CS 747, Autumn 2022: Lecture 16

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2022

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Control with TD learning

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Control with TD learning

Multi-step Returns

- We consider prediction—estimating V^{π}.

Multi-step Returns

- We consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

Multi-step Returns

- We consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- We consider prediction—estimating V^π.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- We consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

- Can we make this update instead?

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 V^{old}(s_3) - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- We consider prediction—estimating V^π.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

- Can we make this update instead?

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 V^{old}(s_3) - V^{\text{old}}(s_2)\}.$$

Yes. It uses a 2-step return as target.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- n-step TD makes updates of the form

$$V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.$$

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the *n-step return* $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- *n-step TD* makes updates of the form

$$V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.$$

- For each $n \geq 1$, we have $\lim_{t \to \infty} V^t = V^\pi$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.
- n-step TD makes updates of the form

$$V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.$$

- For each $n \geq 1$, we have $\lim_{t \to \infty} V^t = V^\pi$.
- What is the effect of n on bootstrapping?

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- n-step TD makes updates of the form

$$V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.$$

- For each $n \geq 1$, we have $\lim_{t\to\infty} V^t = V^\pi$.
- What is the effect of n on bootstrapping? Small n means more bootstrapping.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$$G_{t:t+3}.$$

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes. $G_{t:t+1}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$$\frac{G_{t:t+1} + G_{t:t+2}}{2}.$$

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\frac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\frac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$. No.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\frac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$. No.

- Can use any convex combination of the applicable G's.

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$G_t^\lambda \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}$$

where $s^T = s_\top$ (otherwise $T = \infty$).

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$G_t^{\lambda} \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}$$

  where $s^T = s_\top$ (otherwise $T = \infty$).
- Observe that $G_t^0 = G_{t:t+1}$, yielding full bootstrapping.
- Observe that $G_t^1 = G_{t:\infty}$, a Monte Carlo estimate.
- In general, λ controls the amount of bootstrapping.

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$G_t^{\lambda} \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}$$

where $s^T = s_\top$ (otherwise $T = \infty$).

- Observe that $G_t^0 = G_{t:t+1}$, yielding full bootstrapping.
- Observe that $G_t^1 = G_{t:\infty}$, a Monte Carlo estimate.
- In general, λ controls the amount of bootstrapping.

- If $\lambda > 0$, transition (s^t, r^t, s^{t+1}) contributes to the update of every previously-visited state: that is, $s^0, s^1, s^2, \ldots, s^t$.
- The amount of contribution falls of geometrically.
- Updating with the λ-return as target can be implemented elegantly by keeping track of the "eligibility" of each previous state to be updated.

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Control with TD learning

TD(λ) algorithm

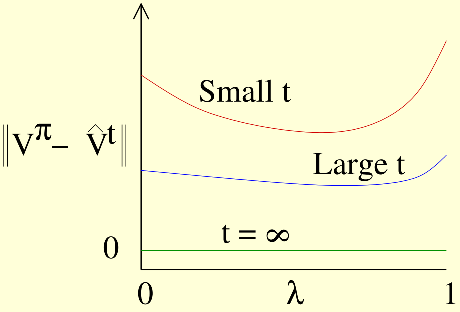
- Maintains an eligibility trace $z : S \to \mathbb{R}$.
- Implementation often called the backward view.

TD(λ) algorithm

- Maintains an eligibility trace $z : S \to \mathbb{R}$.
- Implementation often called the backward view.

Initialise $V : S \to \mathbb{R}$ arbitrarily.
Repeat for each episode:
  Set $z \to \mathbf{0}$.//Eligibility trace vector.
  Assume the agent is born in state s.
  Repeat for each step of episode:
    Take action a; obtain reward r, next state s'.
    $\delta \leftarrow r + \gamma V(s') - V(s)$.
    $z(s) \leftarrow z(s) + 1$.
    For all s:
      $V(s) \leftarrow V(s) + \alpha \delta z(s)$.
      $z(s) \leftarrow \gamma \lambda z(s)$.
    $s \leftarrow s'$.

Effect of λ



- Lower λ: more bootstrapping, more bias (less variance).
- Higher λ: more dependence on empirical rewards, more variance (less bias).
- For finite t, error is usually lowest for intermediate λ value.

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Control with TD learning

Sketch

1. Maintain action value function estimate $\hat{Q}^t : S \times A \to \mathbb{R}$ for $t \geq 0$, initialised arbitrarily.
   We would like to get $\hat{Q}^t$ to converge to $Q^\star$.

Sketch

1. Maintain action value function estimate $\hat{Q}^t : S \times A \to \mathbb{R}$ for $t \geq 0$, initialised arbitrarily.
   We would like to get $\hat{Q}^t$ to converge to $Q^\star$.

2. Follow policy π^t at time step $t \geq 0$, for example one that is ϵ_t-greedy with respect to $\hat{Q}^t$.
   Set ϵ_t to ensure infinite exploration of every state-action pair and also being greedy in the limit.

Sketch

1. Maintain action value function estimate $\hat{Q}^t : S \times A \to \mathbb{R}$ for $t \geq 0$, initialised arbitrarily.
   We would like to get $\hat{Q}^t$ to converge to $Q^\star$.

2. Follow policy π^t at time step $t \geq 0$, for example one that is ϵ_t-greedy with respect to $\hat{Q}^t$.
   Set ϵ_t to ensure infinite exploration of every state-action pair and also being greedy in the limit.

3. Every transition (s^t, a^t, r^t, s^{t+1}) conveys information about the underlying MDP. Update $\hat{Q}^t$ based on the transition.
   Can use TD learning (suitably adapted) to make the update.

Sketch

1. Maintain action value function estimate $\hat{Q}^t : S \times A \to \mathbb{R}$ for $t \geq 0$, initialised arbitrarily.
   We would like to get $\hat{Q}^t$ to converge to $Q^\star$.

2. Follow policy π^t at time step $t \geq 0$, for example one that is ϵ_t-greedy with respect to $\hat{Q}^t$.
   Set ϵ_t to ensure infinite exploration of every state-action pair and also being greedy in the limit.

3. Every transition (s^t, a^t, r^t, s^{t+1}) conveys information about the underlying MDP. Update $\hat{Q}^t$ based on the transition.
   Can use TD learning (suitably adapted) to make the update.

   We consider three different update rules.

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.
- Update made to $\hat{Q}^t$ after observing transition s^t, a^t, r^t, s^{t+1}:

$$\hat{Q}^{t+1}(s^t, a^t) \leftarrow \hat{Q}^t(s^t, a^t) + \alpha_{t+1}\{\text{Target} - \hat{Q}^t(s_t, a^t)\}.$$

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.
- Update made to $\hat{Q}^t$ after observing transition s^t, a^t, r^t, s^{t+1}:

$$\hat{Q}^{t+1}(s^t, a^t) \leftarrow \hat{Q}^t(s^t, a^t) + \alpha_{t+1}\{\text{Target} - \hat{Q}^t(s_t, a^t)\}.$$

Q-learning: $\text{Target} = r^t + \gamma \max_{a \in A} \hat{Q}^t(s^{t+1}, a).$

Sarsa: $\text{Target} = r^t + \gamma \hat{Q}^t(s^{t+1}, a^{t+1}).$

Expected Sarsa: $\text{Target} = r^t + \gamma \sum_{a \in A} \pi^t(s^{t+1}, a)\hat{Q}^t(s^{t+1}, a).$

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.
- Update made to $\hat{Q}^t$ after observing transition s^t, a^t, r^t, s^{t+1}:

$$\hat{Q}^{t+1}(s^t, a^t) \leftarrow \hat{Q}^t(s^t, a^t) + \alpha_{t+1}\{\text{Target} - \hat{Q}^t(s_t, a^t)\}.$$

Q-learning: $\text{Target} = r^t + \gamma \max_{a \in A} \hat{Q}^t(s^{t+1}, a)$.

Sarsa: $\text{Target} = r^t + \gamma \hat{Q}^t(s^{t+1}, a^{t+1})$.

Expected Sarsa: $\text{Target} = r^t + \gamma \sum_{a \in A} \pi^t(s^{t+1}, a)\hat{Q}^t(s^{t+1}, a)$.

- Q-learning's update is off-policy; the other two are on-policy.

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.
- Update made to $\hat{Q}^t$ after observing transition s^t, a^t, r^t, s^{t+1}:

$$\hat{Q}^{t+1}(s^t, a^t) \leftarrow \hat{Q}^t(s^t, a^t) + \alpha_{t+1}\{\text{Target} - \hat{Q}^t(s_t, a^t)\}.$$

Q-learning: $\text{Target} = r^t + \gamma \max_{a \in A} \hat{Q}^t(s^{t+1}, a).$

Sarsa: $\text{Target} = r^t + \gamma \hat{Q}^t(s^{t+1}, a^{t+1}).$

Expected Sarsa: $\text{Target} = r^t + \gamma \sum_{a \in A} \pi^t(s^{t+1}, a)\hat{Q}^t(s^{t+1}, a).$

- Q-learning's update is off-policy; the other two are on-policy.
- $\lim_{t \to \infty} \hat{Q}^t = Q^\star$ for all three algorithms if π^t is ϵ_t-greedy w.r.t. $\hat{Q}^t$.

Three Control Algorithms

- From state s^t, action taken is $a^t \sim \pi^t(s^t)$.
- Update made to $\hat{Q}^t$ after observing transition s^t, a^t, r^t, s^{t+1}:

$$\hat{Q}^{t+1}(s^t, a^t) \leftarrow \hat{Q}^t(s^t, a^t) + \alpha_{t+1}\{\text{Target} - \hat{Q}^t(s_t, a^t)\}.$$

Q-learning: $\text{Target} = r^t + \gamma \max_{a \in A} \hat{Q}^t(s^{t+1}, a).$

Sarsa: $\text{Target} = r^t + \gamma \hat{Q}^t(s^{t+1}, a^{t+1}).$

Expected Sarsa: $\text{Target} = r^t + \gamma \sum_{a \in A} \pi^t(s^{t+1}, a)\hat{Q}^t(s^{t+1}, a).$

- Q-learning's update is off-policy; the other two are on-policy.
- $\lim_{t \to \infty} \hat{Q}^t = Q^\star$ for all three algorithms if π^t is ϵ_t-greedy w.r.t. $\hat{Q}^t$.
- If $\pi^t = \pi$ (time-invariant) and it still visits every state-action pair infinitely often, then $\lim_{t \to \infty} \hat{Q}^t$ is Q^π for Sarsa and Expected Sarsa, but is $Q^\star$ for Q-learning!

Temporal Difference Learning: Review

- Temporal difference (TD) learning is at the heart of RL.
- It is an instance of on-line learning (computationally cheap updates after each interaction).

Temporal Difference Learning: Review

- Temporal difference (TD) learning is at the heart of RL.
- It is an instance of on-line learning (computationally cheap updates after each interaction).
- Bootstrapping exploits the underlying Markovian structure, which Monte Carlo methods ignore.
- The TD(λ) family of algorithms, $\lambda \in [0, 1]$, allows for controlling the extent of bootstrapping: $\lambda = 0$ implements "full bootstrapping" and $\lambda = 1$ is "no bootstrapping."

Temporal Difference Learning: Review

- Temporal difference (TD) learning is at the heart of RL.
- It is an instance of on-line learning (computationally cheap updates after each interaction).
- Bootstrapping exploits the underlying Markovian structure, which Monte Carlo methods ignore.
- The TD(λ) family of algorithms, $\lambda \in [0, 1]$, allows for controlling the extent of bootstrapping: $\lambda = 0$ implements "full bootstrapping" and $\lambda = 1$ is "no bootstrapping."
- TD learning applies to both prediction and control.
- Q-learning, Sarsa, Expected Sarsa are all model-free (use $\theta(|S||A|)$-sized memory); can still be optimal in the limit.
- Sarsa(λ) commonly used in practice.