CSCI E-89C Deep Reinforcement Learning

Harvard Summer School

Dmitry Kurochkin

Summer 2020
Lecture 7

Contents

Contents

Quiz 6

Question 1

4 / 4 pts

The environment has three states: s_A, s_B, and s_C. In each state there are two actions, a_1 and a_2, available.

Suppose we want to estimate $v_\pi(s)$ using the 1-step TD learning, where the policy π is to select action a_1 in all states.

We generate the sequence under this policy π and observe:

$s_A, a_1, R_1 = 5, s_B, \ldots$

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values are $V(s_A) = 1$, $V(s_B) = 2$, and $V(s_C) = 3$, what is $V(s_A)$ after the first update according to the 1-step TD method?

Hint: use the following n-step on-policy TD updates:

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

with $n = 1$.

Correct!

1.58

Quiz 6

Question 1

4 / 4 pts

The environment has three states: s_A, s_B, and s_C. In each state there are two actions, a_1 and a_2, available.

Suppose we want to estimate $v_\pi(s)$ using the 1-step TD learning, where the policy π is to select action a_1 in all states.

We generate the sequence under this policy π and observe:

$s_A, a_1, R_1 = 5, s_B, \ldots$

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values are $V(s_A) = 1$, $V(s_B) = 2$, and $V(s_C) = 3$, what is $V(s_A)$ after the first update according to the 1-step TD method?

$$
\begin{aligned}
V_1(S_0) &= V_0(S_0) + \alpha\left[\overbrace{R_1 + \gamma V_0(S_1)}^{G_{0:1}} - V_0(S_0)\right] \\
&= V_0(s_A) + \alpha\left[R_1 + \gamma V_0(s_B) - V_0(s_A)\right] \\
&= 1 + 0.1\left[5 + 0.9 \cdot 2 - 1\right] \\
&= 1.58
\end{aligned}
$$

Quiz 6

Question 2

4 / 4 pts

If the environment is stationary, the TD control SARSA algorithm,

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

should use non-constant learning rate α. Specifically, $1/N(S_t, A_t)$, where $N(S_t, A_t)$ is the number of times the state-action pair (S_t, A_t) was visited, would result in the most efficient algorithm.

- True
- False

Correct!

Quiz 6

Question 3 4 / 4 pts

Q-learning is unbiased.

- True
- Correct! False

Quiz 6

Question 4

4 / 4 pts

Double Q-learning requires twice more iterations than Q-learning.

- True
- **Correct!** False

Quiz 6

Question 5

4 / 4 pts

The environment has two states: s_A and s_B. In each state there are two actions, a_1 and a_2, available.

The sample below is generated under policy b with $b(a_1|s_A) = 0.2$ and $b(a_1|s_B) = 0.3$:

$s_A, a_1, R_1 = 7, s_B, \ldots$

Suppose we want to estimate $v_\pi(s)$ using the 1-step off-policy TD learning, where the policy π is to select action a_1 in all states.

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values are $V(s_A) = 10$ and $V(s_B) = 20$, what is $V(s_A)$ after the first update?

Hint: use the following n-step off-policy TD updates:

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

with $n = 1$.

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

Correct!

21.5

Quiz 6

Question 5

4 / 4 pts

The environment has two states: s_A and s_B. In each state there are two actions, a_1 and a_2, available.

The sample below is generated under policy b with $b(a_1|s_A) = 0.2$ and $b(a_1|s_B) = 0.3$:

$s_A, a_1, R_1 = 7, s_B, \ldots$

Suppose we want to estimate $v_\pi(s)$ using the 1-step off-policy TD learning, where the policy π is to select action a_1 in all states.

If $\alpha = 0.1, \gamma = 0.9$, and initial values are $V(s_A) = 10$ and $V(s_B) = 20$, what is $V(s_A)$ after the first update?

$$
\begin{aligned}
V_1(S_0) &= V_0(S_0) + \alpha\left[\overbrace{\frac{\pi(A_0|S_0)}{b(A_0|S_0)}}^{\rho_{0:0}}\overbrace{\Big(R_1 + \gamma V_0(S_1)\Big)}^{G_{0:1}} - V_0(S_0)\right] \\
&= V_0(s_A) + \alpha\left[\frac{\pi(a_1|s_A)}{b(a_1|s_A)}\Big(R_1 + \gamma V_0(s_B)\Big) - V_0(s_A)\right] \\
&= 10 + 0.1\left[\frac{1}{0.2}\Big(7 + 0.9 \cdot 20\Big) - 10\right] \\
&= 21.5
\end{aligned}
$$

Contents

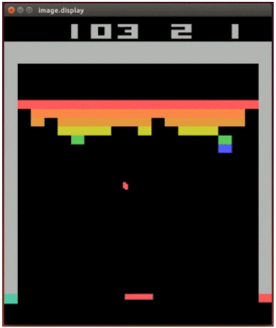
Tabular Methods

Tabular methods require to estimate

1. $|\mathcal{S}|$ entries of $v_\pi(s)$ in case of prediction
2. $\sum_{s \in \mathcal{S}} |\mathcal{A}(s)|$ entries of $q_*(s, a)$ in case of control (i.e. $\propto |\mathcal{S}| \times |\mathcal{A}|$)

Contents

Example: Atari Breakout



Example: Atari Breakout

The solution via approximation of $q_*(s, a)$:

NIPS 2013, DeepMind, Playing Atari with Deep Reinforcement Learning, `https://arxiv.org/abs/1312.5602`

Contents

Function Approximations

Given policy π, assume that for some weights $\mathbf{w} = (w_1, w_2, \ldots, w_d)^T$ (usually $d \ll |\mathcal{S}|$) we can approximate:

$$v_\pi(s) \approx \hat{v}(s, \mathbf{w}).$$

Function Approximations

Given policy π, assume that for some weights $\mathbf{w} = (w_1, w_2, \ldots, w_d)^T$ (usually $d \ll |\mathcal{S}|$) we can approximate:

$$v_\pi(s) \approx \hat{v}(s, \mathbf{w}).$$

Examples:

1. $\mathcal{S} = \{s_A, s_B, s_C\}$:

$$\hat{v}(s, \mathbf{w}) = w_1 \cdot \mathbb{1}_{(s=s_A)} + w_2 \cdot \mathbb{1}_{(s=s_B)} + w_3 \cdot \mathbb{1}_{(s=s_C)}$$

2. $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$:

$$\hat{v}(s_k, \mathbf{w}) = w_1 + w_2 \cdot k \quad \text{for all } k \in \{1, 2, \ldots, n\}$$

Contents

Stochastic gradient Descent (SGD) Method

Given policy π, assume that for some weights $\mathbf{w} = (w_1, w_2, \ldots, w_d)^T$ (usually $d \ll |\mathcal{S}|$) we can approximate:

$$v_\pi(s) \approx \hat{v}(s, \mathbf{w}).$$

Stochastic gradient Descent (SGD) Method

Given policy π, assume that for some weights $\mathbf{w} = (w_1, w_2, \ldots, w_d)^T$ (usually $d \ll |\mathcal{S}|$) we can approximate:

$$v_\pi(s) \approx \hat{v}(s, \mathbf{w}).$$

The Stochastic gradient descent (SGD) method that minimizes the mean-squared error

$$J(\mathbf{w}) \doteq E_\pi \left[\left(v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}) \right)^2 \right]$$

is then

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \frac{1}{2} \alpha \nabla \left[v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t) \right]^2 \\ &= \mathbf{w}_t + \alpha \left[v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t) \end{aligned}$$

Stochastic gradient Descent (SGD) Method

Since we do not know $v_\pi(S_t)$, we use an approximation U_t of the state value function (for example G_t in case of MC). The weights then can be obtained as follows:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{U_t}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

SGD Constant-α MC for Estimating v_π

Let $U_t \doteq G_t$:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{G_t}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

Algorithm:

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[G_t - \hat{v}(S_t, \mathbf{w}) \right] \nabla \hat{v}(S_t, \mathbf{w})$

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

Semi-gradient 1-step TD for Estimating v_π

Let $U_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t)$:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t)}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla\hat{v}(S_t, \mathbf{w}_t)$$

Algorithm:

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S is terminal

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

Semi-gradient n-step TD for Estimating v_π

Let $U_t \doteq G_{t:(t+n)}(\mathbf{w}_{t+n-1})$, where
$G_{t:(t+n)}(\mathbf{w}_{t+n-1}) \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1})$:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \left[\underbrace{G_{t:(t+n)}(\mathbf{w}_{t+n-1})}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_{t+n-1}) \right] \nabla \hat{v}(S_t, \mathbf{w}_{t+n-1})$$

Algorithm:

n-step semi-gradient TD for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal}, \cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, a positive integer n
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (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 v̂(S_{τ+n}, w)          (G_{τ:τ+n})
   |     w ← w + α [G − v̂(S_τ, w)] ∇v̂(S_τ, w)
   Until τ = T − 1
```

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

Contents

Semi-gradient SARSA for Estimating q_*

Let $U_t \doteq R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t)$:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t)}_{\approx q_\pi(S_t, A_t)} - \hat{q}(S_t, A_t, \mathbf{w}_t) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

Algorithm:

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        If S' is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[R - \hat{q}(S, A, \mathbf{w})\big] \nabla \hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[R + \gamma \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big] \nabla \hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

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

Semi-gradient n-step SARSA for Estimating q_*

Let $U_t \doteq G_{t:(n+1)}(\mathbf{w}_{t+n-1})$, where
$G_{t:(t+n)}(\mathbf{w}_{t+n-1}) \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1})$:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \left[\underbrace{G_{t:(t+n)}(\mathbf{w}_{t+n-1})}_{\approx q_\pi(S_t, A_t)} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})$$

Algorithm:

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_*$ or q_π

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | 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:
  | $T \leftarrow t + 1$
  | else:
  | Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $(G_{\tau:\tau+n})$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})\right] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$

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