CSCI E-89C Deep Reinforcement Learning

Harvard Summer School

Dmitry Kurochkin

Summer 2020
Lecture 6

Contents

Contents

Q-learning: Off-Policy Estimation of $q_*(s, a)$

Recall SARSA:

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

Do we need to use A_{t+1}?

Q-learning: Off-Policy Estimation of $q_*(s,a)$

Recall SARSA:

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

Do we need to use A_{t+1}?

Q-learning:

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

will converge to $q_*(s,a)$ as long as all pairs (s,a) continue to be updated, i.e. all state-action pairs (S_t,A_t) continue to be visited.

Q-learning: Off-Policy Estimation of $q_*(s, a)$

Q-learning:

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

Algorithm:

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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A) \right]$
    $S \leftarrow S'$
  until S is terminal

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

Maximization Bias

Q-learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \underbrace{\max_a Q(S_{t+1}, a)}_{Q(S_{t+1}, \text{argmax}_a Q(S_{t+1}, a))} - Q(S_t, A_t) \Big]$$

i.e. same samples are used to

- determine optimal action
- estimate its value

$\Rightarrow$ *Maximization bias*

Example:
Bandit problem with all $q_*(a) = 0$.
Notice that $\max_a Q_t(a)$ is expected to be strictly positive for all t.

Double Learning

Keep two estimates, Q_1 and Q_2, of $q_*(s,a)$ and instead of

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

update only one of them (each case has probability 0.5 of being selected) at a time as follows:

$$\begin{cases} Q_1(S_t,A_t) \leftarrow Q_1(S_t,A_t) + \alpha\left[R_{t+1} + \gamma Q_2(S_{t+1}, \text{argmax}_a Q_1(S_{t+1},a)) - Q_1(S_t,A_t)\right] \\ Q_2(S_t,A_t) \leftarrow Q_2(S_t,A_t) + \alpha\left[R_{t+1} + \gamma Q_1(S_{t+1}, \text{argmax}_a Q_2(S_{t+1},a)) - Q_2(S_t,A_t)\right] \end{cases}$$

Double Learning

Algorithm:

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 $Q(terminal, \cdot) = 0$

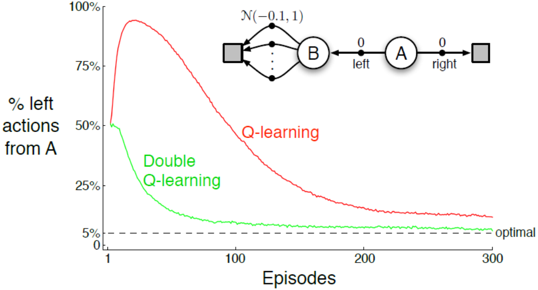
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    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

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

Maximization Bias Example

$\varepsilon = 0.1$, $\alpha = 0.1$, and $\gamma = 1$. Average over 10,000 runs:



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

n-step TD: Estimating $v_\pi(s)$

Let

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}),$$

where $V_t(s)$ denotes the estimate of $v_\pi(s)$ at time t.

Convention:
if $t + n \geq T$, then all missing terms are zero
(in this case $G_{t:(t+n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T = G_t$).

The <u>n-step TD</u> updates are

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_{t:(t+n)} - V_{t+n-1}(S_t) \right]$$

n-step TD: Estimating $v_\pi(s)$

n-step TD

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_{t:(t+n)} - V_{t+n-1}(S_t) \right],$$

where $G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$.

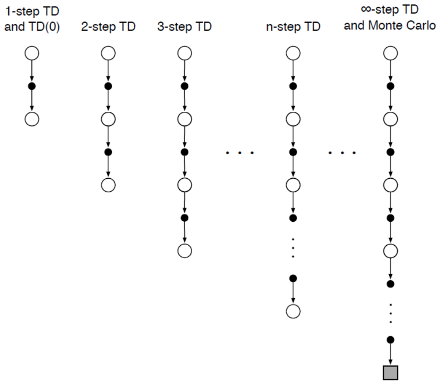
Notice that

1. if $n = 1$, then $V_{t+1}(S_t) \leftarrow V_t(S_t) + \alpha \left[G_{t:(t+1)} - V_t(S_t) \right]$,
   where $G_{t:(t+1)} = R_{t+1} + \gamma V_t(S_{t+1})$,
   i.e. one-step TD method!
2. if $t + n \geq T$, then

   $$V_{t+1}(S_t) \leftarrow V_t(S_t) + \alpha \left[G_t - V_t(S_t) \right],$$

   where $G_{t:(t+n)} = G_t$,
   i.e. constant-α MC method!

Backup Diagrams of n-step TD



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

n-step TD Algorithm: Estimating $v_\pi(s)$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_{t:(t+n)} - V_{t+n-1}(S_t)\right]$$
$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$$

Algorithm:

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n + 1$

```
Loop for each episode:
   Initialize and store S_0 ≠ terminal
   T ← ∞
   Loop for t = 0, 1, 2, ... :
   |  If t < T, then:
   |     Take an action according to π(·|S_t)
   |     Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
   |     If S_{t+1} is terminal, then T ← t + 1
   |  τ ← t − n + 1   (τ is the time whose state's estimate is being updated)
   |  If τ ≥ 0:
   |     G ← Σ_{i=τ+1}^{min(τ+n,T)} γ^{i−τ−1} R_i
   |     If τ + n < T, then: G ← G + γ^n V(S_{τ+n})                  (G_{τ:τ+n})
   |     V(S_τ) ← V(S_τ) + α [G − V(S_τ)]
   Until τ = T − 1
```

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

n-step SARSA: Estimating $q_\pi(s,a)$ or $q_*(s,a)$

Let

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}),$$

where $Q_t(s,a)$ denotes the estimate of $q_\pi(s)$ at time t.

Convention:
if $t+n \geq T$, then all missing terms are zero
(in this case $G_{t:(t+n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T = G_t$).

The n-step SARSA updates are

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:(t+n)} - Q_{t+n-1}(S_t, A_t) \right]$$

n-step SARSA: Estimating $q_\pi(s,a)$ or $q_*(s,a)$

n-step SARSA

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:(t+n)} - Q_{t+n-1}(S_t, A_t)\right],$$

where

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}).$$

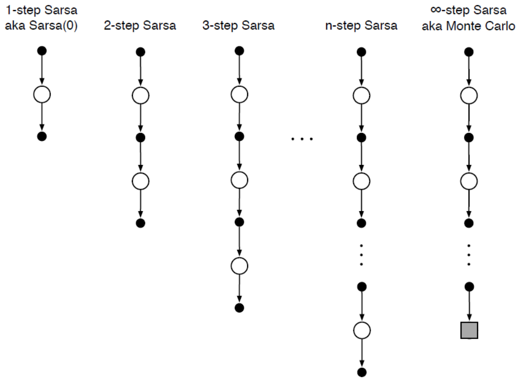
Notice that

1. if $n = 1$, then $Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha \left[G_{t:(t+1)} - Q_t(S_t, A_t)\right]$,
   where $G_{t:(t+1)} = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1})$,
   i.e. (one-step) SARSA method!
2. if $t + n \geq T$, then

   $$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha \left[G_t - Q_t(S_t, A_t)\right],$$

   where $G_{t:(t+n)} = G_t$,
   i.e. constant-α MC method ("∞-step SARSA")!

Backup Diagrams of n-step SARSA



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

n-step SARSA Algorithm: Estimating $q_\pi(s,a)$ or $q_*(s,a)$

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha\left[G_{t:(t+n)} - Q_{t+n-1}(S_t, A_t)\right]$$
$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$$

Algorithm:

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n+1$

```
Loop for each episode:
   Initialize and store S_0 ≠ terminal
   Select and store an action A_0 ~ π(·|S_0)
   T ← ∞
   Loop for t = 0, 1, 2, ... :
   |  If t < T, then:
   |      Take action A_t
   |      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
   |      If S_{t+1} is terminal, then:
   |          T ← t + 1
   |      else:
   |          Select and store an action A_{t+1} ~ π(·|S_{t+1})
   |  τ ← t − n + 1    (τ is the time whose estimate is being updated)
   |  If τ ≥ 0:
   |      G ← Σ_{i=τ+1}^{min(τ+n,T)} γ^{i−τ−1} R_i
   |      If τ + n < T, then G ← G + γ^n Q(S_{τ+n}, A_{τ+n})            (G_{τ:τ+n})
   |      Q(S_τ, A_τ) ← Q(S_τ, A_τ) + α [G − Q(S_τ, A_τ)]
   |      If π is being learned, then ensure that π(·|S_τ) is ε-greedy wrt Q
   Until τ = T − 1
```

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

n-step Off-policy TD: Estimating $v_\pi(s)$

Let

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}),$$

where $V_t(s)$ denotes the estimate of $v_\pi(s)$ at time t.

Convention:
if $t + n \geq T$, then all missing terms are zero
(in this case $G_{t:(t+n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T = G_t$).

The n-step off-policy TD updates are

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[\rho_{t:(t+n-1)} G_{t:(t+n)} - V_{t+n-1}(S_t)\right],$$

where

$$\rho_{t:h} \doteq \prod_{k=t}^{\min\{h,T-1\}} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$

Contents

n-step Off-policy SARSA: Estimating $q_\pi(s,a)$ or $q_*(s,a)$

Let

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}),$$

where $Q_t(s,a)$ denotes the estimate of $q_\pi(s,a)$ at time t.

Convention:
if $t+n \geq T$, then all missing terms are zero
(in this case $G_{t:(t+n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T = G_t$).

The n-step SARSA updates are

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \left[\rho_{(t+1):(t+n)} G_{t:(t+n)} - Q_{t+n-1}(S_t, A_t)\right],$$

where

$$\rho_{t:h} \doteq \prod_{k=t}^{\min\{h, T-1\}} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$