November 19, 2022

10 Exercises

10.1 Question

We have not explicitly considered or given pseudocode for any Monte Carlo methods or in this chapter. What would they be like? Why is it reasonable not to give pseudocode for them? How would they perform on the Mountain Car task?

Answer

Monte Carlo method is just an extreme case of n-step methods. One can achieve MC effect by using a sufficiently large n value.

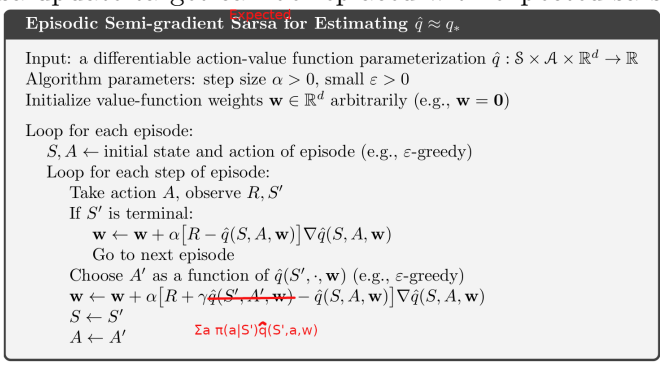
MC is slower to learn. MC cannot start learning before an episode ends.

10.2 Question

Give pseudocode for semi-gradient one-step Expected Sarsa for control.

Answer

We are already given a pseudocode for semi-gradient one-step sarsa control. Sarsa update target can be replaced with expected sarsa update target.



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)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

10.3 Question

Why do the results shown in Figure 10.4 have higher standard errors at large n than at small n?

Answer

The answer is similar to TD methods vs MC methods debate.

In n-step learning, intermediate values of n works best. n-values less than the optimum n value, tends to behave like TD(0) while n-values larger than the optimum n value, tends to behave like MC.

MC method updates are known to be high variance due to large number of steps thus large number of noise involved.

10.4 Question

Give pseudocode for a differential version of semi-gradient Q-learning.

Answer

We are already given a pseudocode for a differential version of semi-gradient sarsa. Sarsa update target can be replaced with expected q-learning update target.

Differential semi-gradient ~~Sarsa~~ Q-learning 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 sizes $\alpha, \beta > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)

Initialize state S, ~~and action A~~
Loop for each step:
  Select action A as argmaxa $\hat{q}$(S,a,w)
  Take action A, observe R, S'
  ~~Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)~~
  $\delta \leftarrow R - \bar{R} + \max_a \hat{q}(S', a, \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$

10.5 Question

What equations are needed (beyond 10.10) to specify the differential version of TD(0)?

Answer

10.10 gives differential form of the two TD errors. 10.12 gives the average reward version of semi-gradient Sarsa. We can leverage those two equations to obtain differential version of TD(0).

$w_{t+1} = w_t + \alpha\delta_t\Delta\hat{V}(S_t, w_t)$

10.6 Question

Consider a Markov reward process consisting of a ring of three states A, B,and C, with state transitions going deterministically around the ring. A reward of +1 is received upon arrival in A and otherwise the reward is 0. What are the differential values of the three states?

Answer

Ring of 3 states.

$\mathbf{A} \to 0 \to \mathbf{B} \to 0 \to \mathbf{C} \to +1 \to \mathbf{A}$

Average reward as per 10.7:

$r(\pi) = \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)r$

$r(\pi) = 1/3 * 1.0 * 1.0 * 0 + 1/3 * 1.0 * 1.0 * 0 + 1/3 * 1.0 * 1.0 * 1 = 1/3$

State values as per 10.9+:

$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r - r(\pi) + v_\pi(s')]$

$v_\pi(A) = 0 - 1/3 + v_\pi(B) = v_\pi(B) - 1/3$

$v_\pi(B) = 0 - 1/3 + v_\pi(C) = v_\pi(C) - 1/3$

$v_\pi(C) = 1 - 1/3 + v_\pi(A) = v_\pi(A) + 2/3$

Linear equations turns out to have many solutions e.g. a plane: Let's pick up a value for state B from availability space as $V(B) = 0$. Then:

$V(A) = -1/3$

$V(C) = 1/3$

10.7 Question

. Suppose there is an MDP that under any policy produces the deterministic sequence of rewards $+1, 0, +1, 0, +1, 0, \ldots$ going on forever. Technically, this is not allowed because it violates ergodicity; there is no stationary limiting distribution μ_π and the limit (10.7) does not exist. Nevertheless, the average reward (10.6) is well defined; What is it? Now consider two states in this MDP. From A, the reward sequence is exactly as described above, starting with a $+1$, whereas, from B, the reward sequence starts with a 0 and then continues with $+1, 0, +1, 0, \ldots$. The differential return (10.9) is not well defined for this case as the limit does not exist. To repair this, one could alternately define the value of a state as

$$v_\pi(s) \doteq \lim_{\gamma \to 1} \lim_{h \to \infty} \sum_{t=0}^{h} \gamma^t \Big(\mathbb{E}_\pi[R_{t+1} | S_0 = s] - r(\pi) \Big). \tag{10.13}$$

Under this definition, what are the values of states A and B? □

Answer

Average reward as per 10.6:
$r(\pi) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} E[R_t|S_0, A_0\pi]$
$r(\pi) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h/2} E[R_{2t-1}|S_0, A_0\pi] + E[R_{2t}|S_0, A_0\pi]$
$r(\pi) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h/2} 1 + 0$
$r(\pi) = \lim_{h\to\infty} \frac{1}{h} \frac{h}{2} = 0.5$
Now consider two states in this MDP, as per 10.13:
$v_\pi(s) = \lim_{\gamma\to 1} \lim_{h\to\infty} \sum_{t=0}^{h} \gamma^t (E[R_{t+1}|S_0 = s] - r(\pi))$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} \sum_{t=0}^{h} \gamma^t (E[R_{t+1}|S_0 = A] - r(\pi))$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} \sum_{t=0}^{h/2} \gamma^{2t} (E[R_{2t+1}|S_0 = A] - r(\pi)) + \gamma^{2t+1} (E[R_{2t+2}|S_0 = A] - r(\pi))$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} \sum_{t=0}^{h/2} \gamma^{2t} 0.5 + \gamma^{2t+1}(-0.5)$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} \sum_{t=0}^{h/2} \gamma^{2t} 0.5(1-\gamma)$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} 0.5(1-\gamma) \sum_{t=0}^{h/2} \gamma^{2t}$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} 0.5(1-\gamma) \sum_{t=0}^{h/2} (\gamma^2)^t$
there we have a geometric series of form $\frac{1-x^{n+1}}{1-x} = \sum_{k=0}^{n} x^k$
$v_\pi(A) = \lim_{\gamma\to 1} \lim_{h\to\infty} 0.5(1-\gamma) \frac{1-(\gamma^2)^{h/2+1}}{1-\gamma^2}$
We have $\lim_{h\to\infty} (\gamma^2)^{h/2+1} = \gamma^{\inf} = 0$
$v_\pi(A) = \lim_{\gamma\to 1} 0.5(1-\gamma) \frac{1}{1-\gamma^2}$
$v_\pi(A) = \lim_{\gamma\to 1} 0.5 \frac{1}{1+\gamma}$
$v_\pi(A) = 0.5 \frac{1}{2} = 0.25$

Calculation for state B is similar:

$v_\pi(B) = \lim_{\gamma \to 1} \lim_{h \to \infty} \sum_{t=0}^{h/2} \gamma^{2t}(E[R_{2t+1}|S_0 = A] - r(\pi)) + \gamma^{2t+1}(E[R_{2t+2}|S_0 = A] - r(\pi))$
$v_\pi(B) = \lim_{\gamma \to 1} \lim_{h \to \infty} \sum_{t=0}^{h/2} \gamma^{2t}(-0.5) + \gamma^{2t+1}0.5$
$v_\pi(B) = \lim_{\gamma \to 1} \lim_{h \to \infty} -1 \sum_{t=0}^{h/2} \gamma^{2t}0.5 - \gamma^{2t+1}0.5$
Which makes:
$v_\pi(B) = -1 v_\pi(A) = -0.25$

10.8 Question

The pseudocode in the box on page 251 updates $\bar{R}_{t+1}$ using δ_t as an error rather than simply $R_{t+1} - \bar{R}_{t+1}$. Both errors work, but using δ_t is better. To see why, consider the ring MRP of three states from Exercise 10.6. The estimate of the average reward should tend towards its true value of $\frac{1}{3}$. Suppose it was already there and was held stuck there. What would the sequence of $R_t - R_t$ errors be? What would the sequence of δ_t errors be (using (10.10))? Which error sequence would produce a more stable estimate of the average reward if the estimate were allowed to change in response to the errors? Why? □

Answer

Using state values from 10.6. v(A) = -1/3, v(B) = 0, v(C) = 1/3.
Average reword $\bar{R} = \frac{1}{3}$ as given in the question.

Let's calculate average reward using 10.10 after each state transition.
$\delta = R - \bar{R} + \hat{q}(S', A', w) - \hat{q}(S', A, w)$ $\bar{R} = \bar{R} + \beta\delta$
State A, transition to state B
$\delta = 0 - \frac{1}{3} + 0 - (-\frac{1}{3}) = 0$
State B, transition to state C
$\delta = 0 - \frac{1}{3} + \frac{1}{3} - 0 = 0$
State C, transition to state A
$\delta = 1 - \frac{1}{3} + (-\frac{1}{3}) - \frac{1}{3} = 0$

If we were to use the simpler update, then update amounts would be:
State A, transition to state B
$R - \bar{R} = 0 - \frac{1}{3} = -\frac{1}{3}$
State B, transition to state C
$R - \bar{R} = 0 - \frac{1}{3} = -\frac{1}{3}$
State C, transition to state A
$R - \bar{R} = 1 - \frac{1}{3} = \frac{2}{3}$

The simpler update rule changes what was supposed to be true value of the average reward. TD error does not change the average reward. Thus

TD error based update produce a more stable estimate. Because the simpler update rule does not care about convergence of the average reward. TD error based update rule on the other hand slows down if close to convergence.

10.9 Question

In the differential semi-gradient n-step Sarsa algorithm, the step-size parameter on the average reward, β, needs to be quite small so that $\bar{R}$ becomes a good long-term estimate of the average reward. Unfortunately, $\bar{R}$ will then be biased by its initial value for many steps, which may make learning inefficient. Alternatively, one could use a sample average of the observed rewards for $\bar{R}$. That would initially adapt rapidly but in the long run would also adapt slowly. As the policy slowly changed, $\bar{R}$ would also change; the potential for such long-term nonstationarity makes sample-average methods ill-suited. In fact, the step-size parameter on the average reward is a perfect place to use the unbiased constant-step-size trick from Exercise 2.7. Describe the specific changes needed to the boxed algorithm for differential semi-gradient n-step Sarsa to use this trick. □

Answer

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

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$, a policy π
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average-reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Algorithm parameters: step size $\alpha, \beta > 0$, a positive integer n
All store and access operations (S_t, A_t, and R_t) can take their index mod $n + 1$
Initialize and store S_0 and A_0
Loop for each step, $t = 0, 1, 2, \ldots$:
  Take action A_t
  Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$, 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 $\tau \geq 0$:
    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n} (R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$