CSCI E-89C Deep Reinforcement Learning

Harvard Summer School

Dmitry Kurochkin

Summer 2020
Lecture 5

Contents

Contents

Target Policy v.s. Behavior Policy

1. The goal is to learn *target* policy $\pi(a|s)$, usually deterministic greedy policy
2. Off-policy methods estimate *target* policy using data generated according to *behavior* policy $b(a|s)$
3. Off-policy methods usually have larger variance and slower convergence
4. On-policy learning, i.e. whenever $b(a|s) = \pi(a|s)$, is a special case of off-policy
5. Off-policy methods can be used to learn from available data generated by a non-learning controller
6. Assumption of <u>coverage</u>: $\pi(a|s) > 0 \;\Rightarrow\; b(a|s) > 0$

Contents

Importance Sampling

If we start in state S_t and follow policy π, then

$$
\begin{aligned}
&P_\pi\{A_t, S_{t+1}, A_{t+1}, \dots, S_T | S_t\} \\
&\quad = P_\pi\{A_t|S_t\} P_\pi\{S_{t+1}, A_{t+1}, \dots, S_T | S_t, A_t\} \\
&\quad = P_\pi\{A_t|S_t\} P_\pi\{S_{t+1}|S_t, A_t\} P_\pi\{A_{t+1}, \dots, S_T | S_t, A_t, S_{t+1}\} \\
&\quad = \pi(A_t|S_t) p(S_{t+1}|S_t, A_t) P_\pi\{A_{t+1}, \dots, S_T | S_{t+1}\} \\
&\quad\ \vdots \\
&\quad = \pi(A_t|S_t) p(S_{t+1}|S_t, A_t) \pi(A_{t+1}|S_{t+1}) \dots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)
\end{aligned}
$$

Importance Sampling

For any policy $\pi(a|s)$:

$$P_\pi\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T | S_t\} = \prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)$$

Importance Sampling

For any policy $\pi(a|s)$:

$$P_\pi\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T | S_t\} = \prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)$$

Define the importance-sampling ratio:

$$\begin{aligned}\rho_{t:(T-1)} &\doteq \frac{P_\pi\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T | S_t\}}{P_b\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T | S_t\}} \\ &= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} \\ &= \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}\end{aligned}$$

Given trajectories, $\rho_{t:(T-1)}$ depends on the policies only!

Importance Sampling

We notice that under policy b,

$$E_b\left[G_t|S_t = s\right] = v_b(s),$$

but the expected transformed cumulative discounted return is

$$E_b\left[\rho_{t:(T-1)}G_t|S_t = s\right] = E_\pi\left[G_t|S_t = s\right] = v_\pi(s),$$

where data were generated under b.

Contents

Off-policy Estimation of $v_\pi(s)$

Let index t run through episodes. In order to estimate $v_\pi(s)$, the agent can follow policy $b \neq \pi$ but transform the observations for G_t's as follows:

1. Ordinary importance sampling (unbiased in case of first-visit MC):

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:(T-1)} G_t}{|\mathcal{T}(s)|}.$$

2. Weighted importance sampling (biased):

$$V(s) \doteq \begin{cases} \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:(T-1)} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:(T-1)}}, & \text{if } \sum_{t \in \mathcal{T}(s)} \rho_{t:(T-1)} \neq 0, \\ 0 & \text{, otherwise.} \end{cases}$$

Here, $\mathcal{T}(s)$ is either
(a) set of all time steps t in which state s is first visited (first-visit MC); or
(b) set of all time steps t in which state s is visited (every-visit MC).

Off-policy Estimation of $q_\pi(s,a)$

Let index t run through episodes. In order to estimate $q_\pi(s,a)$, the agent can follow policy $b \neq \pi$ but transform the observations for G_t's as follows:

1. Ordinary importance sampling (unbiased in case of first-visit MC):

$$Q(s,a) \doteq \frac{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:(T-1)} G_t}{|\mathcal{T}(s,a)|}.$$

2. Weighted importance sampling (biased):

$$Q(s,a) \doteq \begin{cases} \frac{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:(T-1)} G_t}{\sum_{t\in\mathcal{T}(s,a)} \rho_{t:(T-1)}}, & \text{if } \sum_{t\in\mathcal{T}(s,a)} \rho_{t:(T-1)} \neq 0, \\ 0 & , \text{ otherwise.} \end{cases}$$

Here, $\mathcal{T}(s,a)$ is either
(a) set of all time steps t in which the pair (s,a) is first visited (first-visit MC); or
(b) set of all time steps t in which the pair (s,a) is visited (every-visit MC).

Contents

Incremental Implementation

Let's fix pair (s, a). We need to estimate $q(s, a)$ from the sequence of returns

$$G_1, G_2, \ldots, G_k, \ldots, G_{n-1},$$

where k represents the k-th visit to pair (s, a):

$$Q_n \doteq \frac{\sum_{k=1}^{n-1} w_k G_k}{\sum_{k=1}^{n-1} w_k}, n \geq 2.$$

Here, $w_k = \rho_{t_k:(T(t_k)-1)}$ are the corresponding weights, where t_k is the time step in which (s, a) is visited k-th time.

The the updating rule is

$$Q_{n+1} \doteq Q_n + \frac{w_n}{C_n}\left[G_n - Q_n\right], n \geq 1$$

$$C_{n+1} \doteq C_n + w_{n+1}$$

Off-policy Estimation of $\pi_*(s, a)$

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
    $b \leftarrow$ any soft policy
    Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
        If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
        $W \leftarrow W \frac{1}{b(A_t|S_t)}$

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

Contents

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

Recall that the state-value function is defined as follows:

$$v_\pi(s) \doteq E_\pi\left[G_t | S_t = s\right].$$

Every-visit MC method for nonstationary environment (*constant*-α MC):

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right],\ \alpha \in (0, 1],$$

i.e. <u>need</u> to wait until the end of the episode because of G_t.

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

Recall that the state-value function is defined as follows:

$$v_\pi(s) \doteq E_\pi\left[G_t | S_t = s\right].$$

Every-visit MC method for nonstationary environment (*constant*-α MC):

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right], \ \alpha \in (0, 1],$$

i.e. need to wait until the end of the episode because of G_t.

Also, recall that

$$v_\pi(s) = E_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right].$$

Then at time $t + 1$ can update as follows (*one-step* TD or TD(0)):

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right], \ \alpha \in (0, 1],$$

i.e. no need to wait until the end of the episode!

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

One-step TD prediction:

$$V(S_t) \leftarrow V(S_t) + \alpha \underbrace{\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]}_{\doteq \delta_t}.$$

TD error, δ_t, is available at time time $t+1$.

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

One-step TD prediction:

$$V(S_t) \leftarrow V(S_t) + \alpha \underbrace{\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]}_{\doteq \delta_t}.$$

TD error, δ_t, is available at time time $t+1$.

Note that the MC estimate $V(S_t)$ does not change over the episode and the MC error is related to δ_t as follows:

$$\begin{aligned}
G_t - V(S_t) =& R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
=& \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] + \gamma \left(G_{t+1} - V(S_{t+1})\right) \\
=& \delta_t + \gamma \left(G_{t+1} - V(S_{t+1})\right) \\
& \vdots \\
=& \delta_t + \gamma \delta_{t+1} + \ldots + \gamma^{T-t-1} \delta_{T-1} = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k.
\end{aligned}$$

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

One-step TD prediction:

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

Algorithm:

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 $V(terminal) = 0$

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'
    $V(S) \leftarrow V(S) + \alpha\left[R + \gamma V(S') - V(S)\right]$
    $S \leftarrow S'$
  until S is terminal

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

Contents

Advantages of TD Methods

One-step TD prediction:

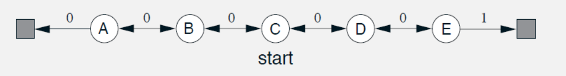
$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right].$$

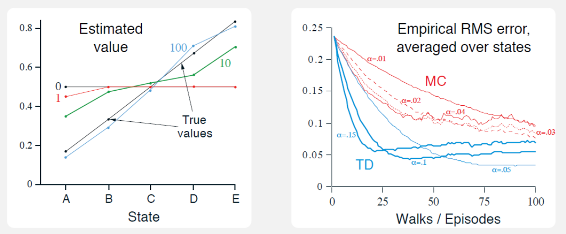
1. Model-free
2. No need to wait until the end of the episode - TD may converge substantially faster
3. Can be applied to continuing tasks with no episodes
4. MC must discount some of the episodes - TD may converge substantially faster
5. Given the step-size parameter α is sufficiently small, TD(0) converges to v_π

Contents

Random Walk Example

In the example below the only action in all states is "wait." All transitions are equally likely. When the random walk terminates on the right, the reward is 1, otherwise all rewards are 0.





Left: TD(0) with $\alpha = 0.1$ results for 0, 1, 10, and 100 episodes.
Right: Learning curves for MC and TD(0) and various step-size parameters α.

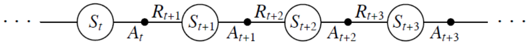
Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

SARSA: Estimation of $q_*(s, a)$

Similarly to TD prediction, the updating rule for $Q(S_{t+1}, A_{t+1})$

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



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

SARSA: Estimation of $q_*(s, a)$

Similarly to TD prediction, the updating rule for $Q(S_{t+1}, A_{t+1})$

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

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 $Q(\mathit{terminal}, \cdot) = 0$

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

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