**Chapter 10 solutions** **Hanna Gábor**

1. *We have not explicitly considered or given pseudocode for any Monte Carlo methods 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?*

   The Monte Carlo is basically $n$-step Sarsa with $n = T$. I expect it to perform poorly on the Mountain Car example: until it doesn't reach the goal, it doesn't learn anything, so the first episode might be really long. As opposed to this, $n$-step Sarsa tries new actions after it sees that the previous actions didn't lead to the end of the episode, so it will get to the goal line eventually.

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

**Algorithm 1** Semi-gradient expected Sarsa for control

Inputs: a differentiable action-value function parametrization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: steps size $\alpha > 0, \epsilon > 0$
Initialize value-function weights $w \in \mathbb{R}^d$ arbitrarily
**for** each episode **do**
  Initialize $S_0$ non-terminal state.
  **for** $t = 0, 1, 2 \ldots$ **do**
    Take action $A_t$ according to the $\epsilon$-greedy policy w.r.t. $\hat{q}(S_0, \cdot, w)$
    Observe reward $R_{t+1}$ and next state $S_{t+1}$.
    **if** $S_{t+1}$ is terminal **then**
      $w \leftarrow w + \alpha R_{t+1} \nabla \hat{q}(S_t, A_t, w)$
      go to next episode
    **else**
      $w \leftarrow w + \alpha \Big( R_{t+1} + \sum_a \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, w) - \hat{q}(S_t, A_t, w) \Big) \nabla \hat{q}(S_t, A_t, w)$, where
      $\pi$ is the $\epsilon$-greedy policy w.r.t. $\hat{q}(S_0, \cdot, w)$
      $S_t \leftarrow S_{t+1}$
    **end if**
  **end for**
**end for**

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

   If $n$ is large, it's less predictable where we will end up in $n$ steps. This causes our estimates to have a bigger standard deviation. That in turn, will cause the steps to be more varied among different runs.

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

**Algorithm 2** Differential version of semi-gradient Q-learning

Inputs: a differentiable action-value function parametrization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: steps size $\alpha > 0, \epsilon > 0$
Initialize value-function weights $w \in \mathbb{R}^d$ arbitrarily.
$\bar{R} \leftarrow 0$
$visits \leftarrow 0$
**for** each episode **do**
  Initialize $S_0$ non-terminal state.
  **for** $t = 0, 1, 2 \ldots$ **do**
    Choose action $A_t$ according to the $\epsilon$-greedy policy w.r.t. $\hat{q}$
    Observe reward $R_{t+1}$ and next state $S_{t+1}$.
    $w \leftarrow w + \alpha(R_{t+1} - \bar{R} + \max_a(\hat{q}(S_{t+1}, a, w)) - \hat{q}(S_t, A_t, w))\nabla\hat{q}(S_t, A_t, w)$
    $\bar{R} \leftarrow \frac{visits}{visits+1}(\bar{R} + R_{t+1})$
    $visits \leftarrow visits + 1$
    **if** $S_{t+1}$ is terminal **then**
      Go to next episode.
    **end if**
  **end for**
**end for**