Lecture 3: Tabular 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

Tabular Solution Methods

Tabular Solution Methods

- Multi-Armed Bandits
- Finite Markov Decision Process
- Dynamic Programming
- Monte Carlo Methods
- Temporal Difference Learning

K-Armed Bandit Problem

- **Set up**
  - Action A_t at time t is $a \in \{a_1, a_2, \ldots, a_K\}$
  - Obtain a random reward R_t following unknown distribution p_{A_t}
- **Goal**
  - Maximize the total reward over time, i.e.,

$$R_1 + R_2 + R_3 + \ldots, R_n + \ldots$$

as a consequence of

$$A_1, A_2, A_3, \ldots, A_n, \ldots$$

  - Assuming independence, and remove randomness by expectation, it is sufficient to maximize

$$\mathbb{E}[R_1 | A_1] + \mathbb{E}[R_2 | A_2] + \mathbb{E}[R_3 | A_3] + \ldots, \mathbb{E}[R_n | A_n] + \ldots$$

which is equivalent to maximizing $\mathbb{E}[R_t | A_t]$ by choosing A_t

K-Armed Bandit Problem

- **New goal**
  - Maximize $\mathbb{E}[R_t | A_t]$ by choosing A_t
- **True value function**
  - $q_*(a) := \mathbb{E}[R_t | A_t = a]$ tells us how to choose the action A_t

$$A_t = \arg\max_a q_*(a) = \arg\max_a \mathbb{E}[R_t | A_t = a]$$

to maximize $\mathbb{E}[R_t | A_t]$

- **Estimated value function**
  - Find $Q_t(a) \approx q_*(a)$ through interactions

$$A_0, R_0, A_1, R_1, \ldots, A_n, R_n, \ldots$$

K-Armed Bandit Problem

New goal: find $Q_t(a) \approx q_*(a)$ through interactions

$$A_0, R_0, A_1, R_1, \ldots, A_n, R_n, \ldots$$

Action-value methods

- Estimation

$$Q_t(a) := \frac{\text{sum of rewards when a taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}}$$

- Greedy selection

$$A_t = \arg\max Q_t(a)$$

  - Each action will be explored at least once
  - "Local maximizer", if the rewards for $\arg\max_a q_*(a)$ happen to be smaller than they should be
  - No exploration to escape "local maximizers"

K-Armed Bandit Problem

New goal: find $Q_t(a) \approx q_*(a)$ through interactions

$$A_0, R_0, A_1, R_1, \ldots, A_n, R_n, \ldots$$

Action-value methods

- Estimation

$$Q_t(a) := \frac{\text{sum of rewards when a taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}}$$

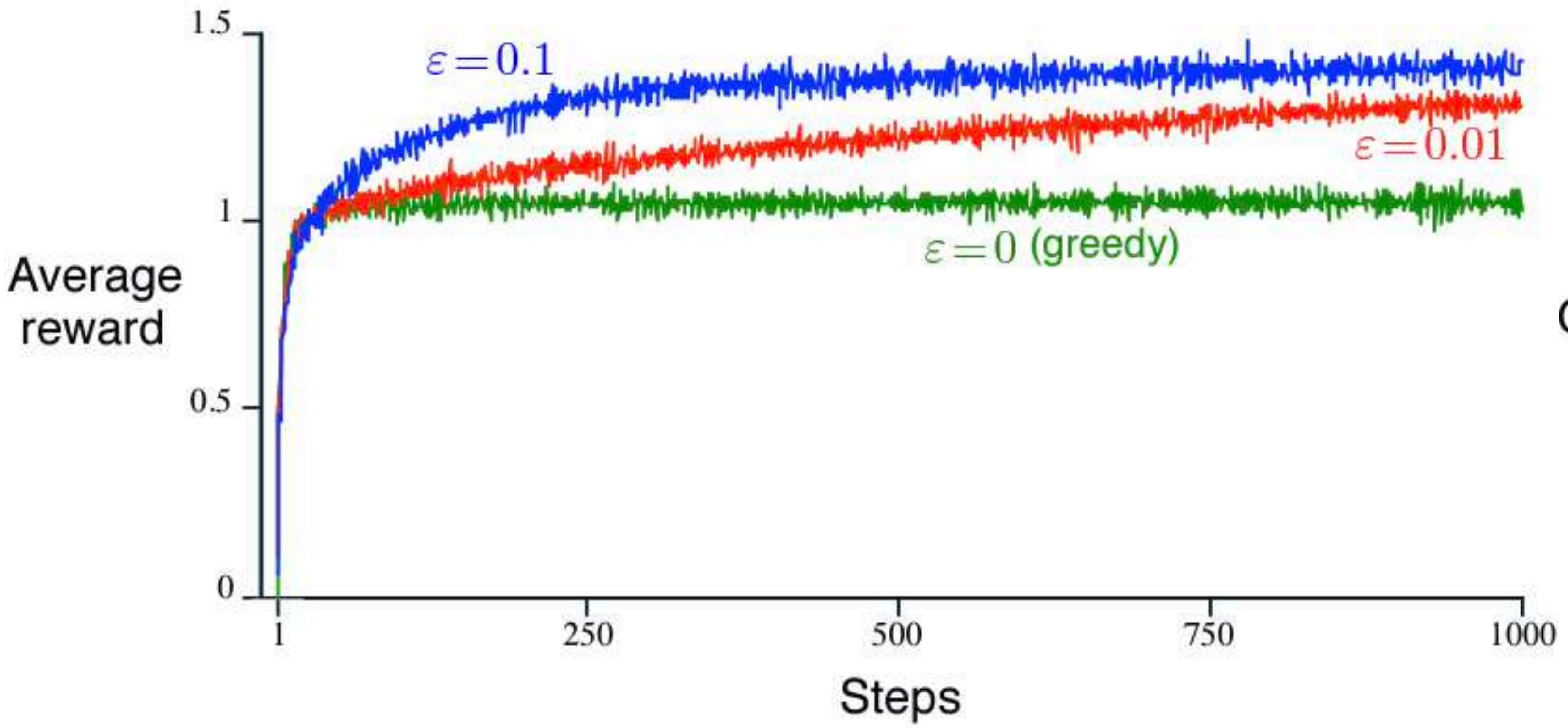
- ε-greedy selection

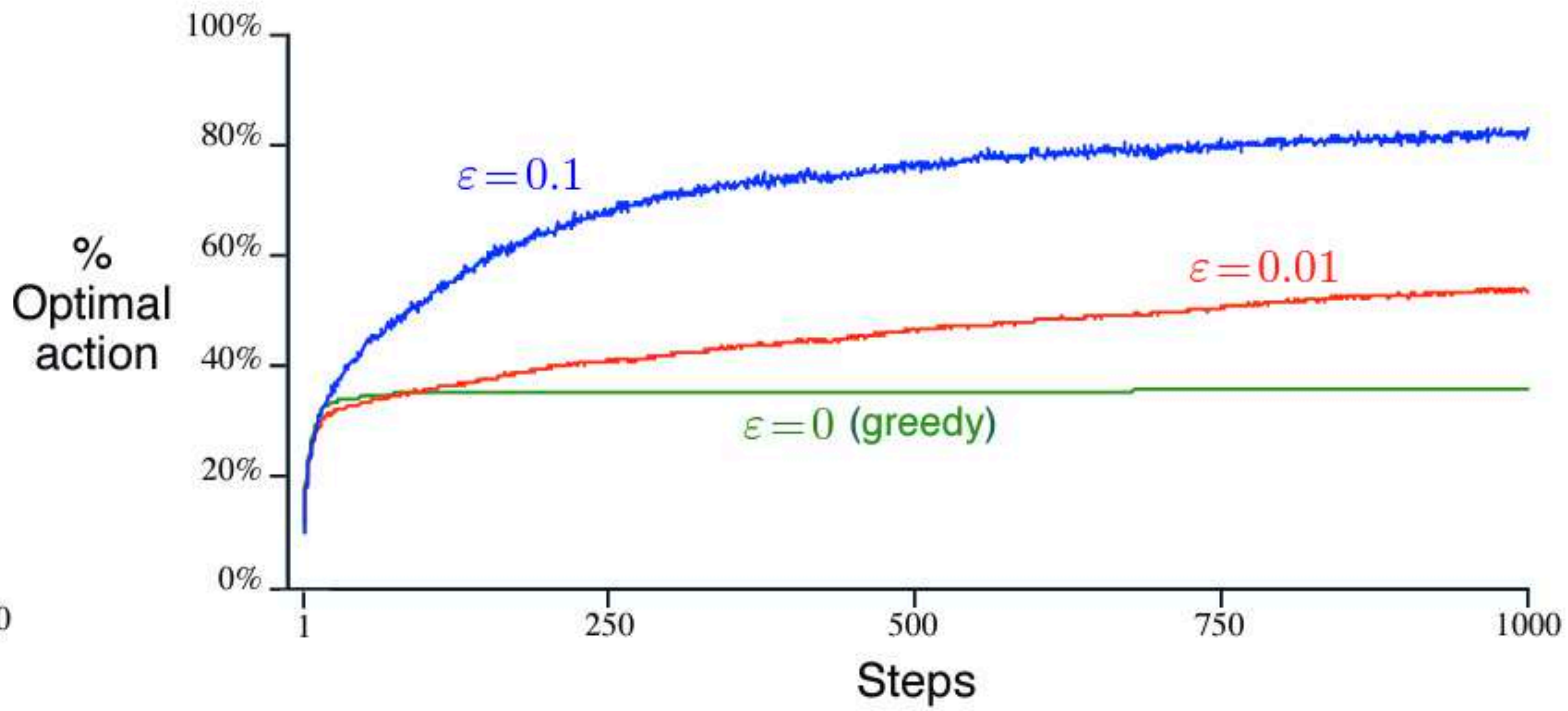
$$A_t \sim \pi_t(a) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{K}, & \text{if } a = \arg\max Q_t(a) \\ \frac{\varepsilon}{K}, & \text{otherwise} \end{cases}$$

  - Enhance exploration via setting $\varepsilon > 0$ to avoid "local maximizer"
  - $\pi_t(a)$ converges to a Dirac delta distribution center at $\arg\max_a Q_t(a)$ as $\varepsilon \to 0$
  - $\varepsilon \to 0$ when $t \to \infty$ to reduce redundant exploration

K-Armed Bandit Problem

Action Value Methods





By Richard S. Sutton and Andrew G. Barto.

K-Armed Bandit Problem

○ Action-value methods

- Direct estimation

$$Q_t(a) := \frac{\text{sum of rewards when a taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}}$$

- Incremental implementation for computational efficiency

Suppose we only have one action, then

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) = \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)Q_n\right) = \frac{1}{n}\left(R_n + nQ_n - Q_n\right) = Q_n + \frac{1}{n}\left[R_n - Q_n\right]$$

K-Armed Bandit Problem

○ Action-value methods

- Incremental implementation when we have only one action

$$Q_{n+1} = Q_n + \frac{1}{n}\left[R_n - Q_n\right]$$

- General rule true for other cases

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \left[Target - OldEstimate\right]$$

$$Q_t(a) = Q_t(a) + \frac{1}{N_t(a)}\left[R_t - Q_t(a)\right]$$

K-Armed Bandit Problem

○ Action-value methods

- Incremental update rule in general

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \left[Target - OldEstimate \right]$$

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$$R \leftarrow bandit(A)$$
$$N(A) \leftarrow N(A) + 1$$
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$$

By Richard S. Sutton and Andrew G. Barto.

K-Armed Bandit Problem

○ Action-value methods

- Incremental update rule for non-stationary reward probability with StepSize as a constant α

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \Big[Target - OldEstimate \Big]$$

- Resulting in a weighted average of past rewards emphasizing recent rewards

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i.
\end{aligned}$$

- More general, time dependent StepSize $\alpha_t(a)$

K-Armed Bandit Problem

Action-value methods

- ε-greedy selection: no preference for non-greedy actions

$$A_t \sim \pi_t(a) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{K}, & \text{if } a = \arg\max Q_t(a) \\ \frac{\varepsilon}{K}, & \text{otherwise} \end{cases}$$

- Upper-Confidence Bound selection: preference based on uncertainty

$$A_t = \arg\max \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

K-Armed Bandit Problem

Action-value methods

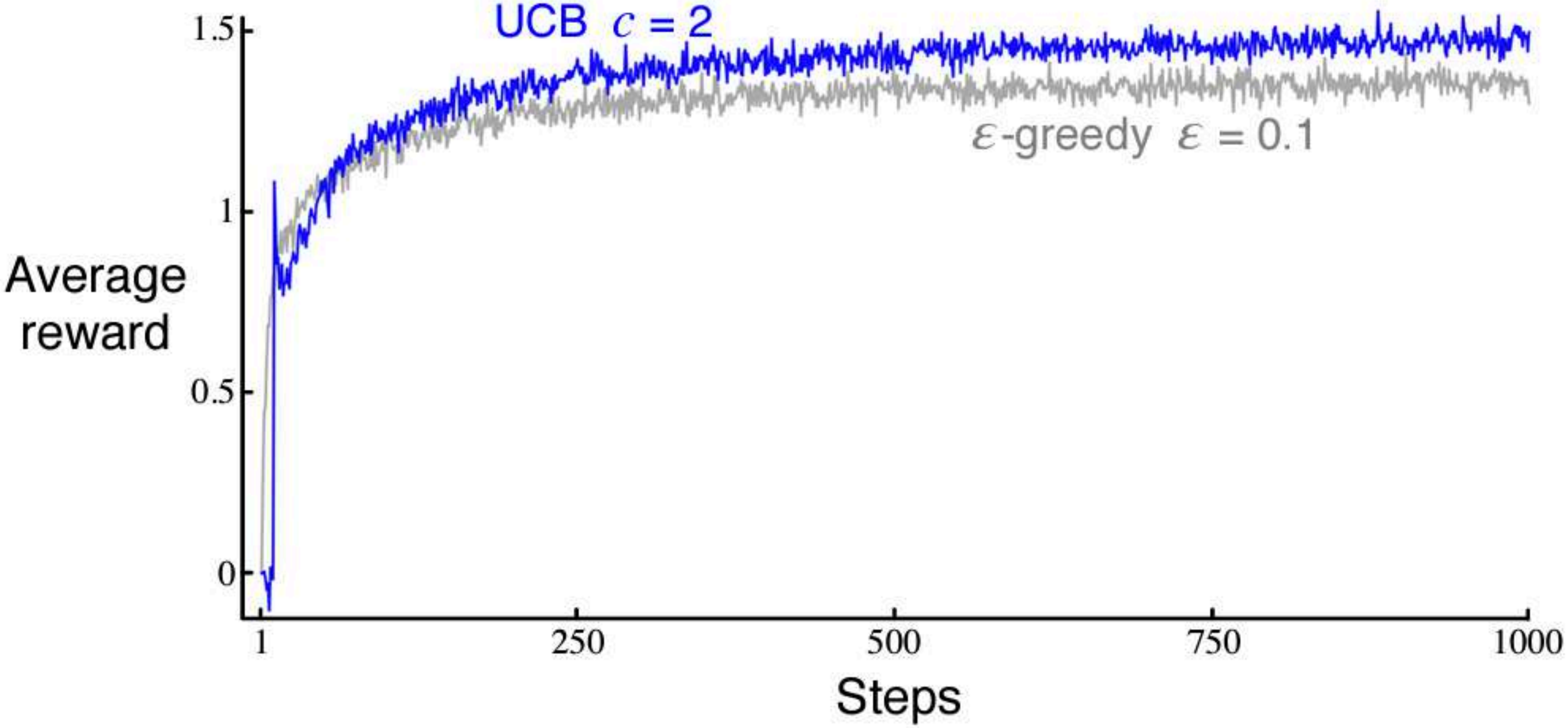
○ Upper-Confidence Bound selection: preference based on uncertainty

$$A_t = \arg\max \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

- Enhance exploration via adding $c\sqrt{\frac{\ln t}{N_t(a)}}$ to avoid "local maximize"
- Smaller $N_t(A_t)$, larger uncertainty on A_t, more preference to use $c\sqrt{\frac{\ln t}{N_t(A_t)}}$ to choose A_t
- Larger $N_t(A_t)$, smaller uncertainty on A_t, more preference to use $Q_t(A_t)$ to choose A_t
- $c\sqrt{\frac{\ln t}{N_t(a)}} \rightarrow 0$ when $t \rightarrow \infty$ to reduce redundant exploration

K-Armed Bandit Problem

ε-greedy selection v.s. Upper-Confidence Bound selection



By Richard S. Sutton and Andrew G. Barto.

K-Armed Bandit Problem

- **Action-value methods:**
  - Maximize the expected reward $\mathbb{E}[R_t \mid A_t]$ by choosing A_t appropriately
  - Construct $Q_t(a)$ converging to $q_*(a)$ to choose A_t
- **Policy-gradient methods:**
  - A probability distribution $\pi_t(a)$ as a policy to determine A_t without estimating Q_t
  - $\pi_*(a) := \delta_{\arg\max_{\bar{a}} q_*(\bar{a})}(a)$ is the optimal policy to determine A_t
  - Update $\pi_t(a)$ to approach $\pi_*(a)$
  - The updating rule is equivalent to stochastic gradient accent to maximize $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t \mid A_t]\right]$ as a functional of $\pi_t(a)$

K-Armed Bandit Problem

○ Policy-gradient methods

- Find $\pi_t(a) \to \pi_*(a)$ that maximizes $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t \,|\, A_t]\right]$
- Action selection and modeling of $\pi_t(a)$

$$\Pr\{A_t = a\} := \frac{e^{H_t(a)}}{\sum_{b=1}^{K} e^{H_t(b)}} := \pi_t(a)$$

where $H_t(a)$ is the preference for choosing $A_t = a$ at time t

- $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t \,|\, A_t]\right]$ is a functional of $H_t(a)$
- Find $H_t(a) \to H_*(a)$ that maximizes $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t \,|\, A_t]\right]$

K-Armed Bandit Problem

○ Policy-gradient methods

- Find $H_t(a) \to H_*(a)$ that maximizes $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t | A_t]\right]$
- Update $H_t(a)$ in a sense of the stochastic gradient accent for $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t | A_t]\right]$ via

$$H_{t+1}(A_t) := H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

  and

$$H_{t+1}(a) := H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) \quad \text{for } a \neq A_t$$

  where $\bar{R}_t$ is the average reward up to but not including time t.

- Then as $t \to \infty$,

$$H_t \to H_* \Leftrightarrow \pi_t \to \pi_*$$

  that maximizes $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t | A_t]\right]$

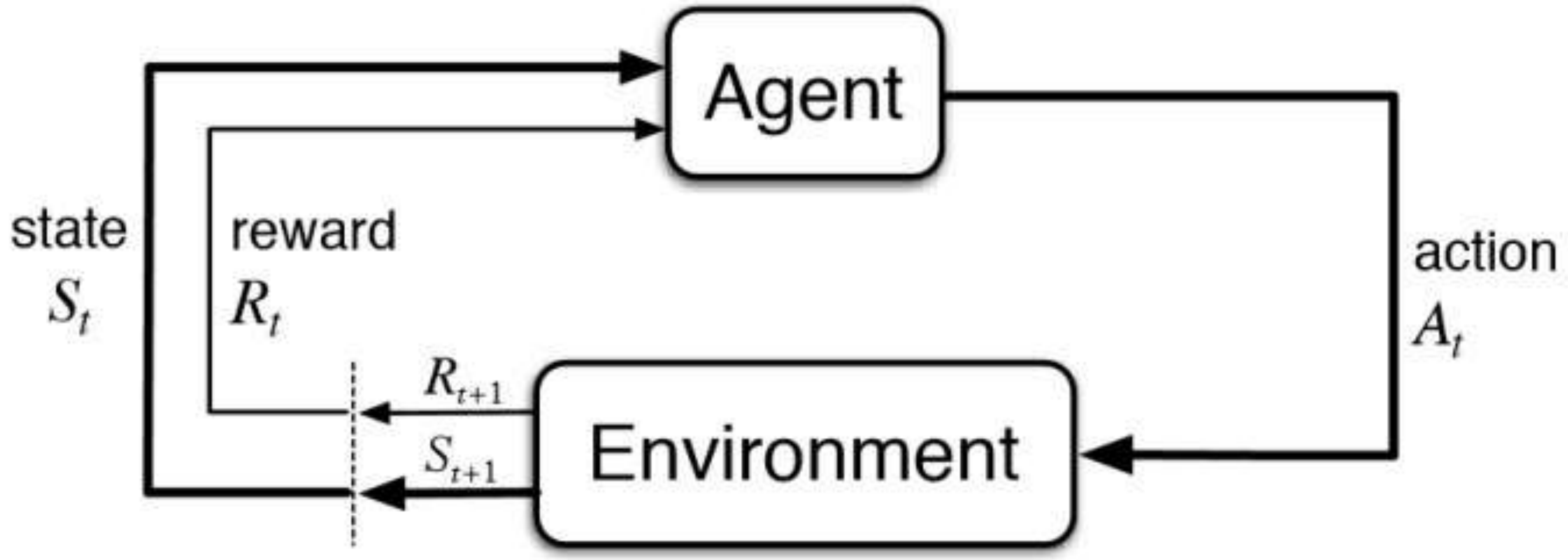
K-Armed Bandit Problem

Summary

- Setup: Single state and K actions
- Methods:
  - Value function method: find $Q_t(a) \rightarrow q_*(a)$ to maximize $\mathbb{E}[R_t | A_t]$ by choosing $A_t \rightarrow \arg\max_a q_*(a)$
  - Policy-gradient method: find $H_t \rightarrow H_* \Leftrightarrow \pi_t \rightarrow \pi_*$ that maximizes $\mathbb{E}_{A_t \sim \pi_t} \left[\mathbb{E}[R_t | A_t] \right]$
- Looking for $\pi_*(a) := \delta_{\arg\max_{\bar{a}} q_*(\bar{a})}(a)$ is equivalent to identifying $\arg\max_a q_*(a)$
- Initialization and hyper-parameters are important!

Finite Markov Decision Processes

Finite Markov Decision Process



By Richard S. Sutton and Andrew G. Barto.

- **Goal:** Learn how to take actions in order to maximize reward via iteraction
- **Components:** environment in a state $S_t \in \mathcal{S}$ returning a reward $R_t \in \mathcal{R}$, agent taking actions $A_t \in \mathcal{A}$
- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **Markov decision process with a probability distribution:**

$$p(s', r \,|\, s, a) := \mathbf{Pr}\{S_t = s', R_t = r \,|\, S_{t-1} = s, A_{t-1} = a\}$$

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **The probability distribution** p **defines the MDP and is the most basic unknown target:**

$$p(s', r \,|\, s, a) := \mathbf{Pr}\{S_t = s', R_t = r \,|\, S_{t-1} = s, A_{t-1} = a\}$$

- **State-transition probability:**

$$p(s' \,|\, s, a) := \mathbf{Pr}\{S_t = s' \,|\, S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \,|\, s, a)$$

- **Expected rewards for state-action pairs:**

$$r(s, a) := \mathbf{E}\{R_t \,|\, S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \,|\, s, a)$$

- **Expected rewards for state-action-next-state triples:**

$$r(s, a, s') := \mathbf{E}\{R_t \,|\, S_{t-1} = s, A_{t-1} = a, S_t = s'\} = \sum_{r \in \mathcal{R}} r \frac{p(s', r \,|\, s, a)}{p(s' \,|\, s, a)}$$

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **Goal:** maximize the expected discounted return

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1}$$

  - γ is a parameter in $[0,1]$ called the discount rate
  - R_t and R_{t+1} both depends on S_k for $k \leq t-1$
- **Application types:**

  1) episodes with a termination time T and a state state s_+

  2) continuing tasks without termination

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \dots$
- **Question:** how to choose actions to maximize the expected discounted return G_t?

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1}$$

- **What we have learned in the case of a single state?**
  - Value function method: find $Q_t(a) \to q_*(a)$ to maximize $\mathbb{E}[R_t \mid A_t]$ by choosing $A_t \to \arg\max_a q_*(a)$
  - Policy-gradient method: find $H_t \to H_* \Leftrightarrow \pi_t \to \pi_*$ that maximizes $\mathbb{E}_{A_t \sim \pi_t}\left[\mathbb{E}[R_t \mid A_t]\right]$
  - Consider $\mathbb{E}[R_t \mid A_t]$ instead of G_t because R_t's are independent when A_t's are given
- **For MDP, we can also use policy and value function**

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **Question:** how to choose actions to maximize the expected discounted return G_t?

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1}$$

- **Policy in MDP:** a conditional probability distribution $\pi(a \mid s)$ for $A_t = a$ and $S_t = s$
- **Value functions in MDP following the policy π**
  - Value function of states: $v_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right]$
  - Value function of state-action pair:

$$q_\pi(s, a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right]$$

Finite Markov Decision Process

- **Bellman equation for the policy π**

$$
\begin{aligned}
v_\pi(s) := {} & \mathbb{E}_\pi[G_t \mid S_t = s] \\
= {} & \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
= {} & \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\Big] \\
= {} & \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big]
\end{aligned}
$$

for all $s \in \mathcal{S}$

- $v_\pi(s)$ **is the unique solution (i.e., fixed point) of its Bellman equation**
- **Given π, a fixed point iteration of the Bellman equation returns** $v_\pi(s)$

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **Question:** how to choose actions to maximize the expected discounted return G_t?
- **Policy:** $\pi(a \mid s)$ and **Value functions following the policy** π**:** $v_\pi(s)$ and $q_\pi(s, a)$
- **How to compare policies?**

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

- **Any optimal policy** π_***?**
  - $\max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$ share the same optimizer π_*

    choose $\pi_*(a \mid s)$ to be the best probability for each fixed s

  - π_* may not be unique

Finite Markov Decision Process

- **Interaction:** $S_0, A_0, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$
- **Question:** how to choose actions to maximize the expected discounted return G_t?
- **Policy:** $\pi(a \mid s)$ and **Value functions following the policy π:** $v_\pi(s)$ and $q_\pi(s, a)$
- **How to compare policies?**

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

- **The optimal policy π_* share the same**
  - optimal state value function $v_*(s) := \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$
  - optimal state-action value function $q_*(s, a) := \max_\pi q_\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Finite Markov Decision Process

The optimal value functions

- $v_*(s) := \max_\pi v_\pi(s)$ and $q_*(s,a) := \max_\pi q_\pi(s,a)$ determine each other
- $q_*(s,a) := \max_\pi \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a]$

$$= \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a]$$

$$= \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) \,|\, S_t = s, A_t = a]$$

- Furthermore, $v_*(S_{t+1}) = \max_{a'} q_*(s', a')$
- Hence, $q_*(s,a) = \mathbb{E}_{\pi_*}[R_{t+1} + \gamma \max_{a'} q_*(s', a') \,|\, S_t = s, A_t = a]$

$$= \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

- The above is the Bellman equation of π_* for computing $q_*(s,a)$ via a fixed point iteration

Finite Markov Decision Process

The optimal value functions

- $v_*(s) := \max_\pi v_\pi(s)$ and $q_*(s,a) := \max_\pi q_\pi(s,a)$ determine each other
- $$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_{a \in \mathcal{A}(s)} \mathbb{E}_{\pi_*}[G_t \,|\, S_t = s, A_t = a] \\ &= \max_{a \in \mathcal{A}(s)} \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a] \\ &= \max_{a \in \mathcal{A}(s)} \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) \,|\, S_t = s, A_t = a] \\ &= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma v_*(s')] \end{aligned}$$
- The above is the Bellman equation of π_* for computing $v_*(s)$ via a fixed point iteration

Finite Markov Decision Process

Summary

- Introduced MDP
- Introduced policy
- Introduced value functions
- The optimal policy is associated with the optimal value functions
- Identify optimal value functions via fixed point iterations of the Bellman equation
- Derive ε-greedy policy using value functions,

  e.g., $A_t = \begin{cases} \arg\max_a q_*(S_t, a) & \text{with probability } 1 - \varepsilon \\ \text{uniform random action} & \text{with probability } \varepsilon \end{cases}$
- Model-free method

Dynamic Programming

Dynamic Programming

Overview

- Model-based method: $p(s', r \mid s, a)$ is known
- Expense computation
- Interesting theoretically
- Help us to understand some model-free methods
- Assume termination state s_+ and let $\mathcal{S}^+ := s_+ \cup \mathcal{S}$
- The environment is assumed to be a finite MDP for simplicity here

Dynamic Programming

Computation motivation: optimal value function evaluation

- Model-based method: $p(s', r \mid s, a)$ is known
- The optimal value function can derive the optimal policy
- The optimal value function satisfies the Bellman optimality equation:

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned}$$

or

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r \mid s, a)[r + \gamma \max_{a'} q_*(s', a')] \end{aligned}$$

for all $s \in \mathcal{S},\ a \in \mathcal{A}(s),$ and $s' \in \mathcal{S}^+$

- Use the above equations to derive iterative methods to compute the optimal value functions

Dynamic Programming

Computation Step 1: policy evaluation

- Model-based method: $p(s', r \mid s, a)$ is known
- Goal: given a policy π, evaluate its value in v_π
- Observation:

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &:= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \end{aligned}$$

- Iterative procedure for the fixed point v_π,; then $v_k \to v_\pi$

$$\begin{aligned} v_{k+1}(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')] \end{aligned}$$

Dynamic Programming

Computation Step 1: policy evaluation

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

Computation Step 2: policy improvement

- Goal: improve a policy π based on its value in v_π
- Observation: π' below is better than or as good as π

$$
\begin{aligned}
\pi'(s) &:= \arg\max_a q_\pi(s, a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s', r} p(s', r \mid s, a)[r + \gamma v_\pi(s')]
\end{aligned}
$$

- Iterative procedure: use π' as a new improved policy
- When π' is as good as π, then $\pi' = \pi = \pi_*$, because v_* is the unique fixed point of the Bellman equation

Dynamic Programming

Complete algorithm

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

Complete algorithm

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \underline{\sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]}$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

Greedy evaluation: $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \underline{\arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]}$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Greedy policy: $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$

By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

Complete algorithm: fixed point iteration via the Bellman optimality equation

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

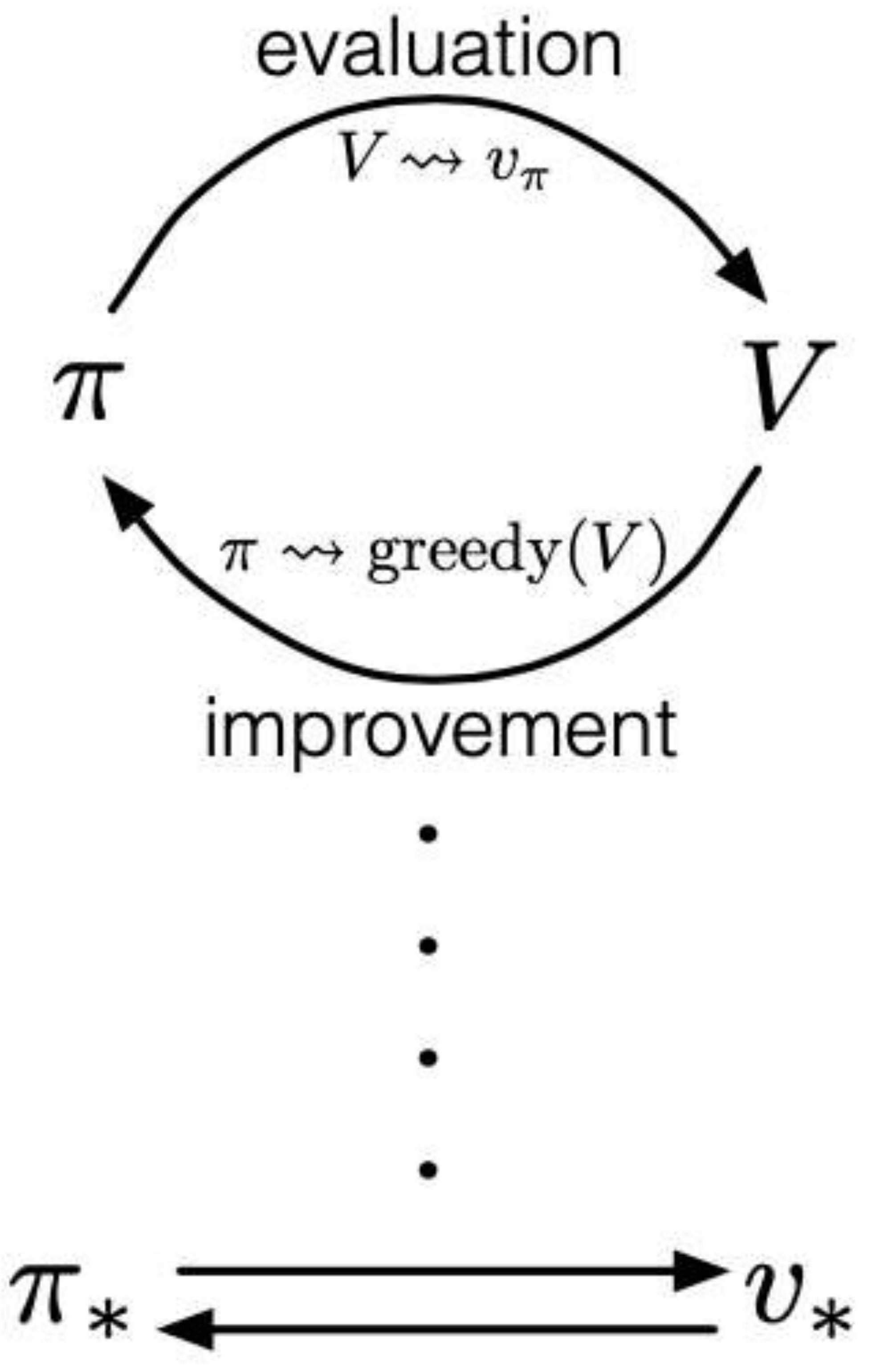
Generalized Policy Iteration

Observation

- Policy iteration for the optimal policy — two simultaneous, interacting processes; each process takes many iterations
- Value iteration for the optimal policy — two simultaneous, interacting processes; each process takes one step

Question

- Can we make it more flexible?



By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

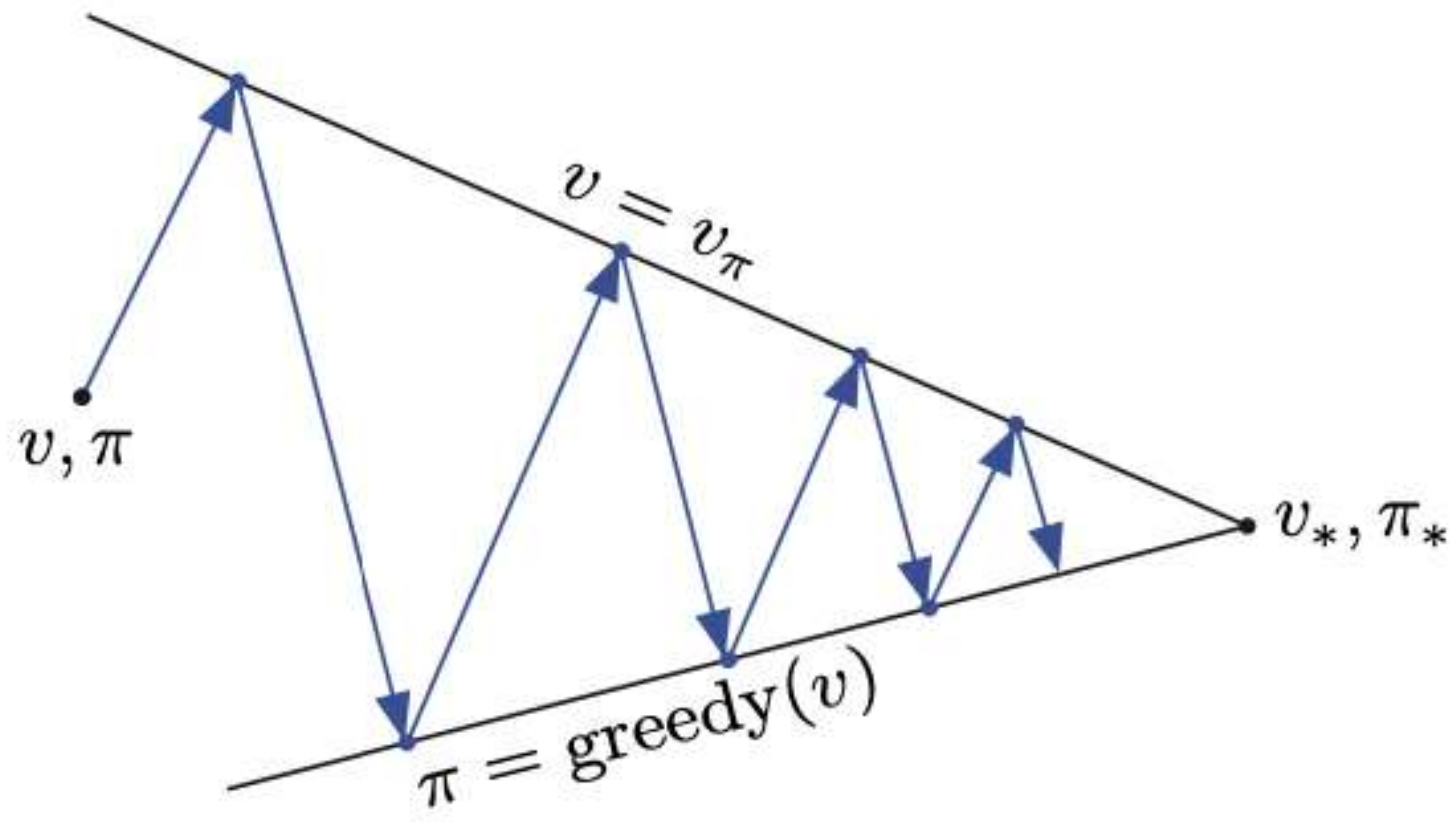
Generalized Policy Iteration

Question

- Can we make it more flexible?

Answer

- Yes
- Randomly or alternatively choose a subset of states to carry out these two processes
- $O(1)$ steps in each process
- This is called the generalized policy iteration (GPI)



By Richard S. Sutton and Andrew G. Barto.

Dynamic Programming

Summary

- Model-based method
- Introduced policy iteration method to compute the optimal policy
- Introduced value iteration method to compute the optimal policy
- Generalized policy iteration (GPI)
- Expensive computation

Monte Carlo Methods

Monte Carlo Methods

Overview

- Model-free method: $p(s', r \mid s, a)$ is unknown
- Assume samples from $p(s', r \mid s, a)$ are available
- Cheaper computation
- Offline episode-by-episode update instead of online step-by-step update
- Interesting in real applications
- The environment is assumed to be a finite MDP for simplicity here

Monte Carlo Methods

- Basic idea: $\mathbb{E}[f(x)] \approx E_n := \frac{1}{n}\sum_i f(x_i)$
- Law of large number: $E_n \to \mathbb{E}[f(x)]$ as $n \to \infty$
- Incremental implementation: $E_{n+1} = \frac{nE_n + f(x_{n+1})}{n+1}$

Monte Carlo Methods

Algorithms to be discussed

- Policy value evaluation
  - State value function $v_\pi(s)$ — can help to determine a good policy if model is known, e.g., model-based method DP
  - State action value function $q_\pi(s, a)$ — directly help to determine policies without a model, e.g., model-free method MC
- Policy improvement

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

$G \leftarrow 0$

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 $Returns(S_t)$

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

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

Every visit MC for policy evaluation: average of the returns following every visit to s

Every

~~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 $Returns(S_t)$

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

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

First visit MC for state-action evaluation: average of the returns following first visits to (s,a)

First-visit MC prediction, for estimating ~~$V \approx v_\pi$~~ $q(s,a) \approx q_\pi(s,a)$

Input: a policy π to be evaluated

Initialize:

~~$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$~~ $q(s,a) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

~~$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$~~ $Returns(s,a) \leftarrow$ an empty list, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

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 $S_0, S_1, \ldots, S_{t-1}$:~~ Unless (S_t, A_t) appears in $(S_0, A_0), \ldots, (S_{t-1}, A_{t-1})$

~~Append G to $Returns(S_t)$~~ Append G to $Returns(S_t, A_t)$

~~$V(S_t) \leftarrow$ average$(Returns(S_t))$~~ $q(S_t, A_t) \leftarrow average(Returns(S_t, A_t))$

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

Every visit MC for state-action evaluation: average of the returns following every visit to (s,a)

Every

~~First~~-visit MC prediction, for estimating $\cancel{V \approx v_\pi}$ $q(s,a) \approx q_\pi(s,a)$

Input: a policy π to be evaluated

Initialize:

$\cancel{V(s) \in \mathbb{R}}$, ~~arbitrarily, for all~~ $\cancel{s \in \mathcal{S}}$ $\quad q(s,a) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

$\cancel{Returns(s) \leftarrow}$ ~~an empty list, for all~~ $\cancel{s \in \mathcal{S}}$ $\quad Returns(s,a) \leftarrow$ an empty list, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

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~~ $\cancel{S_t}$ ~~appears in~~ $\cancel{S_0, S_1, \ldots, S_{t-1}}$:

~~Append~~ $\cancel{G}$ ~~to~~ $\cancel{Returns(S_t)}$ $\quad$ Append G to $Returns(S_t, A_t)$

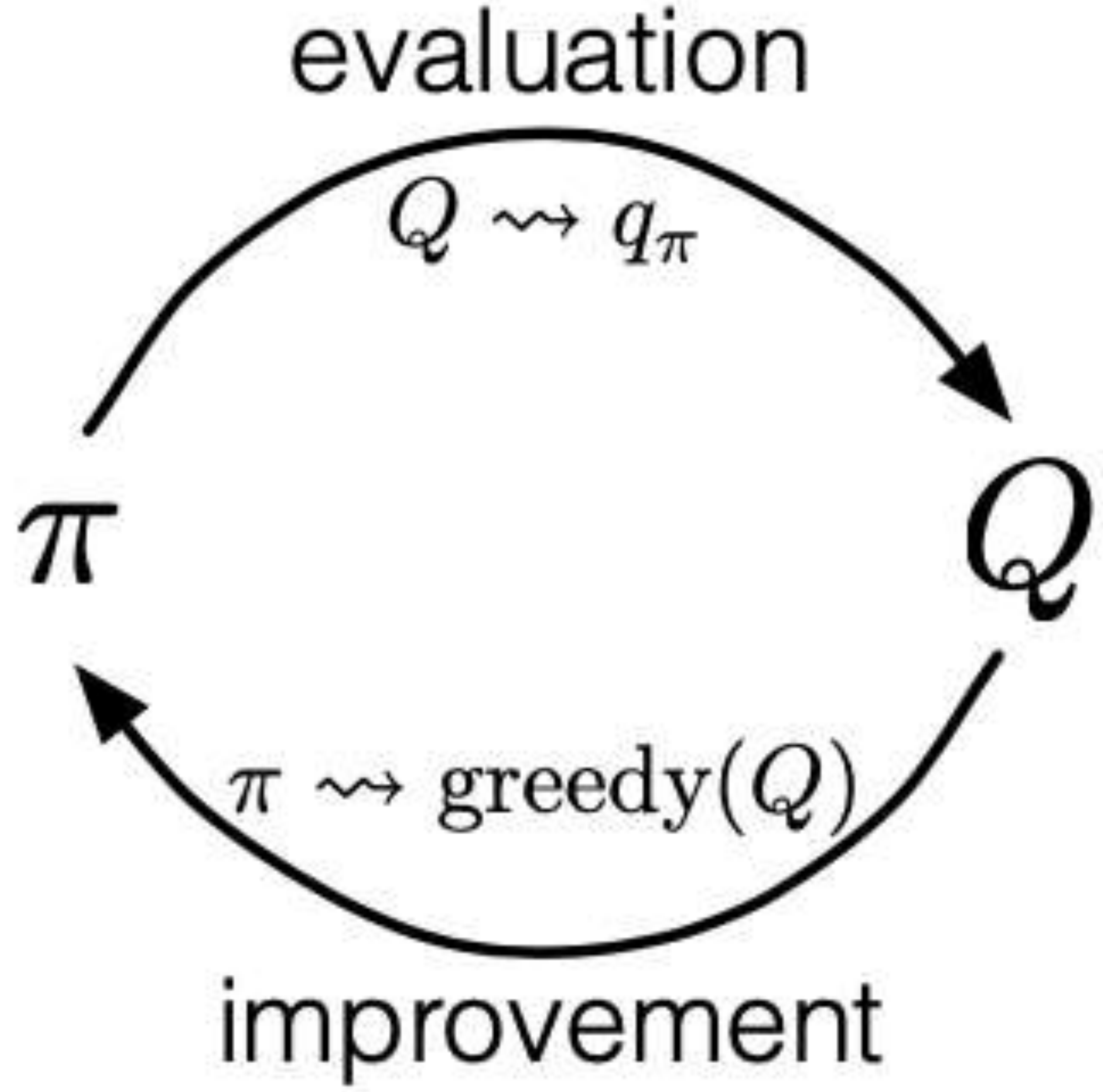
$\cancel{V(S_t) \leftarrow \text{average}(Returns(S_t))}$ $\quad q(S_t, A_t) \leftarrow average(Returns(S_t, A_t))$

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

Generalized Policy Iteration (GPI): Expensive

Assumptions: 1) Exploring start; 2) Infinitely many episodes



By Richard S. Sutton and Andrew G. Barto.

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

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

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

Monte Carlo Methods

Generalized Policy Iteration (GPI) : Cheap

Assumptions: 1) Exploring start; 2) Infinitely many episodes

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 $Returns(S_t, A_t)$

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

$\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

How to avoid exploring starts?

Two situations

- *On-policy* method: attempt to evaluate or improve the policy that is used to make decisions
  - e.g., all methods that we have learned so far
  - Change greedy policy $\pi(s)$ to ε-greedy policy $\pi(a \mid s) > 0$ for all s and a
  - One policy has to balance exploration (not greedy) and exploitation (greedy)
- *Off-policy* method: evaluate or improve a policy different from that used to generate the data

Monte Carlo Methods

How to avoid exploring starts (on-policy)?

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 $Returns(S_t, A_t)$

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

      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

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

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

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

How to avoid exploring starts?

Two situations

- *On-policy* method: attempt to evaluate or improve the policy that is used to make decisions
  - One policy has to balance exploration (not greedy) and exploitation (greedy)
- *Off-policy* method: evaluate or improve a policy different from that used to generate the data
  - Use two policies
  - *Behavior policy* encourages exploration and generate samples, e.g., stochastic policy $\pi_b(a \mid s) > 0$ for all s and a
  - *Target policy* encourages exploitation and approaches to the optimal policy, e.g., deterministic policy $\pi(s)$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Off-policy method:

- Algorithm: use behavior policy $\pi_b(a \mid s)$ (non-greedy) to generate data to estimate target policy $\pi(s)$ (greedy)
- We cannot completely believe data and need to process them !

.

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Question: How can we process the data?

Answer: Important sampling

- A general technique for estimating expected values under one distribution given samples from another
- e.g., estimating the mean of a Bernoulli random variable X_p from the samples of another Bernoulli random variable X_q
- $\Pr(X_p = 1) = p = 1 - \Pr(X_p = 0)$ and $\mathrm{E}[X_p] = (1 - p) * 0 + p * 1 = p$
- $\Pr(X_q = 1) = q = 1 - \Pr(X_q = 0)$ and $\mathrm{E}[X_q] = q$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Important sampling

- Goal: estimate $\mathrm{E}[X_p]$ using the samples of X_q: $\{x_i\}_{i=1}^n$
- Law of large number: $\mathrm{E}[X_q] \approx \frac{1}{n}\sum_{i=1}^{n} x_i$
- e.g., there should be about nq x_i's as 1, and $(1-q)n$ x_i's as 0
- How can we make the average of these samples look like the average of the samples of X_p?
- If we had the samples of X_p, there should be about np numbers as 1, and $(1-p)n$ numbers as 0
- Hence, we modify $\{x_i\}_{i=1}^n$ to obtain $\{\bar{x}_i\}_{i=1}^n$ as $\bar{x}_i = \begin{cases} \frac{p}{q}x_i, \text{ if } x_i = 1 \\ \frac{1-p}{1-q}x_i \text{ if } x_i = 0 \end{cases}$
- Finally, $\mathrm{E}[X_p] \approx \frac{1}{n}\sum_{i=1}^{n} \bar{x}_i =$ the average of nq samples of $1 * \frac{p}{q}$ and $n(1-q)$ samples of $0 * \frac{1-p}{1-q} \approx p = \mathrm{E}[X_p]$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Important sampling

- Goal: estimate $\mathrm{E}[X_p]$ using the samples of X_q: $\{x_i\}_{i=1}^n$
- How can we make the average of these samples look like the average of the samples of X_p?
- Hence, we modify $\{x_i\}_{i=1}^n$ to obtain $\{\bar{x}_i\}_{i=1}^n$ as $\bar{x}_i = \dfrac{\Pr[X_p = x_i]}{\Pr[X_q = x_i]} x_i$
- Then $\mathrm{E}[X_p] \approx \dfrac{1}{n} \sum_{i=1}^{n} \bar{x}_i$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Off-policy method:

- Algorithm: use behavior policy $\pi_b(a \mid s)$ (non-greedy) to generate data to estimate target policy $\pi(s)$ (greedy)
- Use important sampling to process the data
- $v_{\pi_b}(s) = \mathrm{E}_{\pi_b}[G_t \mid S_t = s]$
- $v_\pi(s) = \mathrm{E}_\pi[G_t \mid S_t = s] = \mathrm{E}_{\pi_b}[\rho_{t:T-1} G_t \mid S_t = s]$ with T as the termination time and

$$\rho_{t:T-1} = \Pi_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{\pi_b(A_k \mid S_k)} = \frac{\Pr[A_t, S_{t+1}, A_{t+1}, \ldots, S_T] \text{ under } \pi}{\Pr[A_t, S_{t+1}, A_{t+1}, \ldots, S_T] \text{ under } \pi_b}$$

$$= \frac{\Pr[G_t] \text{ under } \pi}{\Pr[G_t] \text{ under } \pi_b}$$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

Off-policy method:

- Most steps are the same as the on-policy methods discussed previously
- Use behavior policy $\pi_b(a \mid s)$ (non-greedy) to generate data to estimate target policy $\pi(s)$ (greedy)
- Use the formula $v_\pi(s) = \mathrm{E}_{\pi_b}[\rho_{t:T-1} G_t \mid S_t = s]$ with $\rho_{t:T-1} = \Pi_{k=t}^{T-1} \dfrac{\pi(A_k \mid S_k)}{\pi_b(A_k \mid S_k)}$ to evaluate and improve the policy $\pi(s)$
- Similarly, $q_\pi(s, a) := \mathbb{E}_{\pi_b}[\rho_{t+1:T-1} G_t \mid S_t = s, A_t = a]$

Monte Carlo Methods

How to avoid exploring starts (off-policy)?

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)
        $W \leftarrow W \frac{1}{b(A_t|S_t)}$

By Richard S. Sutton and Andrew G. Barto.

Monte Carlo Methods

Summary

- Model-free method
- Introduced policy iteration method to compute the optimal policy
- Introduced value iteration method to compute the optimal policy
- Generalized policy iteration (GPI)
- Introduced on-policy and off-policy for exploring starts (without discounted rewards)
- Exploring starts with discounted rewards not discussed

Temporal-Difference Learning

Temporal-Difference (TD) Learning

Overview

- One central and novel idea of RL
- Combining DP and MC
- Learn from raw data without model: model-free
- Update estimates using learned models: model-based
- Policy evaluation and improvement based on generalized policy iteration (GPI)

Temporal-Difference (TD) Learning

Main idea

- $$\begin{aligned} v_\pi(s) :&= \mathrm{E}_\pi[G_t \mid S_t = s] \\ &= \mathrm{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathrm{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \end{aligned}$$
- MC update:

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

  G_t is available only when an episode ends

- TD update:

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

  - MC: R_{t+1} is available right at the next time step, hence, online, fully incremental, and no need for discounts
  - DP: $V(S_{t+1})$ is the currently learned model

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 $V(terminal) = 0$

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.

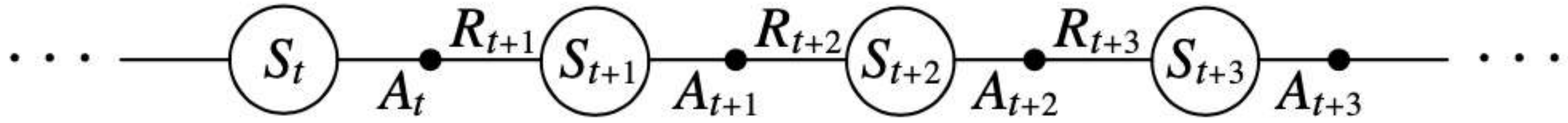
Temporal-Difference (TD) Learning

Main idea

- MC methods:
  - on-policy updates using one ε-greedy policy
  - off-policy updates using important sampling with two policies
- TD update:
  - on-policy updates for policies: Sarsa with one policy
  - off-policy updates for policies: Q-learning with two policies

Temporal-Difference (TD) Learning

Sarsa: on-policy updates



- Value function update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Use ε-greedy policy with $\varepsilon = \dfrac{1}{t}$
- Converges with probability 1 to q_* if all (s, a)-pairs are visited an infinite number of times

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 $Q(\mathit{terminal}, \cdot) = 0$

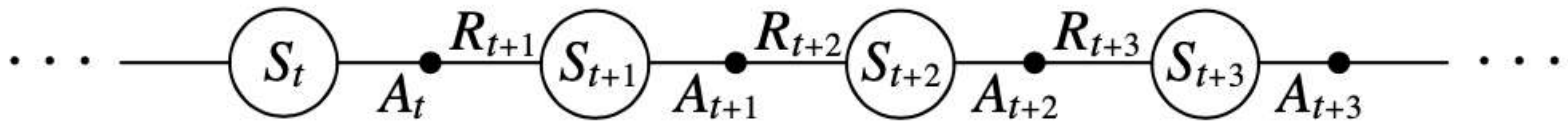
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$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

results of the same policy

By Richard S. Sutton and Andrew G. Barto.

Temporal-Difference (TD) Learning

Q-learning: off-policy updates



- Value function update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

- Use one ε-greedy policy (prefer exploration) to generate data
- Use another greedy policy (prefer exploitation) to update value function
- Converges with probability 1 to q_* since all (s, a)-pairs are visited an infinite number of times

Temporal-Difference Learning

$Q \approx q_*$

Q-learning (off-policy TD control) for estimating ~~$\pi \approx \pi_*$~~

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 $Q(terminal, \cdot) = 0$

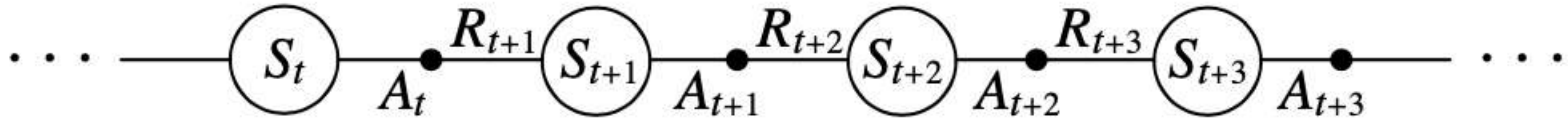
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
 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

results of different policies

By Richard S. Sutton and Andrew G. Barto.

Temporal-Difference (TD) Learning

Expected Sarsa: on-policy updates



- Value function update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathrm{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)]$$

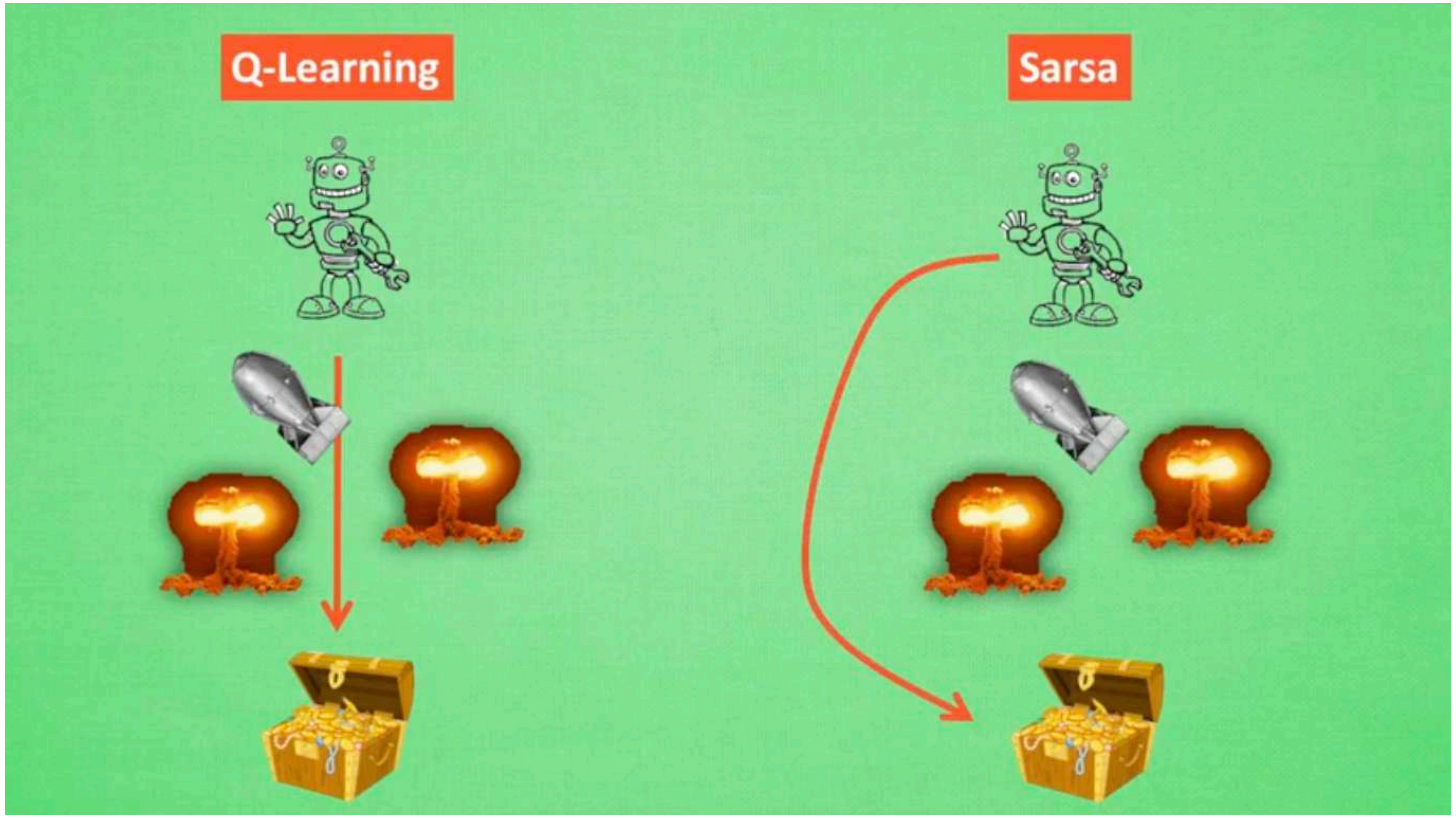
$$= Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)]$$

- Sarsa: use ε-greedy policy π to choose A_{t+1} randomly given S_{t+1}
- Expected Sarsa: use the expectation to eliminate the variance of the randomness

Temporal-Difference (TD) Learning

Comparison

- Sarsa — more conservative
- Q-learning — more aggressive



By Mofanpy

Temporal-Difference (TD) Learning

Comparison

- Expected Sarsa v.s. Sarsa — more stable, working for a wide range of step size α, but more expensive
- Expected Sarsa v.s. Q-learning — more conservative

Temporal-Difference (TD) Learning

Maximization Bias: bias towards positive value function estimation

- $\max_{a \in \mathscr{A}(s)} Q(s, a)$ is used in TD learning
- Example:
  - A single state s with many actions
  - $q_*(s, a) = 0$ for most a's
  - $Q(s, a) \neq 0$ for most a
  - $\max_{a \in \mathscr{A}(s)} Q(s, a)$ has a bias towards actions such that $Q(s, a) > 0$

Temporal-Difference (TD) Learning

How to avoid the maximum bias?

- Idea: Use two policies and alternatively update
- Use $A^* = \arg\max_a Q_1(a)$ to take an action
- Use $Q_2(A^*) = Q_2(\arg\max_a Q_1(a))$ to provide model estimate to update Q_1
- Example:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)]$$

- Similarly, in the next round, use Q_2 to take action to update Q_1

Temporal-Difference (TD) Learning

How to avoid the maximum bias?

- Idea: Use two policies and alternatively update
- Use $A* = \arg\max_a Q_1(a)$ to take an action
- Use $Q_2(A*) = Q_2(\arg\max_a Q_1(a))$ to provide model estimate to update Q_1
- $\arg\max_a Q_1(a)$ has a bias towards actions with positive values in Q_1
- $\arg\max_a Q_1(a)$ has no bias towards actions with positive values in Q_2
- If Q_1 and Q_2 have no bias, then the new Q_2 has no bias
- Similarly, if Q_1 and Q_2 have no bias, then the new Q_1 has no bias

Temporal-Difference (TD) Learning

Double Q-learning to avoid the maximum bias

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
        Take action A, observe R, S'
        With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

        else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

        $S \leftarrow S'$
    until S is terminal

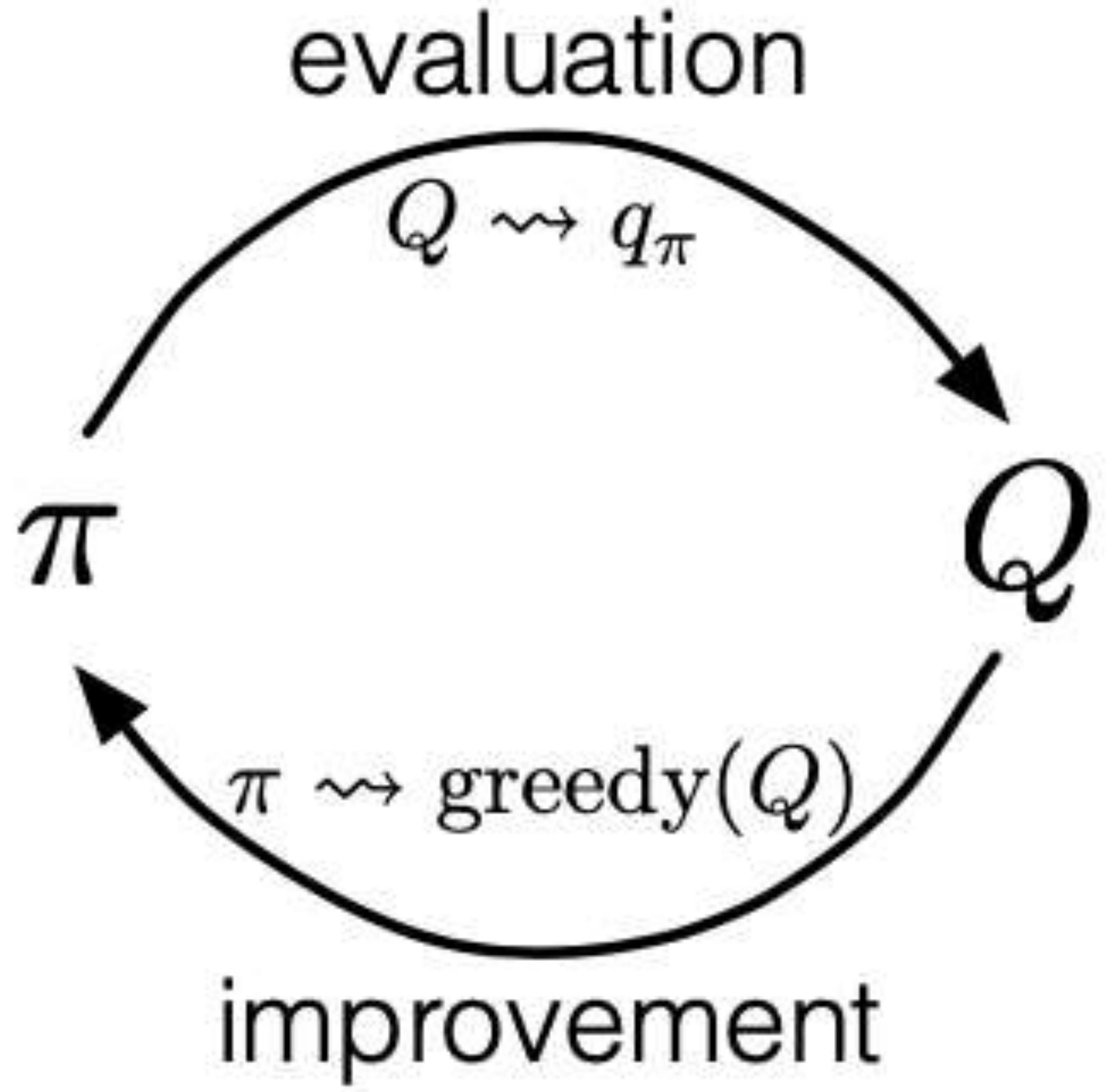
By Richard S. Sutton and Andrew G. Barto.

Temporal-Difference (TD) Learning

Generalized Policy Iteration (GPI): Expensive

On-policy (expected) Sarsa with ε-greedy policies to avoid exploring starts;

Off-policy (double) Q-Leaning with ε-greedy policies to avoid exploring starts;



By Richard S. Sutton and Andrew G. Barto.

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

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

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

Temporal-Difference (TD) Learning

Summary

- Combining model-free and model-based
- Generalized policy iteration (GPI) as in MC and DP
- Introduced value iteration method to compute the optimal policy: Sarsa and Q-learning
- Introduced off-policy double Q-learning to avoid the maximum bias
- Can use on-policy and off-policy for exploring starts as in MC