ST455: Reinforcement Learning

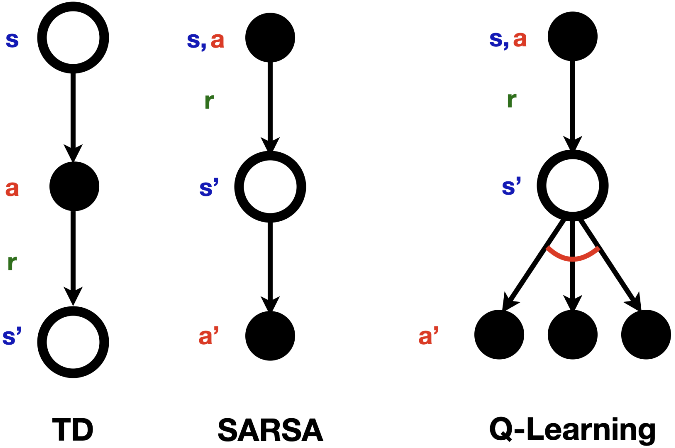
Lecture 4: Temporal Difference (TD) Learning

Chengchun Shi

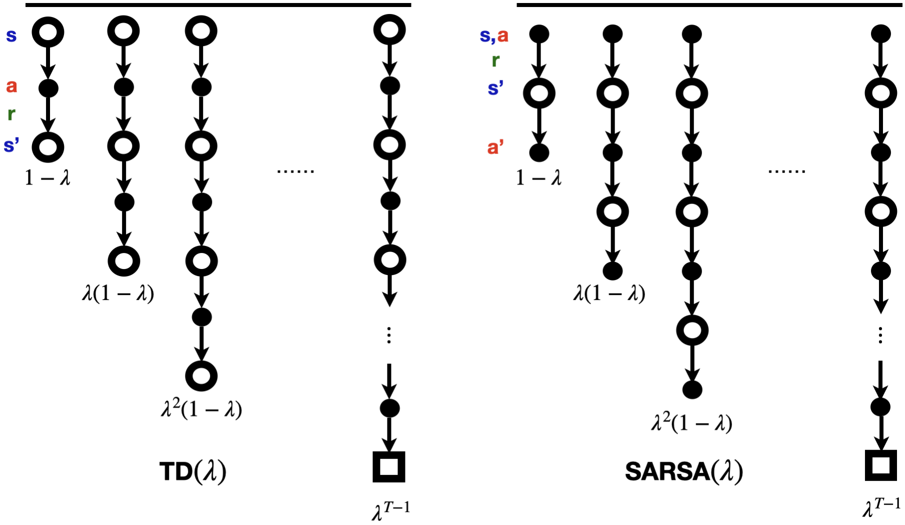
Lecture Outline

1. TD Prediction
2. SARSA
3. Q-Learning
4. TD(λ) and SARSA(λ)

Lecture Outline (Cont'd)



Lecture Outline (Cont'd)



1. TD Prediction

2. SARSA

3. Q-Learning

4. TD(λ) and SARSA(λ)

TD Learning v.s. MC Methods

- **Temporal Difference** (TD) Learning: a learning method that combines ideas from Monte Carlo (MC) methods and dynamic programming
- **Similarities** with MC methods:
  - TD is also **model free**, i.e. learns directly from the experience **without** a model (e.g., state transitions, reward functions)
  - Difference from dynamic programming
- **Difference** with MC methods:
  - MC methods: updates the value **episode-by-episode** (applies to **episodic** tasks)
  - TD methods: updates the value **step-by-step** (applies to **continuous** tasks as well)

MC Prediction

- **Objective**: learns V^π from experience under π
- MC Policy Evaluation: $V(s) \leftarrow \textbf{average}[\textbf{Returns}(s)]$
- Incremental update for every-visit MC prediction:

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

  where α_t is $\frac{1}{\#[\textbf{Returns}(S_t)]}$ at time t
- We may regard G_t as a **target**
- The update can be performed after return G_t is observed
- i.e. after the episode is completed

TD Prediction

- Unlike MC methods, TD methods wait only until **next time step**
- The simplest TD method (so called TD($\mathbf{0}$)) considers the update

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

- This update rule has $R_t + \gamma V(S_{t+1})$ as the **target**
- Considered as a **bootstrap** method: update in part based on an existing estimate
- Different from "bootstrap" in statistics: a resampling method (e.g., sample with replacement) for **uncertainty quantification** of a given estimate

MC vs TD update

- Notice that under the MDP assumption

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}^\pi(G_t | S_t = s) && (1)\\
&= \mathbb{E}^\pi(\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s) \\
&= \mathbb{E}^\pi[R_t + \gamma V^\pi(S_{t+1}) | S_t = s] && (2)
\end{aligned}
$$

- MC methods use as the target the random variable in (1)
- TD methods use as the target the random variable in (2)
  - **Immediate reward** and estimate of the **future value**

Bootstrapping and Sampling

- **Bootstrapping**: update involves an estimate
  - MC does **not** bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling**: update samples an expectation
  - MC samples
  - DP does **not** sample
  - TD samples

TD(0): Pseudocode

- **Input**: π policy to be evaluated, step size α
- **Initialization**: V arbitrary
- **Repeat** for each episode:

    Initialize state s

    Repeat for each step of the episode:

        $a \leftarrow$ action given by π for s

        Take action a, observe reward r and next state s'

        $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$

        $s \leftarrow s'$

    until s is a terminal state

Pros & Cons of MC vs TD

- MC must wait until the **end** of episode
- MC learns from **complete** sequences
- MC only works for **episodic** (terminating) environments

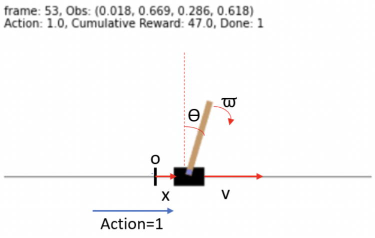
- TD can learn online after **each step**
- TD can learn from **incomplete** sequences
- TD works in **continuing** environments

Pros & Cons of MC vs TD (Cont'd)

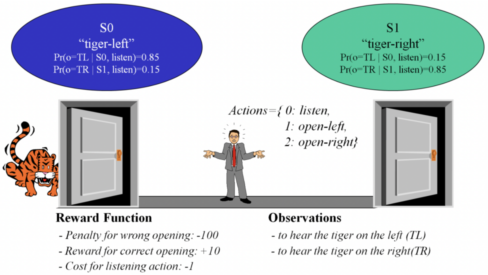
- Bias/Variance Trade-Off
- Return G_t is **unbiased** estimate of $V^\pi(S_t)$
- Oracle target $R_t + \gamma V^\pi(S_{t+1})$ is **unbiased** estimate of $V^\pi(S_t)$
- TD target $R_t + \gamma V(S_{t+1})$ is **biased** estimate of $V^\pi(S_t)$
- TD target has much lower **variance** than the return
  - Return depends on **many** random actions, transitions, rewards
  - TD target depends on **one** random action, transition, reward
- MC has **high variance, zero bias, insensitive to initialization**
- TD has **low variance, some bias, sensitive to initialization**

Pros & Cons of MC vs TD (Cont'd)

- TD exploits **Markov** & **stationary** properties
- Relies on the **Bellman equation**
- More **efficient** in MDP environments



- MC **does** not exploit these properties
- More **flexible** in non-MDP environments (e.g., POMDP)



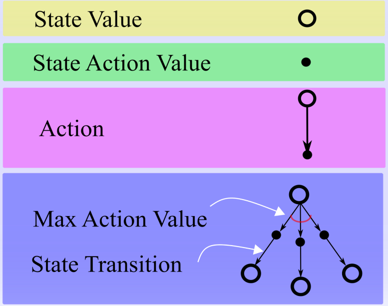
Rate of Convergence

- For i.i.d. random variables $X_1, \cdots, X_n$ with mean μ and variance σ^2,

$$\sqrt{n}(\bar{X} - \mu) \to N(0, \sigma^2),$$

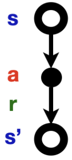
  according to CLT.
- $\bar{X}$ converges to μ at a rate of $n^{-1/2}$.
- For n episodes with T time points per episode, first-visit MC converges at a rate of $n^{-1/2}$.
- For n episodes with T time points per episode, TD converges at a rate of $(nT)^{-1/2}$, with proper choice of step sizes [see e.g., Tadić, 2002].
- First-visit MC requires $n \to \infty$ to be consistent
- TD requires either n or $T \to \infty$ to be consistent

Backup Diagram

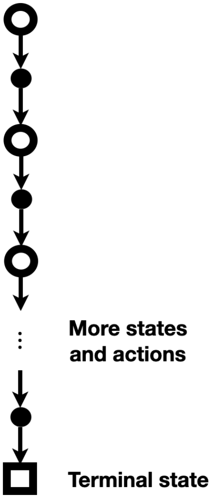


Taken from `https://towardsdatascience.com/all-about-backup-diagram-fefb25aaf804`

Backup Diagram (Cont'd)



TD



MC

1. TD Prediction
2. **SARSA**
3. Q-Learning
4. TD(λ) and SARSA(λ)

SARSA: an On-Policy TD Control

- **SARSA**: a TD method for **policy optimisation**
  - Follows the pattern of policy iteration
  - Uses TD prediction method for **policy evaluation**
  - Uses ε-greedy exploration for **policy improvement**
- Similar to MC control, estimate state-action value $\boldsymbol{Q^{\pi}}(\boldsymbol{s}, \boldsymbol{a})$ (instead of the state value $\boldsymbol{V^{\pi}}(\boldsymbol{s})$) for the control problem
- Different from MC control, update the state-value **every time step**

Bellman Equations

- Bellman equation for the (state) value function:

$$V^\pi(s) = \mathbb{E}[R_t + \gamma V^\pi(S_{t+1}) | S_t = s].$$

- Bellman equation for the state-action value function:

$$Q^\pi(s, a) = \mathbb{E}\left[R_t + \gamma \sum_{a'} \pi(a' | S_{t+1}) Q^\pi(S_{t+1}, a') | A_t = a, S_t = s\right],$$

or equivalently,

$$Q^\pi(s, a) = \mathbb{E}^\pi[R_t + \gamma Q^\pi(S_{t+1}, A_{t+1}) | A_t = a, S_t = s].$$

SARSA: Policy Evaluation

- Incremental estimation of the state-action value function:

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

  for non-terminal state S_{t+1}
- If S_{t+1} is a terminal state, $Q(S_{t+1}, A_{t+1}) = 0$
- This update uses every element of the quintuple of variables:

$$\begin{matrix} (S_t, & A_t, & R_t, & S_{t+1}, & A_{t+1}) \\ S & A & R & S & A \end{matrix}$$

SARSA: Pseudocode

- **Initialization**: Q arbitrary
- **Repeat** for each episode:

    Initialize state s

    Choose action a from s using policy derived from Q (ε-greedy)

    Repeat for each step of the episode:

        Take action a, observe reward r and next state s'

        $a' \leftarrow$ action from s' using policy derived from Q (ε-greedy)

        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$

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

    until s is a terminal state

Convergence of SARSA

Theorem

SARSA converges to the optimal Q-function, $Q(s, a) \to Q^{\pi^{\text{opt}}}(s, a)$ *for any* s *and* a*, if*

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

$$\sum_{t=0}^{\infty} \mathbb{I}(S_t = s, A_t = a) = \infty.$$

- *The policy converges to a greedy policy,*

$$\lim_{t \to \infty} \pi_t(a|s) = \mathbb{I}(a = \arg\max_{a'} Q_t(s, a'))$$

- *Robbins-Monro sequence of step-sizes [Robbins and Monro, 1951],*

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

Convergence of SARSA (Cont'd)

- Condition 1: All state-action pairs are **explored** infinitely many times

  $\Rightarrow \varepsilon$ to be strictly positive
- Condition 2: The policy converges to a **greedy** policy

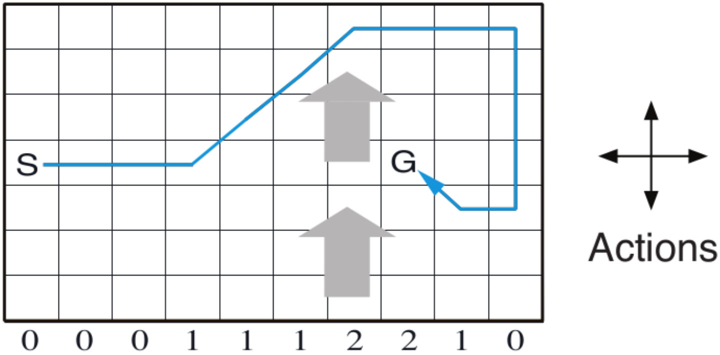
  $\Rightarrow \varepsilon_t$ decays to zero as t grows to infinity
- Condition 3: Robbins-Monro sequence of step-sizes

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

$\Rightarrow \alpha_t$ proportional to t^{-c} for $1/2 < c \leq 1$.

$\sum_t t^{-c} = \infty$ when $c \leq 1$ and $\sum_t t^{-c} < \infty$ when $c > 1$

Windy Gridworld Example



- An **episodic** task
- Rewards of -1 until **goal** is reached
- Strength of wind indicated by numbers

Windy Gridworld Example (Cont'd)

```
Optimal policy is:
['R' 'D' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'D']
['R' 'R' 'U' 'L' 'R' 'R' 'R' 'U' 'R' 'D']
['L' 'U' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'D']
['R' 'R' 'R' 'R' 'R' 'R' 'U' 'G' 'R' 'D']
['R' 'D' 'D' 'R' 'R' 'R' 'U' 'D' 'L' 'L']
['D' 'D' 'D' 'R' 'R' 'U' 'U' 'D' 'R' 'U']
['R' 'R' 'U' 'R' 'U' 'U' 'U' 'U' 'R' 'L']
['0' '0' '0' '1' '1' '1' '2' '2' '1' '0']
```

1. TD Prediction
2. SARSA
3. **Q-Learning**
4. TD(λ) and SARSA(λ)

Q-Learning

- One of the most **popular** class of RL algorithms
- Variants include double Q-learning, fitted Q-iteration, deep Q-network (DQN), quantile DQN (more in later lectures)
- **Main idea**: learn the optimal Q-function $Q^{\pi^{\text{opt}}}$ based on the Bellman **optimality** equation and derive the optimal policy (see Appendix for the proof)

$$\pi^{\text{opt}}(s) = \arg\max_{a} Q^{\pi^{\text{opt}}}(s, a)$$

- Focus on **tabular Q-learning** in this lecture (finite MDP, discrete state and action)

Bellman Optimality Equation

- Bellman optimality equation for the optimal value function:

$$V^{\pi^{\text{opt}}}(s) = \max_{a} \mathbb{E}[R_t + \gamma V^{\pi^{\text{opt}}}(S_{t+1}) | A_t = a, S_t = s].$$

- Bellman optimal equation for the optimal Q-function (see Appendix for the proof):

$$Q^{\pi^{\text{opt}}}(s, a) = \mathbb{E}\left[R_t + \gamma \max_{a'} Q^{\pi^{\text{opt}}}(S_{t+1}, a') | A_t = a, S_t = s\right].$$

Q-Learning: an Off-Policy TD Control

- One-step SARSA update:

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

- One-step Q-learning update:

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

- In Q-learning, the action in the target is independent of the **behavior policy**
- The behavior policy has an effect on which state-actions are visited

Q-Learning: Pseudocode

- **Initialization**: Q arbitrary
- **Repeat** for each episode:

  Initialize state s

  Repeat for each step of the episode:

  $a \leftarrow$ action from s using policy derived from Q (e.g., ε-greedy)

  Take action a, observe reward r and next state s'

  $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

  $s \leftarrow s'$

  until s is a terminal state

On-Policy v.s Off-Policy

- Q-learning is **off-policy**:
  - Updates Q-values using Q-value of next state s' and **greedy** action a'
  - Assumes greedy policy were followed despite that it's not following greedy policy
- SARSA is **on-policy**:
  - Updates Q-values using Q-value of next state s' and **current policy's** action a'
  - Assumes the current policy continues to be followed

Convergence of Q-Learning

Theorem (Melo [2001])

Q-learning converges to the optimal Q-function if

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

$$\sum_{t=0}^{\infty} \mathbb{I}(S_t = s, A_t = a) = \infty.$$

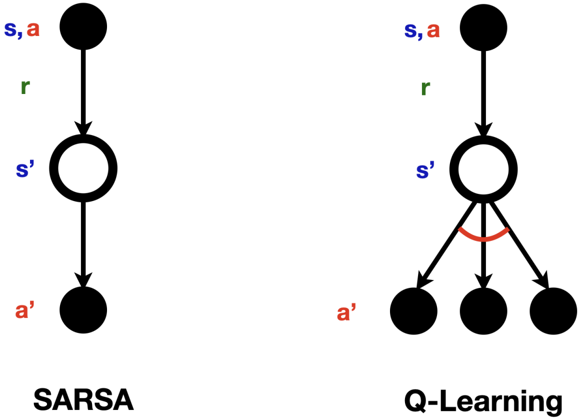
- *Robbins-Monro sequence of step-sizes [Robbins and Monro, 1951],*

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

- Only requires ε to be strictly positive
- **No** need to require ε to decay to zero

- Q-learning converges even if the behavior policy is far from the optimal

Backup Diagram



Cliff Walking Example

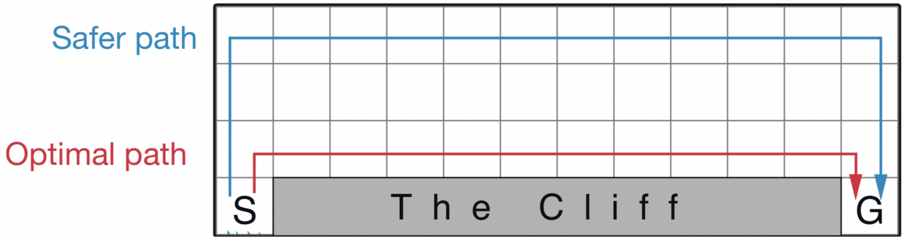


Figure: Illustrations of Cliff Walking

- Undiscounted, **episodic** task
- Actions: up, down, right and left
- Reward of -100 if stepping into **cliff**
- Reward of -1 on other transitions

Cliff Walking Example (Cont'd)

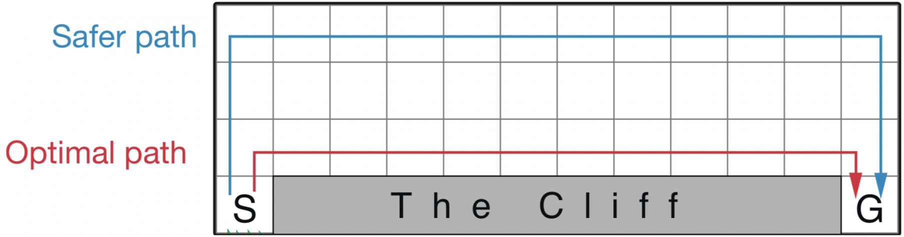


Figure: Illustrations of Cliff Walking

- Q-learning identifies the **optimal** path
- SARSA identifies a **safer** path (the optimal path is not optimal here due to that the ϵ-greedy policy, which might force the agent to fall into the cliff when walking along the optimal path, yielding a low value)

Maximization Bias

- One-step Q-learning update:

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

- Maximum over $Q(S_{t+1}, a)$ can lead to **significant positive** bias
- **Example**:
  - Oracle optimal Q-function $Q^{\pi^{\text{opt}}}(s, a) = 0$ for any (s, a)
  - $\max_a Q^{\pi^{\text{opt}}}(s, a) = 0$ for any s
  - Estimated Q-function $Q(s, a)$: uncertain, some above and some below zero
  - $\max_a Q(s, a)$ likely to be **positive**

Maximization Bias (Cont'd)

- Maximization over $\boldsymbol{Q}$ involves two steps: **greedy-action selection** and **state-action value evaluation**

$$\max_{\boldsymbol{a}} \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) = \boldsymbol{Q}(\boldsymbol{s}, \arg\max_{\boldsymbol{a}} \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}))$$

- Solution: use two different Q-functions for two steps

$$\boldsymbol{Q}_1(\boldsymbol{s}, \arg\max_{\boldsymbol{a}} \boldsymbol{Q}_2(\boldsymbol{s}, \boldsymbol{a}))$$

- **Example**: $\boldsymbol{Q}^{\pi^{\text{opt}}} = \mathbf{0}$. Due to difference between $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$, the above expression is no longer always positive

Double Q-Learning

- **Initialize** two Q-functions Q_1 and Q_2
- **Divide** time steps into two by flipping a coin on each step
- If the coin comes up with head

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

- Otherwise

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

Double Q-Learning: Pseudocode

- **Initialization**: Q_1 and Q_2 arbitrary
- **Repeat** for each episode:

  Initialize state s

  Repeat for each step of the episode:

  $a \leftarrow$ action from s using policy derived from $Q_1 + Q_2$ (e.g., ε-greedy)

  Take action a, observe reward r and next state s'

  With probability $\mathbf{0.5}$:

  $a' \leftarrow \arg\max_a Q_1(S_{t+1}, a)$

  $Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\,[R_t + \gamma Q_2(S_{t+1}, a') - Q_1(S_t, A_t)]$

  else:

  $a' \leftarrow \arg\max_a Q_2(S_{t+1}, a)$

  $Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha\,[R_t + \gamma Q_1(S_{t+1}, a') - Q_2(S_t, A_t)]$

  $s \leftarrow s'$

  until s is a terminal state

1. TD Prediction

2. SARSA

3. Q-Learning

4. **TD(λ) and SARSA(λ)**

n-Step Return

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

$$
\begin{array}{lll}
n = 1 & (\mathrm{TD}) & G_t^{(1)} = R_t + \gamma V^{\pi}(S_{t+1}) \\
n = 2 & & G_t^{(2)} = R_t + \gamma R_{t+1} + \gamma^2 V^{\pi}(S_{t+2}) \\
\vdots & & \vdots \\
n = \infty & (\mathrm{MC}) & G_t^{(\infty)} = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots
\end{array}
$$

- Define n-step return

$$G_t^{(n)} = R_t + \gamma R_{t+1} + \cdots + \gamma^n V^{\pi}(S_{t+n})$$

- n-step temporal difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t^{(n)} - V(S_t)]$$

Averaging n-Step Return

- We can average $\boldsymbol{n}$-step returns over different $\boldsymbol{n}$
- e.g. average the 2-step and 4-step returns

$$\frac{1}{2} G_t^{(2)} + \frac{1}{2} G_t^{(4)}$$

- Combines information from two different time-steps
- Can we combine information from all time-steps?

λ-Return

- The λ-return ${G_t}^{\lambda}$ combines all n-step return ${G_t}^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

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

- Notice that

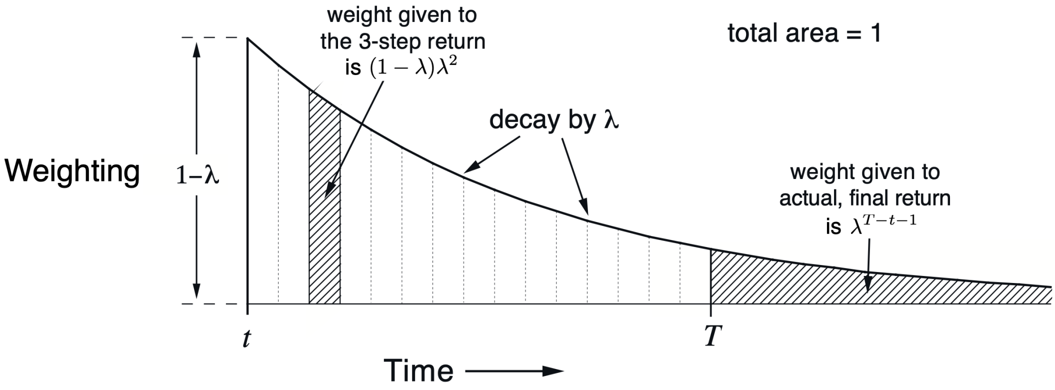
$$ \sum_{n=1}^{+\infty}(1-\lambda)\lambda^{n-1} = 1 $$

- TD(λ)

$$ V(S_t) \leftarrow V(S_t) + \alpha[{G_t}^{\lambda} - V(S_t)] $$

- Like MC, can only be computed from complete episodes

Weighting Function



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

Special Cases

- TD(λ)

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t{}^{\lambda} - V(S_t)]$$

- When $\lambda = 0$, reduces to TD method

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[G_t{}^{(1)} - V(S_t)] \\ &= V(S_t) + \alpha[R_t + \gamma V(S_{t+1}) - V(S_t)] \end{aligned}$$

- When $\lambda = 1$, reduces to MC method

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[G_t{}^{(1)} - V(S_t)] \\ &= V(S_t) + \alpha[R_t + \gamma R_{t+1} + \cdots + \gamma^T R_{t+T} - V(S_t)] \end{aligned}$$

n-Step SARSA

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

$$
\begin{array}{lll}
n = 1 & \text{(SARSA)} & Q_t^{(1)} = R_t + \gamma Q^\pi(S_{t+1}, A_{t+1}) \\
n = 2 & & Q_t^{(2)} = R_t + \gamma R_{t+1} + \gamma^2 Q^\pi(S_{t+2}, A_{t+2}) \\
\vdots & & \vdots \\
n = \infty & \text{(MC)} & Q_t^{(\infty)} = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots
\end{array}
$$

- Define n-step return

$$Q_t^{(n)} = R_t + \gamma R_{t+1} + \cdots + \gamma^n Q^\pi(S_{t+n}, A_{t+n})$$

- n-step Sarsa updates

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[Q_t^{(n)} - Q(S_t, A_t)]$$

SARSA(λ)

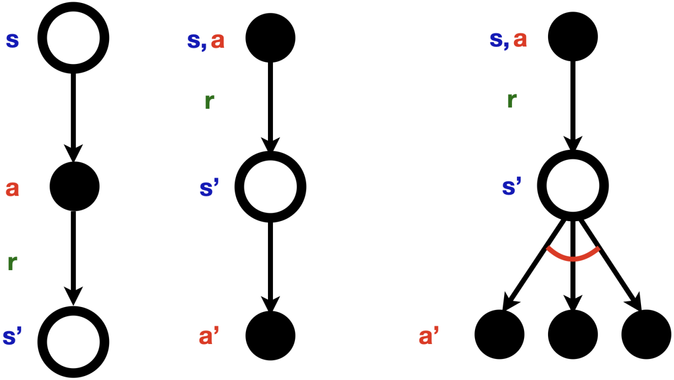
- The Q^λ-return combines all n-step return $Q_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

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

- SARSA(λ)

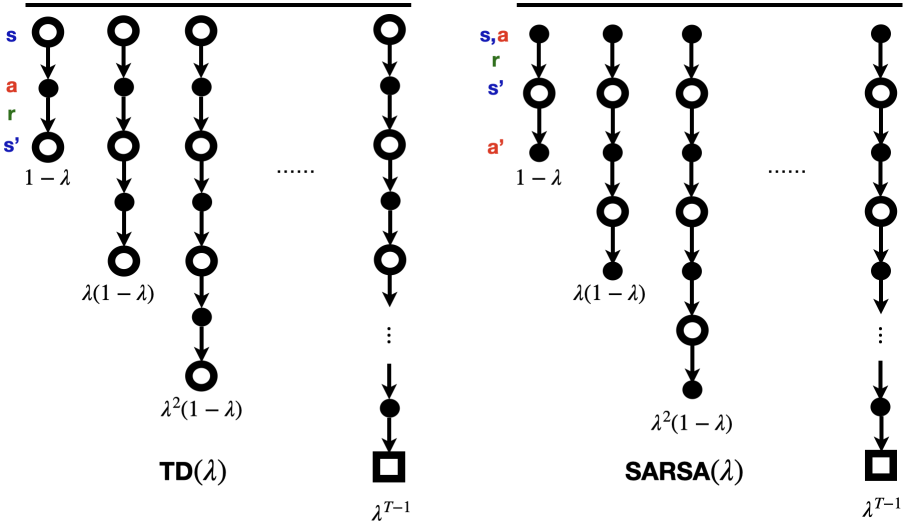
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[Q_t^\lambda - Q(S_t, A_t)]$$

Summary



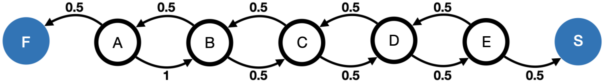
TD SARSA Q-Learning

Summary (Cont'd)

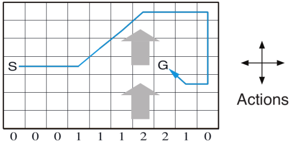


Seminar Exercises

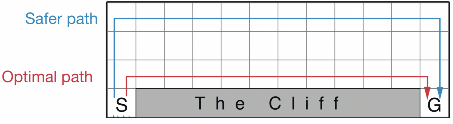
- Solution to HW3 (Deadline: Wed 12pm)
- TD: Random Walk



- Sarsa: Windy GridWorld



- Q-Learning: Cliff Walking Example



References I

Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Vladislav B Tadić. On the almost sure rate of convergence of temporal-difference learning algorithms. *IFAC Proceedings Volumes*, 35(1):455–460, 2002.

Questions

Appendix: $\pi^{\text{opt}}(s) = \arg\max_a Q^{\pi^{\text{opt}}}(s, a)$?

- $Q^{\pi^{\text{opt}}}(s, a)$ is the **value** of the policy that
  - Assigns a at the initial decision time;
  - Follow π^{opt} afterwards
- $Q^{\pi^{\text{opt}}}(s, \pi^{\text{opt}}(s)) = V^{\pi^{\text{opt}}}(s)$ is the value under the optimal policy π^{opt}
- π^{opt} is stationary and is no worse than any **history-dependent** policies (Lecture 2)

$$Q^{\pi^{\text{opt}}}(s, a) \leq V^{\pi^{\text{opt}}}(s) = Q^{\pi^{\text{opt}}}(s, \pi^{\text{opt}}(s)), \qquad \forall a.$$

- It follows that

$$\pi^{\text{opt}}(s) = \arg\max_a Q^{\pi^{\text{opt}}}(s, a)$$

Appendix: Proof of Bellman Optimality Equation

- Bellman optimal equation for the optimal Q-function:

$$Q^{\pi^{\text{opt}}}(s, a) = \mathbb{E}\left[R_t + \gamma \max_{a'} Q^{\pi^{\text{opt}}}(S_{t+1}, a') | A_t = a, S_t = s\right].$$

- **Proof**: according to Bellman equation,

$$Q^{\pi^{\text{opt}}}(s, a) = \mathbb{E}\left[R_t + \gamma Q^{\pi^{\text{opt}}}(S_{t+1}, \pi^{\text{opt}}(S_{t+1})) | A_t = a, S_t = s\right]$$

- Since $\pi^{\text{opt}}(s) = \arg\max_a Q^{\pi^{\text{opt}}}(s, a)$, it follows that

$$\max_{a'} Q^{\pi^{\text{opt}}}(S_{t+1}, a') = Q^{\pi^{\text{opt}}}(S_{t+1}, \pi^{\text{opt}}(S_{t+1}))$$