Lecture 8: Model-Free RL Part II

Ziyu Shao

School of Information Science and Technology
ShanghaiTech University

April 23, 2025

Outline

Outline

Model-Free Reinforcement Learning

- Last lecture:
  - Model-free prediction
  - *Estimate* the value function of an *unknown* MDP
- This lecture:
  - Model-free control
  - *Optimize* the value function of an *unknown* MDP

Uses of Model-Free Control

Some example problems that can be modeled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

Model-free control can solve these problems

On & Off-Policy Learning

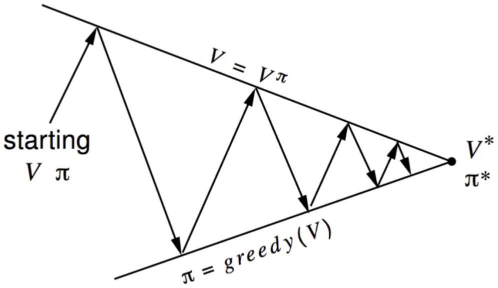
- Behavior-policy: determines which action to take, from which we determine the next state to visit (also called "sampling policy")
- Target-policy: determines the action that appears to be best (also called "learning policy")
- Goal in reinforcement learning:
  - improve the target policy
  - while using a behavior policy to ensure that we visit states often enough (exploration schemes such as ϵ-greedy)
- On-policy learning: when the learning policy and the sampling policy are the same
- Off-policy learning: when the learning policy and the sampling policy are different

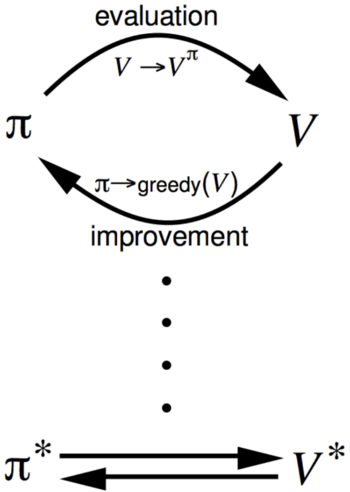
On & Off-Policy Learning

- On-policy learning
  - "Learn on the job"
  - Learn the value of the target policy π from experience sampled from behavior policy π
  - May not ensure the enough exploration of state space
- Off-policy learning
  - "Look over someone's shoulder"
  - Learn the value of the target policy π from experience sampled from behavior policy μ
  - Learning is from experience(data) "off" the target policy
  - Compared to on-policy learning, off-policy learning is
    - ⋆ more powerful & general
    - ⋆ often of greater variance & slower to converge

Outline

Generalized Policy Iteration (Refresher)

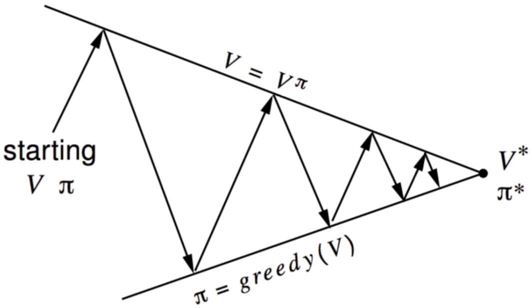




Policy evaluation Estimate v_π
e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
e.g. Greedy policy improvement

Generalized Policy Iteration With Monte-Carlo Evaluation



Policy evaluation Monte-Carlo policy evaluation, $V = v_\pi$?
Policy improvement Greedy policy improvement?

Model-Free Policy Iteration Using Action-Value Function

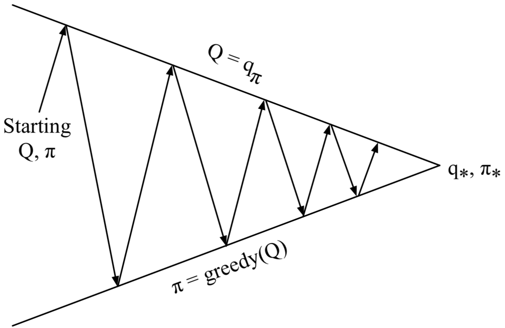
- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \{\mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')\}$$

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

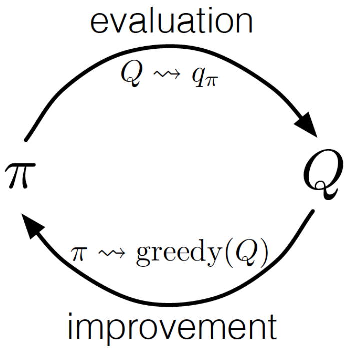
$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \; Q(s, a)$$

Generalized Policy Iteration with Action-Value Function



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$
Policy improvement Greedy policy improvement?

Monte-Carlo Control



- MC policy iteration: policy evaluation using MC methods followed by policy improvement
- Policy improvement: greedify with respect to value (or action-value) function

Example of Greedy Action Selection

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward $+1$
  $V(right) = +1$
- You open the right door and get reward $+3$
  $V(right) = +3$
- You open the right door and get reward $+2$
  $V(right) = +2$

$$\vdots$$

- Are you sure you've chosen the best door?

ϵ-Greedy Exploration

- Tradeoff between exploitation and exploration
- All m actions are tried with non-zero probability
- With probability $1-\epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\arg\max}\, Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

ϵ-Greedy Policy Improvement

Theorem

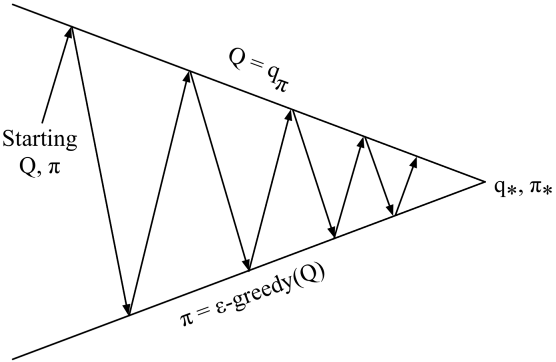
For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\ &= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1-\epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\ &\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1-\epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s) \end{aligned}$$

Therefore from policy improvement theorem, $v_{\pi'}(s) \geq v_\pi(s)$

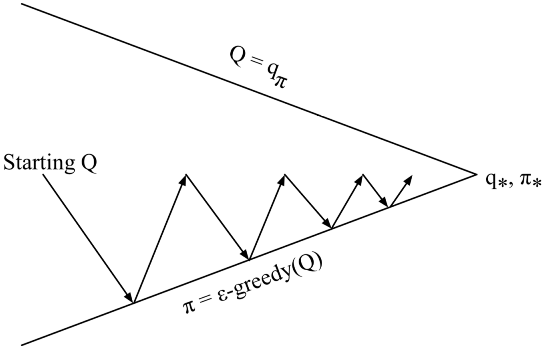
Remark

Monte-Carlo Policy Iteration



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$
Policy improvement ϵ-greedy policy improvement

Monte-Carlo Control



Every episode:
Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$
Policy improvement ϵ-greedy policy improvement

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}(a = \arg\max_{a' \in \mathcal{A}} Q_k(s, a'))$$

GLIE

- For example, ϵ-greedy is GLIE if ϵ reduces to zero at $\epsilon_k = \frac{1}{k}$
- Keep a declining exploration probability at a sufficiently slow rate that try all actions infinitely ofen
- In practice, $\epsilon_k = \frac{1}{k^\beta}$ with $\beta \in (0.5, 1]$

GLIE Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, \ldots, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$

On-Policy First-Visit MC Control

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\quad \pi \leftarrow$ an arbitrary ε-soft policy

$\quad Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

$\quad$ Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$\quad G \leftarrow 0$

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

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

$\quad\quad$ Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

$\quad\quad\quad$ Append G to $Returns(S_t, A_t)$

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

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

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

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

Partial Summary of MC

- MC has several advantages over DP
  - can learn directly from interaction with environment
  - No need for full models
  - No need to learn about ALL states (no bootstrapping)
  - Less harmed by violating Markov property
- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration
  - exploring starts, soft policies

Boltzmann Exploration

- A limitation of ϵ-greedy: when we explore, we choose actions at random without regard to their estimated values
- For larger action spaces, this can mean that we are spending a lot of time evaluating actions that are quite poor
- Boltzmann Exploration (also known as "Gibbs sampling" & "soft-max"): choosing an action based on its estimated value.

$$\pi(a|s) = \frac{e^{\beta \hat{Q}(s,a)}}{\sum_{a'} e^{\beta \hat{Q}(s,a')}}$$

- $\hat{Q}(s, a)$ is an estimate of the value of being in state s and taking action a.
- β is a tunable parameter
  - $\beta = 0$ produces a pure exploration policy
  - $\beta \to \infty$ produces a greedy policy

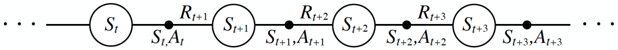
Outline

MC vs. TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S, A)$
  - Use ϵ-greedy policy improvement
  - Update every time-step

Learning An Action-Value Function

Estimate q_π for the current policy π

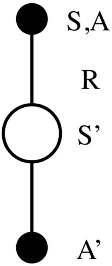


After every transition from a nonterminal state, S_t, do this:

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

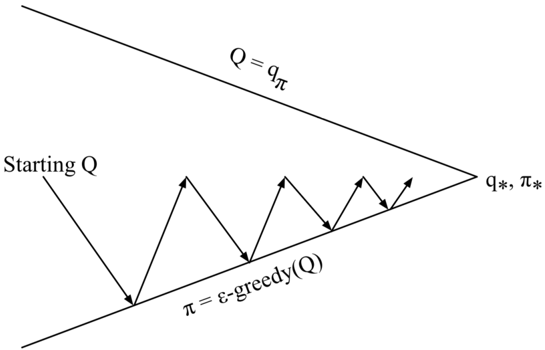
If S_{t+1} is terminal, then define $Q(S_{t+1}, A_{t+1}) = 0$

Updating Action-Value Functions with Sarsa



$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

On-Policy Control With Sarsa



Every time-step:

Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Sarsa Algorithm for On-Policy Control

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

Convergence of Sarsa

Theorem

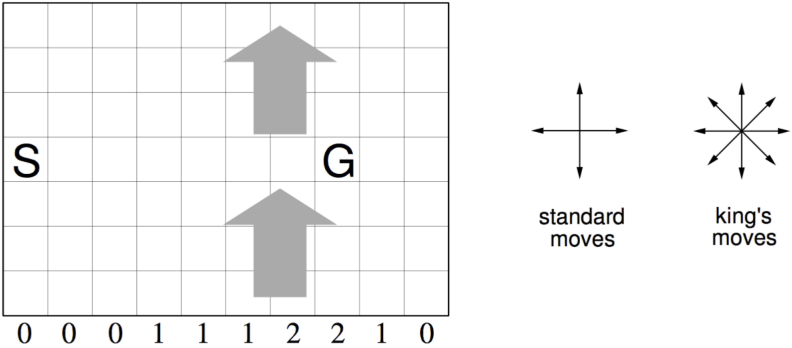
Sarsa converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$, under the following conditions:

- *GLIE sequence of policies $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes α_t*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

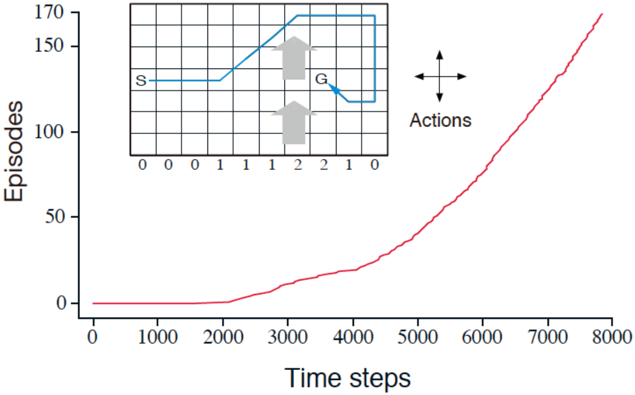
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Windy Gridworld Example



- Reward = -1 per time-step until reaching goal
- Undiscounted

Sarsa on the Windy Gridworld



n-Step Sarsa

- Consider the following n-step returns for $n = 1, 2, \infty$:

$$\begin{array}{lll} n = 1 & (Sarsa) & q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\ n = 2 & & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\ \vdots & & \vdots \\ n = \infty & (MC) & q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T \end{array}$$

- Define the n-step Q-return

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n})$$

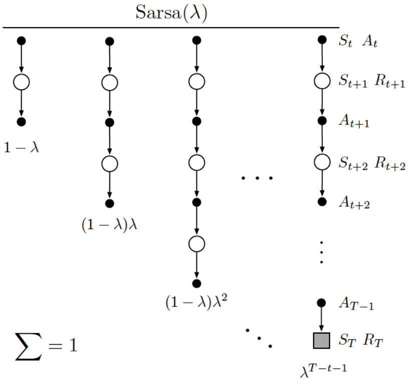
- n-step Sarsa updates $Q(s, a)$ towards the n-step Q-return

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right)$$

Forward View Sarsa(λ)

- Forward-view Sarsa(λ)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{\lambda} - Q(S_t, A_t) \right)$$



- The q^{λ} *return* combines all n-step Q-returns $q_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

$$q_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} q_t^{(n)}$$

Outline

Off-Policy Learning

- Learn the value of the target policy π from experience due to behavior policy μ
- In this sense, learning is from experience(data) "off" the target policy, and the overall process is termed off-policy learning.
- For example, π is the greedy policy (and ultimately the optimal policy) while μ is exploratory (e.g.,ϵ-soft)
- In general, we only require coverage, i.e., that μ generates behavior that covers, or includes, π

$$\mu(a|s) > 0 \text{ for every } s, a \text{ at which } \pi(a|s) > 0$$

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, \ldots, S_T\} \sim \mu$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
  - Learn about *optimal* policy while following *exploratory* policy
  - Learn about *multiple* policies while following *one* policy

Importance Sampling

- Estimate the expectation of a function

$$\mathbb{E}_{X\sim\pi}[g(X)] = \sum \pi(x)f(x) \approx \frac{1}{n}\sum_{k=1}^{n} g(x_k), x_k \sim \pi$$

- But sometimes it is difficult to sample x from π, then we can sample x from another distribution μ, then correct the weight

$$\mathbb{E}_{X\sim\pi}[g(X)] = \sum \pi(x)g(x) = \sum \mu(x)\frac{\pi(x)}{\mu(x)}g(x)$$

$$= \mathbb{E}_{X\sim\mu}\left[\frac{\pi(X)}{\mu(X)}g(X)\right] \approx \frac{1}{n}\sum_{k=1}^{n}\frac{\pi(x_k)}{\mu(X_k)}g(x_k), x_k \sim \mu$$

- For off-policy learning: weight each return by the ratio of the probabilities of the trajectory under the two policies

Importance Sampling for Off-Policy Monte-Carlo

- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies
- Multiply importance sampling corrections along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

- Update value towards *corrected* return

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{\pi/\mu} - V(S_t) \right)$$

- Cannot use if μ is zero when π is non-zero
- Importance sampling can dramatically increase variance

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π

$$
\begin{aligned}
&\Pr\left\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \middle| S_t, A_{t:T-1} \sim \pi\right\} \\
&\quad = \pi\left(A_t|S_t\right) p\left(S_{t+1}|S_t, A_t\right) \pi\left(A_{t+1}|S_{t+1}\right) \cdots p\left(S_T|S_{T-1}, A_{T-1}\right) \\
&\quad = \prod_{k=t}^{T-1} \pi\left(A_k|S_k\right) p\left(S_{k+1}|S_k, A_k\right),
\end{aligned}
$$

- In importance sampling, each arm is weighted by the relative probability of the trajectory under the two policies

$$
\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi\left(A_k|S_k\right) p\left(S_{k+1}|S_k, A_k\right)}{\prod_{k=t}^{T-1} \mu\left(A_k|S_k\right) p\left(S_{k+1}|S_k, A_k\right)} = \prod_{k=t}^{T-1} \frac{\pi\left(A_k|S_k\right)}{\mu\left(A_k|S_k\right)}
$$

- This is called the *importance sampling ratio*
- All importance sampling ratios have expected value 1

$$
\mathbb{E}_{A_k \sim \mu}\left[\frac{\pi\left(A_k|S_k\right)}{\mu\left(A_k|S_k\right)}\right] = \sum_a \mu\left(a|S_k\right) \frac{\pi\left(a|S_k\right)}{\mu\left(a|S_k\right)} = \sum_a \pi\left(a|S_k\right) = 1.
$$

Off-Policy MC Policy Evaluation

Incremental off-policy every-visit MC policy evaluation (returns $Q \approx q_\pi$

Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $C(s,a) \leftarrow 0$

Repeat forever:
  $\mu \leftarrow$ any policy with coverage of π
  Generate an episode using μ:
    $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1, T-2, \ldots$ downto 0:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $W \leftarrow W \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$
    If $W = 0$ then ExitForLoop

Off-Policy MC Control

Off-policy every-visit MC control (returns $\pi \approx \pi_*$)

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow$ arbitrary

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

Repeat forever:

$\mu \leftarrow$ any soft policy

Generate an episode using μ:

$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$

$G \leftarrow 0$

$W \leftarrow 1$

For $t = T-1, T-2, \ldots$ downto 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 ExitForLoop

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

Target policy is greedy and deterministic

Behavior policy is soft, typically ε-greedy

Importance Sampling for Off-Policy TD

- Use TD targets generated from μ to evaluate π
- Weight TD target $R + \gamma V(S')$ by importance sampling
- Only need a single importance sampling correction

$$V(S_t) \leftarrow V(S_t) + \alpha \left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} (R_{t+1} + \gamma V(S_{t+1})) - V(S_t) \right)$$

- Much lower variance than Monte-Carlo importance sampling
- Policies only need to be similar over a single step

Outline

Q-Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action to evaluate is chosen using behavior policy $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$
- Which means a separate policy is used to choose the alternative action in the future
- And update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Off-Policy Control with Q-Learning

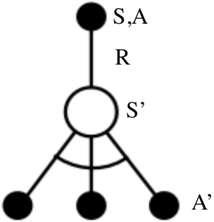
- We now allow both behaviour and target policies to improve
- The target policy π is greedy w.r.t. $Q(s, a)$

$$\pi(S_{t+1}) = \underset{a'}{\arg\max}\, Q(S_{t+1}, a')$$

- The behaviour policy μ is e.g. ϵ-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ = & R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\arg\max}\, Q(S_{t+1}, a')) \\ = & R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

Q-Learning Control Algorithm



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

Theorem

Q-learning control converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$

Q-Learning Algorithm for Off-Policy Control

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

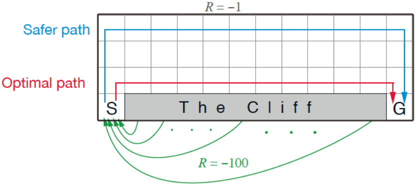
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

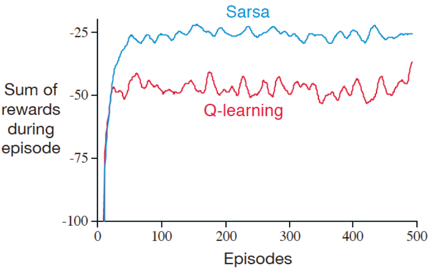
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
 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

Q-Learning Demo

https://www.cs.ubc.ca/~poole/demos/rl/q.html

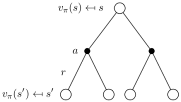
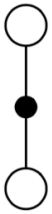
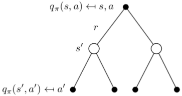
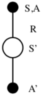
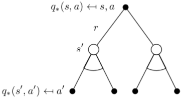
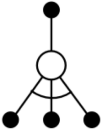
Cliff Walking Example





Outline

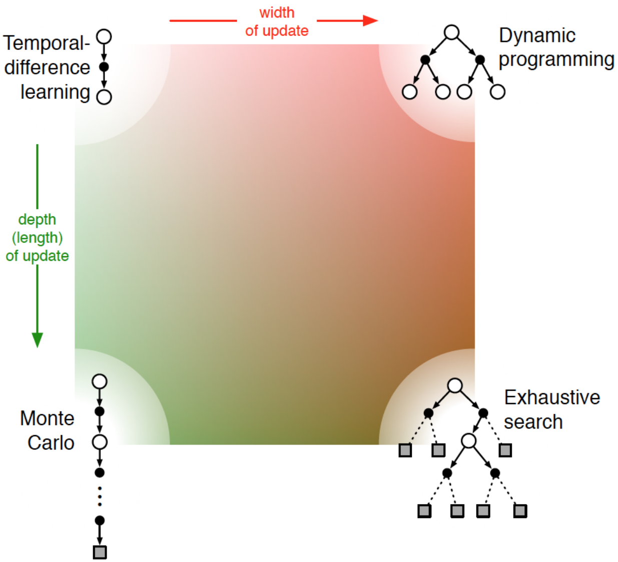
Relationship Between DP and TD

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

Relationship Between DP and TD

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation	TD Learning
$V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$	$V(S) \overset{\alpha}{\leftarrow} R + \gamma V(S')$
Q-Policy Iteration	Sarsa
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$	$Q(S,A) \overset{\alpha}{\leftarrow} R + \gamma Q(S',A')$
Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \overset{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

Unified View of Reinforcement Learning



Outline

Main References

- Reinforcement Learning: An Introduction (second edition), R. Sutton & A. Barto, 2018.
- RL course slides from Richard Sutton, University of Alberta.
- RL course slides from David Silver, University College London.