pi_\theta}(s,a)}]$$

- $A_{\pi_\theta}(s,a)$ is called the Advantage Function,
  - It defines how much better an action is than the other actions
- The advantage function can significantly reduce variance of the policy gradients

Actor Critic

Advantage function estimation (1/2)

- Following the above derivation, we would like to estimate $A_{\pi_\theta}(s, a)$
- We can estimate it from both $V_{\pi_\theta}(s)$ and $Q_{\pi_\theta}(s, a)$ using function approximation as follows

$$V_{w_1}(s) \approx V_{\pi_\theta}(s)$$

$$Q_{w_2}(s, a) \approx Q_{\pi_\theta}(s, a)$$

$$A(s, a) = Q_{w_2}(s, a) - V_{w_1}(s)$$

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) A(s, a)$$

Need three sets of parameters

Actor Critic

Advantage function estimation (2/2)

- $A_{\pi_\theta}(s,a) = Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s)$, where $Q_{\pi_\theta}(s,a) = r_t + V_{\pi_\theta}(s_{t+1})$
- $A_{\pi_\theta}(s,a) = r(s_t,a_t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s)$

Same as TD-error for updating V

- In such a case, we only need to fit $V_{\pi_\theta}(s)$ by TD learning for deriving $A_{\pi_\theta}(s,a)$

$$V_{w_1}(s) \approx V_{\pi_\theta}(s)$$

$$A(s,a) = \delta_v = r(s,a) + V_w(s') - V_w(s)$$

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \delta_v$$

Only two sets of parameters are required

Actor Critic

Actor Critic Algorithm

- Actor Critic
  - Initialize $\pi_\theta(a \mid s)$ with parameter θ, $V_w(s)$ with parameter w
  - Sample $\{a, s\}$ from $\pi_\theta(a \mid s)$
  - $\delta \leftarrow r + \gamma V_w(s') - V_w(s)$
  - $w \leftarrow w + \alpha \delta \nabla_w V_w(s)$
  - $\theta \leftarrow \theta + \beta \nabla_\theta \log \pi_\theta(s \mid a) \delta$

Actor Critic

Critics at Different Time-Scale

- The critic can also be updated using different approaches :
  - Use G_t from MC,
    - $\Delta\theta = \alpha[G_t - V_\theta(s)]\nabla_\theta V_\theta(s)$
  - Use $R_t + \gamma V_\theta(s_{t+1})$ from TD,
    - $\Delta\theta = \alpha[R_t + \gamma V_\theta(s_{t+1}) - V_\theta(s)]\nabla_\theta V_\theta(s)$
  - Use forward-view TD(λ) return,
    - $\Delta\theta = \alpha[G_t^\lambda - V_\theta(s)]\nabla_\theta V_\theta(s)$

Actor Critic

Actors at Different Time-Scale

- Actor can also estimate policy gradient at many time-scale:
  - Use A,

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a \mid s) A(s, a)$$

  - Use G_t as MC,

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)(G_t - V(s_t))$$

  - Use one-step TD,

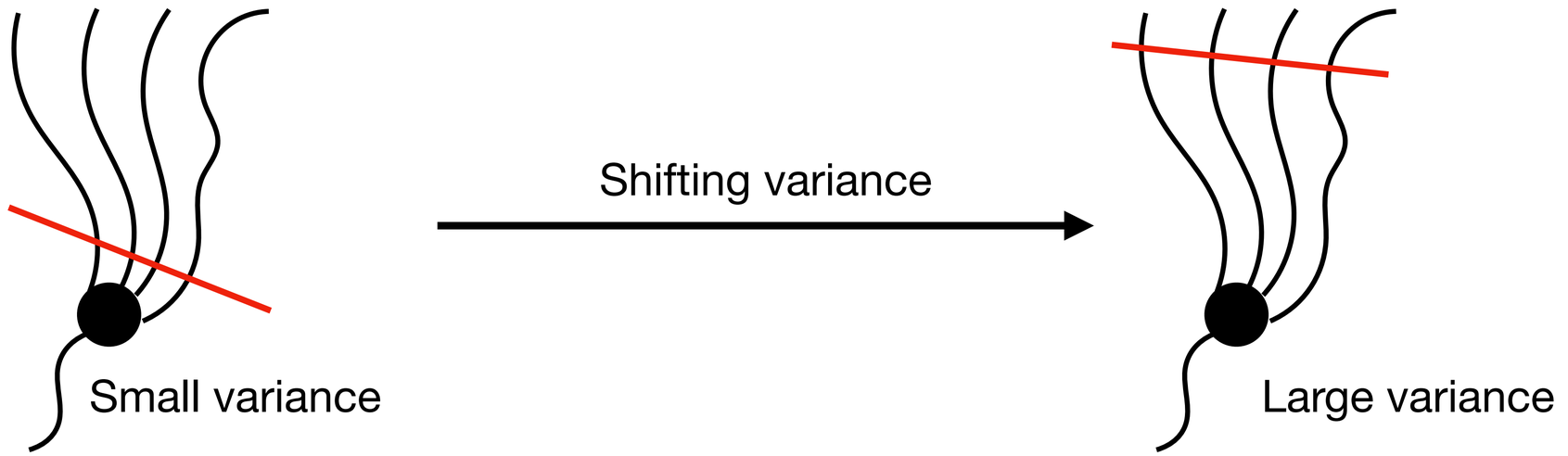
$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)(r + V(s_{t+1}) - V(s_t))$$

Actor Critic

N-step returns

- Use n-step TD

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\sum_{i=0}^{n} r_i + V(s_{n+1}) - V(s_t)\right)$$




ありがとう
ございます♥