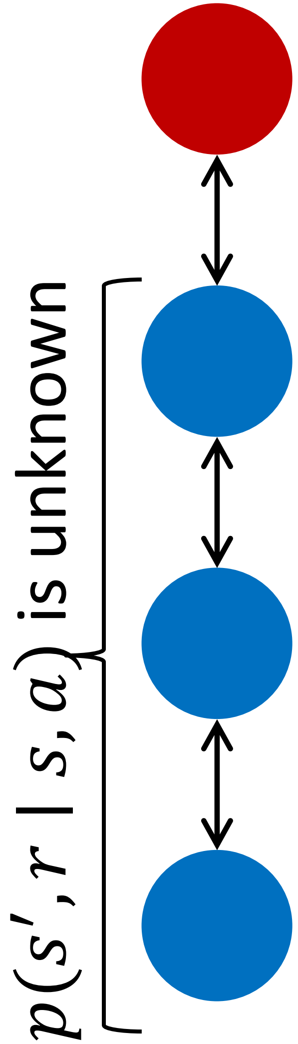
Brain-Inspired Artificial Intelligence
4: Model-Free Reinforcement Learning

Eiji Uchibe
Dept. of Brain Robot Interface
ATR Computational Neuroscience Labs.

Outline

- We have studied Model-based RL
  - Bellman expectation equation and Bellman optimality equation
- We will study model-free approaches to solve MDP problems
- Model-free methods
  - TD-learning
  - SARSA
  - Q-learning

Model-based vs. Model-free Methods



Model-based approach

- $p(s', r \mid s, a)$ is known explicitly
- Policy iteration, value iteration, linear programming

Model learning and model-free approach

- Estimate an environmental model $\hat{p}(s', r \mid s, a)$ from samples
- Apply model-free approach to $\hat{p}$

Experience replay and model-free approach

- Store experienced samples in the buffer
- Apply model-free approach with samples drawn from the buffer

Model-free approach

- Train V^π, Q^π, or π directly from samples drawn from p
- Discard experienced samples without reuse

Model-Free Reinforcement Learning

- Model-based RL needs $p(s', r|s, a)$
- Model-free RL uses samples $\{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$

Model-Free Policy Evaluation

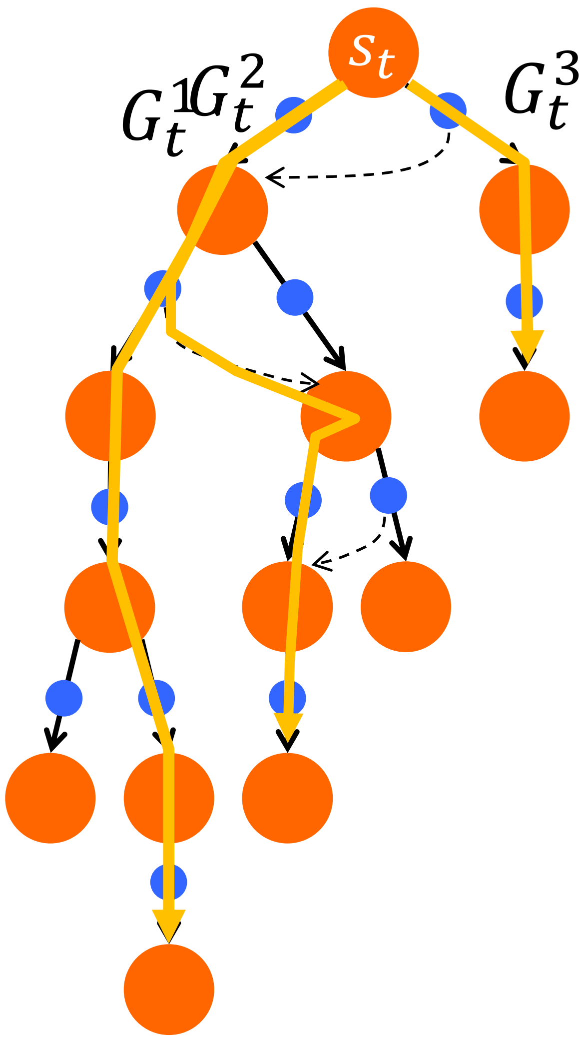
- The goal is to estimate V^π when π is given, but we do not know $p(s', r \mid s, a)$

- Reminder: Definition of $v_\pi(s)$

$$V^\pi(s) = \mathbb{E}_\pi\{G_t \mid s_t = s\},\ G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- A naïve way is to collect many returns $\{G_t^i\}_{i=1}^N$ for every state and compute its empirical average

$$V^\pi(s_t) = \frac{1}{N}\sum_{i=1}^{N} G_t^i$$



Model-Free Policy Evaluation

- To derive an online update rule, we consider the sample-approximate Bellman expectation operator

$$B_\pi V(s) = \sum_{a,s',r} p(a, s', r|s)[r + \gamma V(s')]$$

$$\hat{B}_\pi V(s_t) \triangleq r_{t+1} + \gamma V(s_{t+1})$$

Note

$p(a, s', r \mid s)$ depends on $\pi(a \mid s)$ because

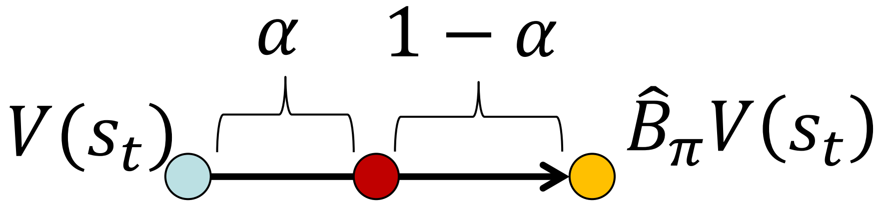
$$p(a, s', r \mid s) = p(s', r \mid s, a)\pi(a \mid s)$$

- It is proved that $\hat{B}_\pi$ converges to B_π when we have $\{s_t^i, a_t^i, r_{t+1}^i, s_{t+1}^i\}_{i=1}^N$ as $N \to \infty$

TD Learning

- Weighted average between the current estimate $V(s_t)$ and $\hat{B}_\pi V(s_t)$

$$V(s_t) \leftarrow (1-\alpha)V(s_t) + \alpha \hat{B}_\pi V(s_t)$$



where $\alpha \in [0, 1]$ is called the learning rate

- Re-write the right hand side of the equation

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t, \qquad \delta_t \triangleq \underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{target value}} - V(s_t)$$

- This is called TD learning, where δ_t is a temporal difference (TD) error

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

R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

TD Approach for Estimating Q^π

- TD-learning is a method of policy evaluation when $p(s', r \mid s, a)$ is unknown

- However, we cannot perform policy improvement in the model-free settings:

$$\pi(s) \leftarrow \arg\max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma V^\pi(s')]$$

- So, we need Bellman expectation operator for Q^π and its sample-approximator

Bellman Expectation Operator for Q^{π}

- Reminder: Bellman expectation operator for v_{π}

$$B_{\pi}V(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

- Bellman expectation operator for Q^{π}

$$T_{\pi}Q(s,a) \triangleq \sum_{s',r} p(s',r \mid s,a)[r + \gamma V(s')]$$

$$= \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(a' \mid s')Q(s',a')\right]$$

Sample-Approximate Bellman Expectation Operator for Q^{π}

- Similarly, we obtain the sample-approximate Bellman expectation operator

$$T_{\pi}Q(s,a) \triangleq \sum_{r,s',a'} p(r,s',a'|s,a)[r+\gamma Q(s',a')]$$

$$\downarrow$$

$$\hat{T}_{\pi}Q(s_t,a_t) = r_{t+1} + \gamma Q(s_{t+1},a_{t+1})$$

Note

$$p(r,s',a' \mid s,a) = \pi(a' \mid s')p(s',r \mid s,a)$$

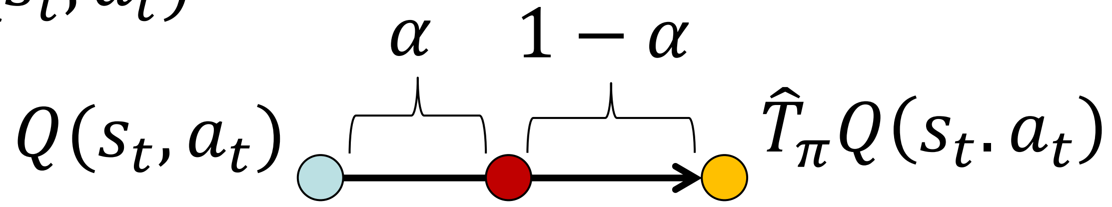
- It is proved that $\hat{T}_{\pi}$ converges to T_{π} when we have $\{s_t^i, a_t^i, r_{t+1}^i, s_{t+1}^i, a_{t+1}^i\}_{i=1}^{N}$ as $N \to \infty$

SARSA: On-Policy TD Control

- Weighted average between the current estimate $Q(s_t, a_t)$ and $\hat{T}_\pi Q(s_t, a_t)$

  $$Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha \hat{T}_\pi Q(s_t, a_t)$$

  where $\alpha \in [0, 1]$ is called the learning rate



- Re-write the right hand side of the equation

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t^{\mathrm{SARSA}}$$

$$\delta_t^{\mathrm{SARSA}} \triangleq \underbrace{r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})}_{\text{target value}} - Q(s_t, a_t)$$

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 $Q(terminal, \cdot) = 0$

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

R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

Note on TD and SARSA

- The operators depend on the current learning policy explicitly

$$B_\pi V(s) = \sum_{a,s',r} p(s',r|s,a)\pi(a \mid s)[r + \gamma V(s')] \approx r_{t+1} + \gamma V(S_{t+1})$$

$$T_\pi Q(s,a) \triangleq \sum_{r,s',a'} \pi(a' \mid s')p(r,s'|s,a)[r + \gamma Q(s',a')]$$

$$\approx r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

- Therefore, samples to approximate the operators should be collected by the learning policy $\pi(a \mid s)$

$\Rightarrow$ on-policy learning

From On-Policy Learning to Off-Policy Learning

- On-policy learning is sample-inefficient because it cannot use samples $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ generated by a behavior policy $b(a \mid s)$ that is different from $\pi(a \mid s)$

- Consider Bellman optimality operator

$$B_* V(s) = \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma V(s')] = \max_a \mathbb{E}[r + \gamma V(s')]$$

$$\cancel{\hat{B}_* V(s) = \max_a [r_{t+1} + \gamma V(s_{t+1})]}$$

- Since the max operator is outside of the expectation, it is not trivial to derive the sample-approximate Bellman optimality operator for V^*

Bellman Optimality Equation for Q^*

- Reminder: Bellman optimality equation for V^*

$$V^*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^*(s')]$$

- Bellman optimality equation for Q^*

$$\begin{aligned} Q^*(s,a) &\triangleq \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^*(s')] \\ &= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} Q^*(s', a')\right] \end{aligned}$$

Bellman Optimality Operator for Q^* and its Sample-Approximation

- Define Bellman optimality operation for q_*

$$T_* Q(s,a) \triangleq \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} Q(s', a')\right]$$

- Sample-approximated Bellman optimality equation for Q^*

$$\hat{T}_* Q(s,a) \triangleq r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$$

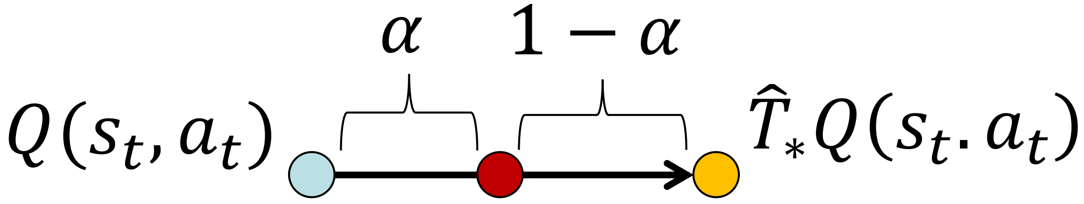
- T_* and $\hat{T}_*$ do not depend on the learning policy π

Q-Learning: Off-Policy TD Control

- Weighted average between the current estimate $Q(s_t, a_t)$ and $\hat{T}_* Q(s_t, a_t)$

  $Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha \hat{T}_* Q(s_t, a_t)$

  where $\alpha \in [0, 1]$ is called the learning rate



- Re-write the right hand side of the equation

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t^{\mathrm{Q}} \qquad \delta_t^{\mathrm{Q}} \triangleq \underbrace{r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')}_{\text{target value}} - Q(s_t, a_t)$$

- This is called Q-learning

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

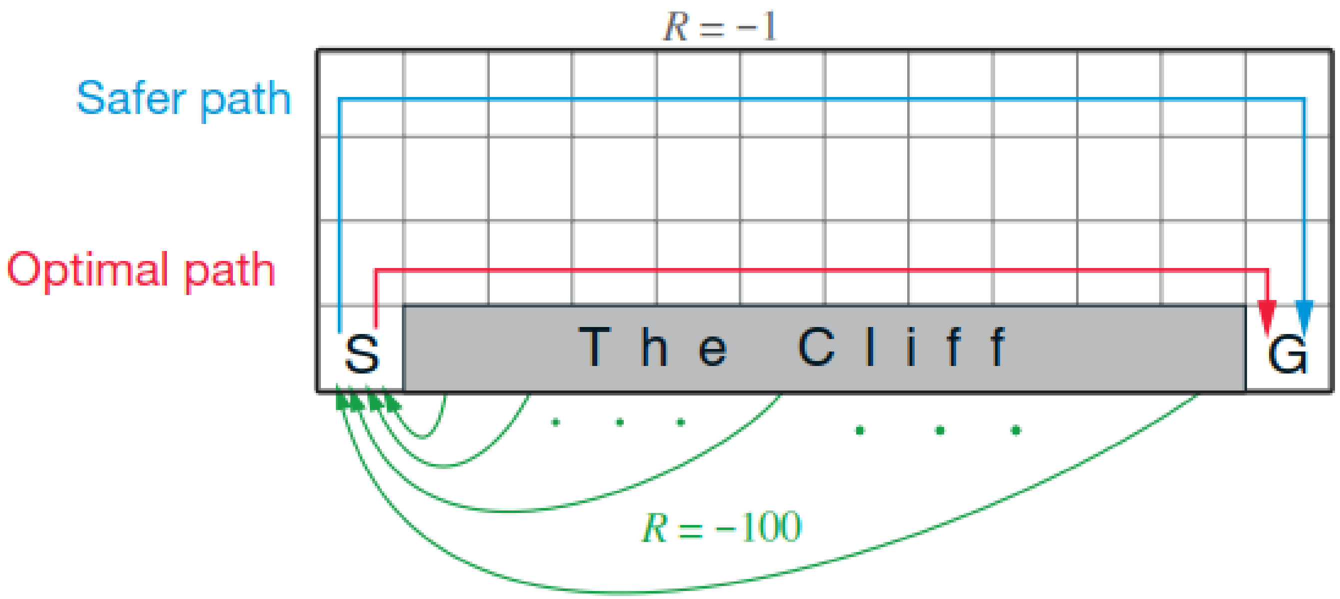
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

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

R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

Difference between Q-learning and SARSA

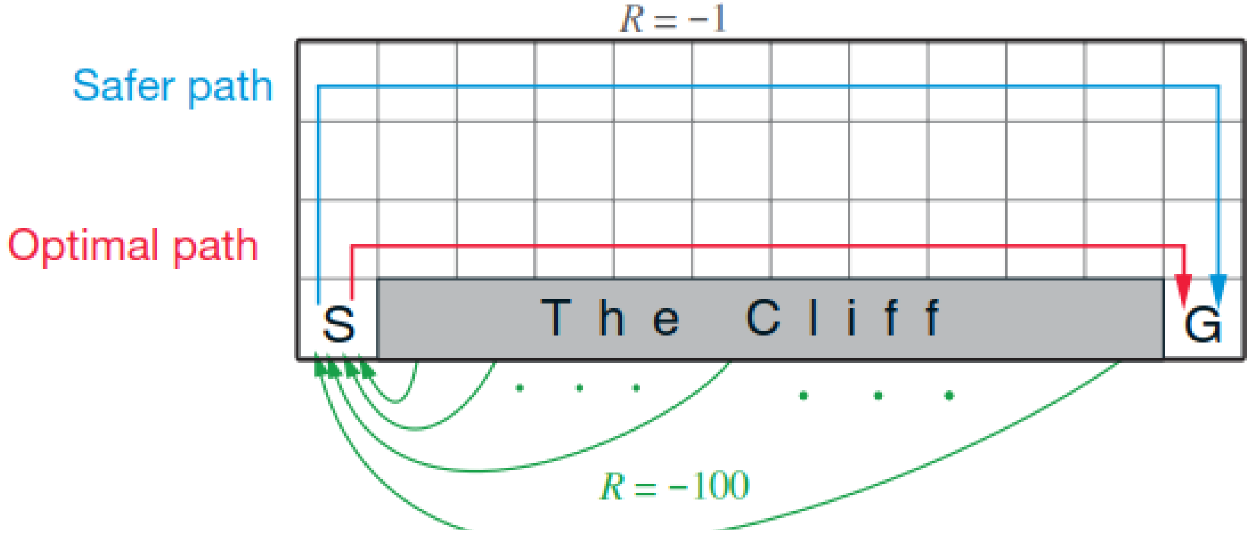
- Deterministic state transition
- 4 actions (left, right, up, and down)
- $\gamma = 1$
- $r(s, a, s') = \begin{cases} 0 \text{ if } s' \in \mathrm{G} \\ -100 \text{ if } s' \text{ is a cliff} \\ -1 \text{ otherwise} \end{cases}$
- When the agent arrives at the cliff, it is sent back to the start state

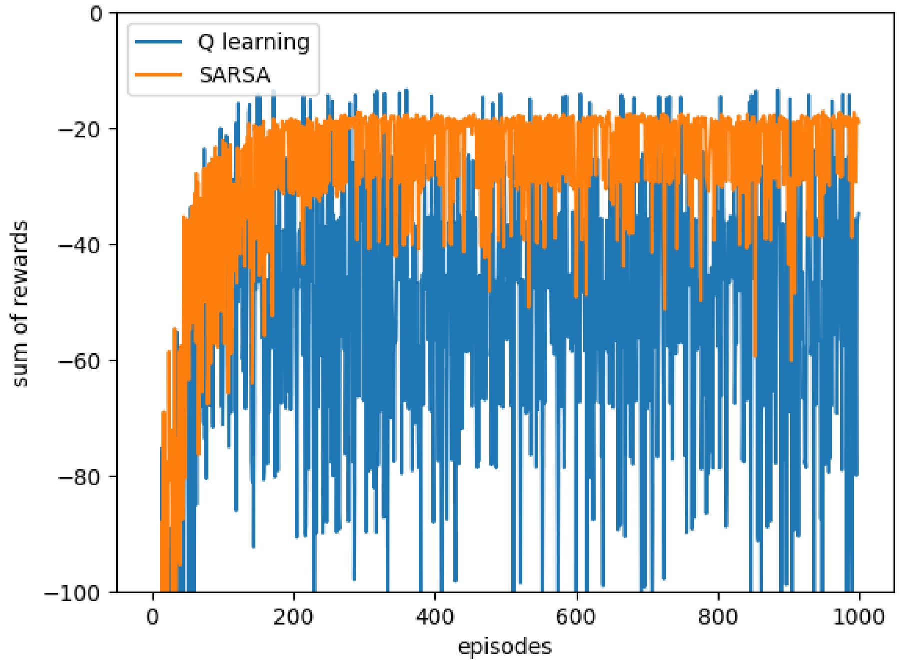


Difference between Q-learning and SARSA

Open in Colab

- SARSA found the safer path
- The agent trained with Q-learning tended to go to the cliff during learning
- In fact, Q-learning found the optimal state-action value function, and the optimal policy could be derived by the greedy policy $(\epsilon = 0)$





Note on Q-Learning

- The operator does not depend on the current learning policy

$$T_* Q(s,a) \triangleq \sum_{r,s'} p(r,s'|s,a)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

$$\approx r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a)$$

- Therefore, the policy $b(a \mid s)$ that is different from the learning policy $\pi(a \mid s)$ can be used to collect samples to approximate the operator

  ➡ off-policy learning

- The off-policy property is critical for sample efficiency