Lecture 4: Approximate solution methods of RL

Haizhao Yang
Department of Mathematics
University of Maryland College Park

2022 Summer Mini Course
Tianyuan Mathematical Center in Central China

Outline

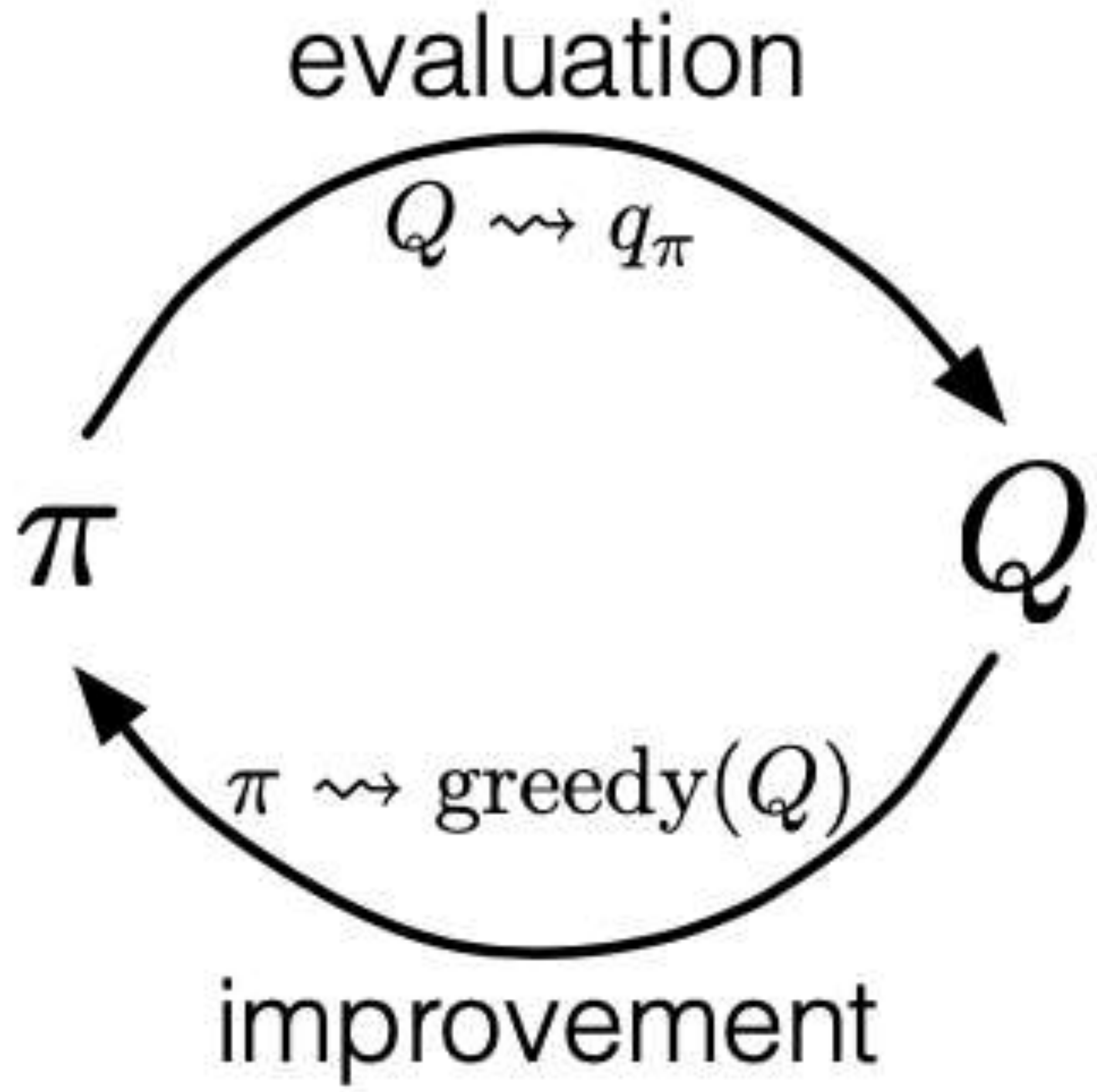
- Introduction to reinforcement learning (RL)
- Tabular solution methods of RL
- **Approximate solution methods of RL**

Reference: Reinforcement Learning: An Introduction. Richard S. Sutton and Andrew G. Barto. Second Edition. MIT Press, Cambridge, MA, 2018

Approximate Solution Methods

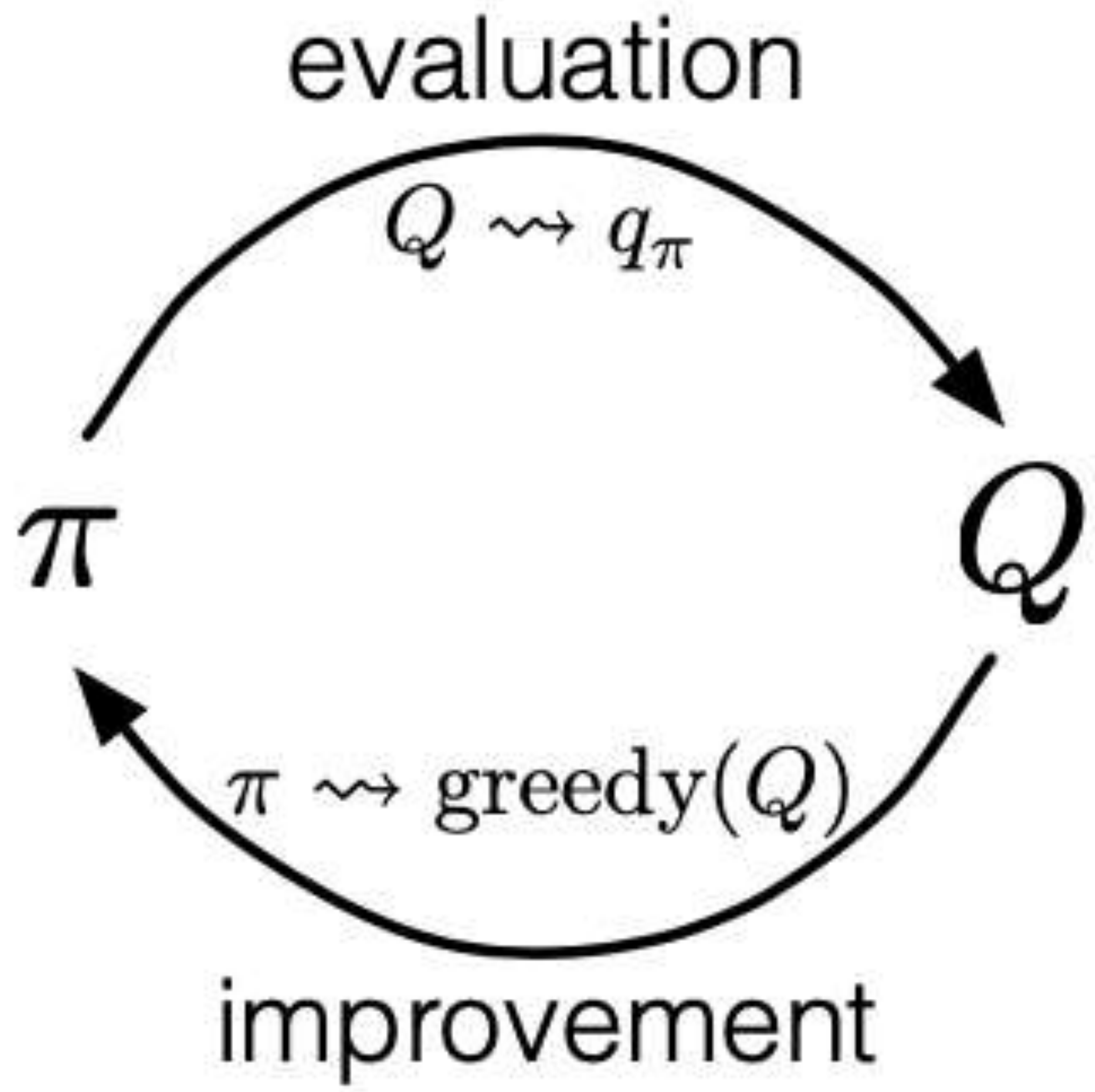
Approximate Solution Methods

- **On-policy Prediction with Approximation**
- **On-policy Control with Approximation**
- **Off-policy Methods with Approximation**
- **Policy Gradient Methods**



On-policy Prediction with Approximation

Value-function Approximation



By Richard S. Sutton and Andrew G. Barto.

State-action value evaluation as before: $\pi_k(s) \to q_{\pi_k}(s,a)$

Greedy policy improvement: $\pi_{k+1}(s) = \arg\max_a q_{\pi_k}(s,a)$

Question: How to discretize and do computation on computers?

On-policy Prediction with Approximation

Value-function Approximation

Question: How to discretize and do computation on computers?

Case 1: Finite MDP

- $Q(s, a)$ is a matrix representing the value function of all the state-action pairs (s, a)
- V(s) is a vector representing the value function of all states s

Case 2: Continuous MDP

- $Q(s, a)$ and V(s) are functions discretized via basis functions or neural networks
- Basis functions: polynomials; Fourier series expansion; radial basis functions; coarse coding; tile coding

On-policy Prediction with Approximation

Value-function Approximation

Question: How to discretize and do computation on computers?

Comparison

- Polynomials, Fourier series expansion, radial basis functions:
  Target function with clear structure or explicit formulas
- Coarse coding and tile coding:
  Target function without clear structure or explicit formulas

On-policy Prediction with Approximation

Value-function Approximation

Question: How to update value functions on computers?

Case 1: Finite MDP

- E.g, Sarsa updating rule with a vector V:
  $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- Choose the entry for S_t and assign the new value
- Update one entry and other entries unchanged

On-policy Prediction with Approximation

Value-function Approximation

Question: How to update value functions on computers?

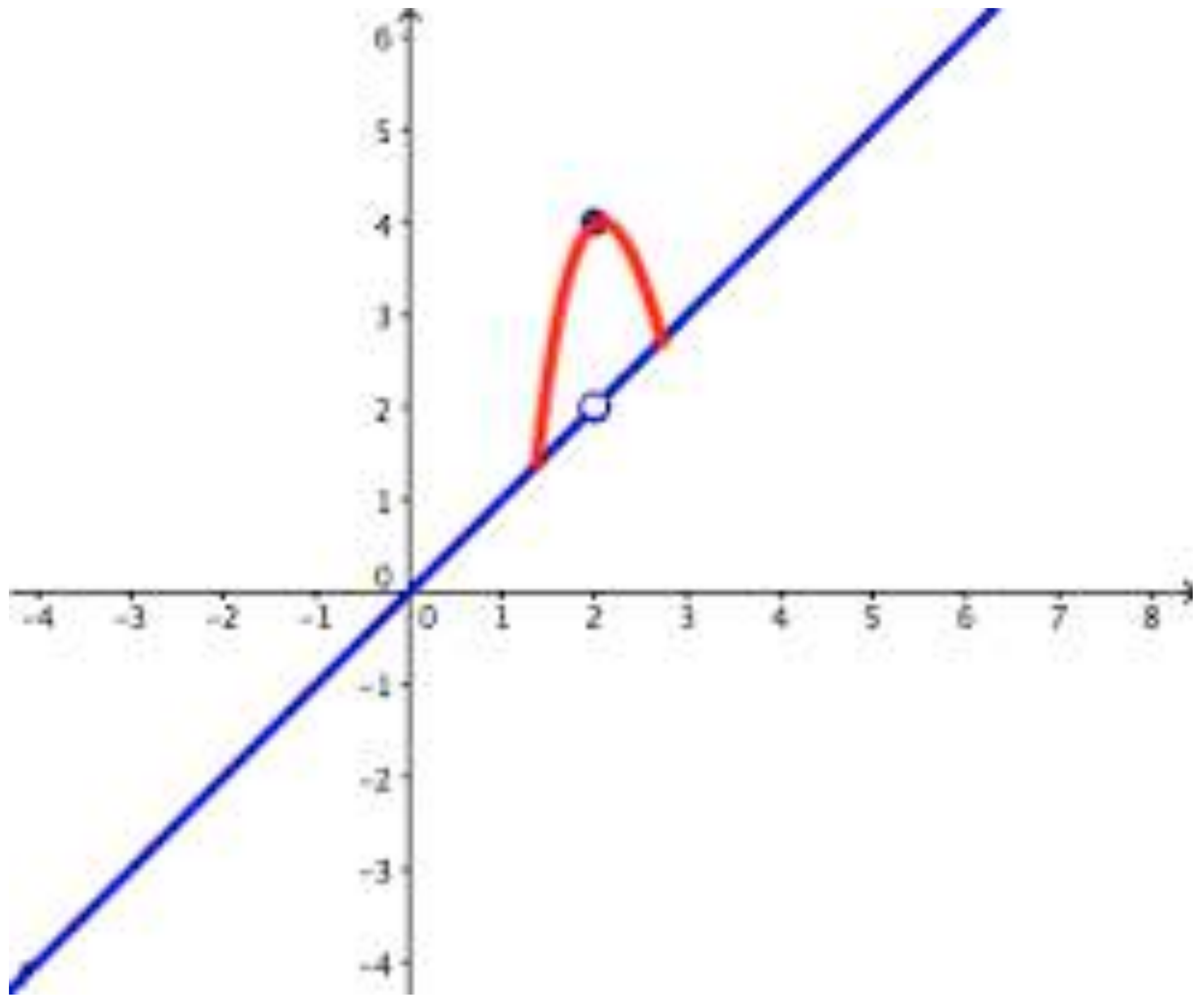
Case 2: Continuous MDP

- E.g, Sarsa updating rule:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

- Polynomial approximation:

$$V(s) \approx \hat{V}(s, \mathbf{w}) := \mathbf{w}^T \mathbf{x}(s) := \sum_{i=1}^{d} w_i x_i(s)$$



- Updating $\hat{V}(s, \mathbf{w})$ at the point S_t will change the value of $\hat{V}$ at other locations
- Update one value requires solving expensive least squares
- But we want cheap online update without disturbing other values too much

On-policy Prediction with Approximation

Value-function Approximation

Question: How accurate the learned value function we expect?

Case 1: Finite MDP

- The target value function v_* is a vector
- The learned value function V_t is a vector
- $\|V_t - v_*\|$ quantifies the learned accuracy
- $\|V_t - v_*\|$ decays to zero as $t \to \infty$ by the law of large numbers

On-policy Prediction with Approximation

Value-function Approximation

Question: How accurate the learned value function we expect?

Case 2: Continuous MDP

- The target v_* is a function
- The learned V_t is a function by a certain approximation algorithm
- $\|V_t - v_*\|$ quantifies the learned accuracy
- $\|V_t - v_*\|$ won't decay to zero as $t \to \infty$ due to approximation error

On-policy Prediction with Approximation

Value-function Approximation

Continuous MDP

- $\|V_t - v_*\|$ quantifies the learned accuracy
- Different states may have different importance
- Introduce a state distribution $\mu(s) \geq 0$ with $\sum_{s \in \mathcal{S}} \mu(s) = 1$
- Introduce the weighted mean square error (MSE)

$$\overline{\mathrm{VE}}(\mathbf{w}) := \sum_{s \in \mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w})]^2$$

On-policy Prediction with Approximation

Value-function Approximation

Continuous MDP

- The best value function approximation needs to minimize the MSE

$$\overline{\mathrm{VE}}(\mathbf{w}) := \sum_{s \in \mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w})]^2$$

- This problem leads to updating rules of **w**
- There may be many global minimizers
- We only expect to obtain a local minimizer

On-policy Prediction with Approximation

Value-function Approximation

Continuous MDP

- Objective function

$$\overline{\mathrm{VE}}(\mathbf{w}) := \sum_{s\in\mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w})]^2$$

- Gradient descent updating rule:

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \frac{1}{2}\alpha \nabla \sum_{s\in\mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w})]^2$$

$$= \mathbf{w}_t + \alpha \sum_{s\in\mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w})] \nabla \hat{v}(s, \mathbf{w})$$

- However, we don't know $v_*(s)$!

On-policy Prediction with Approximation

Value-function Approximation

Continuous MDP

- Gradient descent updating rule:

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \frac{1}{2}\alpha \nabla \sum_{s\in\mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w}_t)]^2$$

$$= \mathbf{w}_t + \alpha \sum_{s\in\mathcal{S}} \mu(s)[v_*(s) - \hat{v}(s, \mathbf{w}_t)] \nabla \hat{v}(s, \mathbf{w}_t)$$

- We have $\mathbf{w}_t$ and $\hat{v}(s, \mathbf{w}_t)$
- We have data to get some samples

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] := U_t$$

i.e., we want $V(S_t) = U_t$ to obtain a better estimation of $v_*(S_t)$

- How can we use the above known information?

On-policy Prediction with Approximation

Value-function Approximation

Continuous MDP

- Stochastic gradient descent updating rule:

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \frac{1}{2}\alpha \nabla \cancel{\sum_{S_t}} \mu(S_t)[\cancel{v_*(S_t)}U_t - \hat{v}(S_t, \mathbf{w}_t)]^2$$

$$= \mathbf{w}_t + \alpha \cancel{\sum_{S_t}} \mu(S_t)[\cancel{v_*(S_t)}U_t - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

$$= = \mathbf{w}_t + \alpha\mu(S_t)[U_t - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

- We have $\mathbf{w}_t$ and $\hat{v}(s, \mathbf{w}_t)$
- We have data to get some samples

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] := U_t$$

i.e., we want $V(S_t) = U_t$ to obtain a better estimation of $v_*(S_t)$

On-policy Prediction with Approximation

Value-function Approximation

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \rightarrow \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
    Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G_t - \hat{v}(S_t,\mathbf{w})\big]\nabla\hat{v}(S_t,\mathbf{w})$

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

First visit MC for policy evaluation: average of the returns following first visits to s

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: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t)$

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

By Richard S. Sutton and Andrew G. Barto.

On-policy Prediction with Approximation

Value-function Approximation

$\hat{q} \approx q_\pi$

Gradient Monte Carlo Algorithm for Estimating $\cancel{\hat{v} \approx v_\pi}$

Input: the policy π to be evaluated
Input: a differentiable function $\cancel{\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}}$ $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G_t - \cancel{\hat{v}(S_t,\mathbf{w})\big]\nabla\hat{v}(S_t,\mathbf{w})}$ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[G_t - \hat{q}(S_t, A_t, \mathbf{w})]\nabla\hat{q}(S_t, A_t, \mathbf{w})$

By Richard S. Sutton and Andrew G. Barto.

On-policy Prediction with Approximation

Value-function Approximation

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S is terminal

By Richard S. Sutton and Andrew G. Barto.

Temporal-Difference Learning

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

By Richard S. Sutton and Andrew G. Barto.

On-policy Prediction with Approximation

Value-function Approximation

Semi-gradient TD(0) for estimating $\cancel{\hat{v} \approx v_\pi}$ $\hat{q} \approx q_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\cancel{\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$ $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S, then choose $A \sim \pi(\cdot \mid S)$
  Loop for each step of episode:
    ~~Choose $A \sim \pi(\cdot|S)$~~
    Take action A, observe R, S', then choose $A' \sim \pi(\cdot \mid S')$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \cancel{\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})}$
    $S \leftarrow S'$, $A \leftarrow A'$
  until S is terminal

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})]\nabla\hat{q}(S_t, A_t, \mathbf{w})$$

By Richard S. Sutton and Andrew G. Barto.

Temporal-Difference Learning

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

results of the same policy

By Richard S. Sutton and Andrew G. Barto.

On-policy Control with Approximation

Value-function Approximation

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

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Loop 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)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

By Richard S. Sutton and Andrew G. Barto.

Approximation Methods

Linear approximation for continuous MDP:

- Approximation: $v(s) \approx \hat{v}(s, \mathbf{w}) := \mathbf{W}^T \mathbf{x}(s) := \sum_{i=1}^{d} w_i x_i(s)$
- Gradient in **w**: $\nabla \hat{v}(s, \mathbf{w}) := \mathbf{x}(s)$
- Updating rule: $\mathbf{w}_{t+1} := \mathbf{w}_t + \alpha\mu(S_t)[U_t - \hat{v}(S_t, \mathbf{w}_t)]\mathbf{x}(S_t)$

Approximation Methods

Non-linear approximation for continuous MDP:

- Approximation: $v(s) \approx \hat{v}(s, \mathbf{w})$ as a deep neural network
- Gradient in $\mathbf{w}$: $\nabla \hat{v}(s, \mathbf{w})$ via backpropagation
- Updating rule: $\mathbf{w}_{t+1} := \mathbf{w}_t + \alpha\mu(S_t)[U_t - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(s, \mathbf{w})$

On-policy Methods with Approximation

Topics not covered:

- Off-policy prediction with approximation
- Off-policy control with approximation

Policy Gradient with Approximation

Value function methods

- Evaluate value functions $v_\pi(s, \mathbf{w})$ or $q_\pi(s, a, \mathbf{w})$
- Use greedy method to derive greedy policies from value functions

Policy gradient methods

- Evaluate the policy $\pi(a \mid s, \theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$ as the probability for taking a given s with the approximation parameter θ
- Directly estimate the optimal θ to maximize rewards
- Stronger convergence guarantee than the value function methods

Actor-Critic methods

- Evaluate the policy $\pi(a \mid s, \theta)$ for actor
- Evaluate value functions for critic, usually state value function
- A special case of policy gradient methods

Policy Gradient with Approximation

Policy gradient methods

- Evaluate the policy $\pi(a \mid s, \theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$ as the probability for taking a given s with the approximation parameter θ
- Directly estimate the optimal θ to maximize rewards

How to find the best θ?

- Define a performance measure $J(\theta)$ and maximize it
- e.g., $J(\theta) := v_{\pi_\theta}(s_0)$ for an episode starting with a state s_0
- Gradient ascend: $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$
- $\widehat{\nabla J(\theta_t)}$ is a stochastic estimate whose expectation approximates $\nabla_{\theta_t} J(\theta_t)$

Policy Gradient with Approximation

How to discretize the policy $\pi(a \mid s, \theta)$?

- Linear methods: polynomials, Fourier basis, etc.
- Nonlinear methods: deep neural networks
- Challenging requirement:
    1. $\sum_{a \in \mathcal{A}(s)} \pi(a \mid s, \theta) = 1$ for all states s
    2. $\pi(a \mid s, \theta) \in (0,1)$ for all state-action pairs to ensure exploration
    3. $\pi(a \mid s, \theta)$ can approach to deterministic policy, i.e., a greedy policy

Policy Gradient with Approximation

Define the policy $\pi(a \mid s, \theta)$ through preference $h(s, a, \theta)$

- $\pi(a \mid s, \theta) := \dfrac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$
- $h(s, a, \theta)$ is easier to discretize without special requirements
- $\pi(a \mid s, \theta)$ approaches to deterministic if $h(s, a, \theta)$ allowed to be infinite
- Linear or nonlinear methods to discretize $h(s, a, \theta)$
- This parametrization is called the soft-max in action preference

Policy Gradient with Approximation

How to find the best θ?

- Define a performance measure $J(\theta) := v_{\pi_\theta}(s_0)$
- Gradient ascend: $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$
- $\widehat{\nabla J(\theta_t)}$ is a stochastic estimate whose expectation approximates $\nabla_{\theta_t} J(\theta_t)$

How to evaluate the gradient?

- Policy gradient theorem: $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a \mid s, \theta)$
- Need to evaluate $q_\pi(s,a)$ and $\nabla \pi(a \mid s, \theta)$ to complete one step of gradient ascend

Policy Gradient with Approximation

MC method

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right].$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta}),$$

Policy Gradient with Approximation

MC method

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \\
&= \mathbb{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] && \text{(replacing } a \text{ by the sample } A_t \sim \pi) \\
&= \mathbb{E}_\pi\left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right], && \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))
\end{aligned}
$$

$$
\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}.
$$

Policy Gradient with Approximation

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

By Richard S. Sutton and Andrew G. Barto.