RL Ch10 On-policy Control with Approximation

Le-Rong Hsu

July 2023

1 Outline

- semi-gradient
- average-reward setting
- semi-gradient + average-reward setting
- example

2 Episodic Semi-gradient Control

Prediction

We now consider examples of the form $S_t, A_t \to U_t$. The update target U_t can be any approximation of $q_\pi(S_t, A_t)$, including full Monte Carlo returns G_t or n-step Sarsa returns $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$. The general gradient-descent update for action-value prediction is

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)]\nabla \hat{q}(S_t, A_t, \mathbf{w}_t) \tag{10.1}$$

Control

- continuous/large discrete state set $\to$ no general resolution
- not too large $\to A^*_{t+1} = \arg\max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t)$

One-step Semi-gradient Sarsa

The update for the one-step Sarsa method is

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)]\nabla \hat{q}(S_t, A_t, \mathbf{w}_t) \tag{10.2}$$

We call this method **episodic semi-gradient one-step Sarsa**.

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

Semi-gradient n-step Sarsa

The n-step return:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}) \quad (10.4)$$

with $G_{t:t+n} = G_t$ if $t + n \geq T$. The n-step update equation is

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})]\nabla\hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), 0 \leq t < T \quad (10.5)$$

Remark. Expected Sarsa: $\gamma\hat{q}(S', A', \mathbf{w}) \to \gamma \sum_a \pi(a|S')\hat{q}(S', a, \mathbf{w})$

Remark. $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$

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 $\tau = T - 1$

3 Average Reward: A New Problem Setting for Continuing Tasks

Why Average Reward

- discounting → degree, "interest" of reward
- Discounting is not necessary in every task.
- The discounted setting is problematic with function approximation.

Average Reward

Average reward is the **quality of policy** π and is defined as the average rate of reward.

$$
\begin{aligned}
r(\pi) \doteq & \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] && (10.6)\\
= & \lim_{t\to\infty} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] && (10.7)\\
= & \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) r
\end{aligned}
$$

The second and third equations hold if the steady-state distribution $\mu_\pi(s) \doteq \lim_{t\to\infty} \mathbf{Pr}\{S_t = s | A_{0:t-1} \sim \pi\}$ exists and is independent of S_0, in other words, if the MDP is **ergodic**.

The steady state distribution μ_π is the special distribution under which, if you select actions according to π, you remain in the same distribution.

$$\sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s'|s, a) = \mu_\pi(s') \tag{10.8}$$

Differential Reward

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots \tag{10.9}$$

This is known as the **differential return**, and the corresponding value functions are known as **differential value** functions ($v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$,$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$).

For value function, we remove γs and replace all rewards by "reward - true average reward".

$$
\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r - r(\pi) + v_\pi(s')],\\
q_\pi(s, a) &= \sum_{r,s'} p(s', r|s, a)[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a')],\\
v_*(s) &= \max_a \sum_{r,s'} p(s', r|s, a)[r - \max_\pi r(\pi) + v_*(s')], \text{ and}\\
q_*(s, a) &= \sum_{r,s'} p(s', r|s, a)[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a')]
\end{aligned}
$$

Let $\bar{R}_t$ be an estimate at time t of the average reward $\mathring{(}\pi)$. Differential form of the two TD errors:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t) \tag{10.10}$$

and

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}\mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \tag{10.11}$$

Average reward version of semi-gradient Sarsa:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t) \tag{10.12}$$

Differential 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 sizes $\alpha, \beta > 0$, small $\varepsilon > 0$
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:
  Take action A, observe R, S'
  Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
  $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$

Remark. For Q-learning, $ChooseA'$ as a function of $\hat{q}(S', \cdot, \mathbf{w}) \to A' = \arg\max \hat{q}(S', \cdot, \mathbf{w})$

Remark. Note that both Sarsa and Q-learning choose A' as a function of $\hat{q}$.

4 Deprecating the Discounted Setting

Discounted Setting and Average Reward

Consider an infinite sequence of returns with **no beginning or end**, and **no clearly identified states**. For example, the state space is very very large and time $t \to \infty$.

Assume all feature vectors are the same. Then there is exactly one reward sequence and performance has to be assessed purely from these.

- **average reward** $\to$ average the rewards over a long interval
- **discounting** $\to$ for each time step we could measure the discounted return; some returns would be small and some big, (why?) so again we would have to average them over a sufficiently large time interval.

Result:

For policy π, the average of the discounted returns is always $\dfrac{r(\pi)}{1-\gamma}$

The Futility of Discounting in Continuing Problems

Perhaps discounting can be saved by choosing an objective that sums discounted values over the distribution with which states occur under the policy:

$$
\begin{aligned}
J(\pi) &= \sum_s \mu_\pi(s) v_\pi^\gamma(s) && \text{(where } v_\pi^\gamma \text{ is the discounted value function)} \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \left[r + \gamma v_\pi^\gamma(s') \right] && \text{(Bellman Eq.)} \\
&= r(\pi) + \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \gamma v_\pi^\gamma(s') && \text{(from (10.7))} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s' \,|\, s, a) && \text{(from (3.4))} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \mu_\pi(s') && \text{(from (10.8))} \\
&= r(\pi) + \gamma J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 r(\pi) + \gamma^3 r(\pi) + \cdots \\
&= \frac{1}{1-\gamma} r(\pi).
\end{aligned}
$$

The proposed discounted objective orders policies identically to the undiscounted (average reward) objective. The discount rate γ does not influence the ordering!

Policy Improvement

Function approximation we have lost the policy improvement theorem. It is no longer true that if we change the policy to improve the discounted value of one state then we are guaranteed to have improved the overall policy.

5 Differential Semi-gradient n-step Sarsa

$$G_{t:t+n} \doteq R_{t+1} - \bar{R}_{t+n-1} + \cdots + R_{t+n} - \bar{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}) \quad (10.14)$$

where $\bar{R}$ is an estimate of $r(\pi)$, $n \geq 1$ and $t + n < T$. If $t + n \geq T$, then $G_{t:t+n} \doteq G_t$. Then the TD error will be

$$\delta_t \doteq G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}), \quad (10.15)$$

after which we can apply our usual semi-gradient Sarsa update (10.12).

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 sizes $\alpha, \beta > 0$, small $\varepsilon > 0$, 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 $\tau \geq 0$:
    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n} (R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$

6 Example: Mountain Car Task

Setting

- reward = -1 until the car reach the goal
- up the opposite slope and then apply full throttle
- position x_{t+1}, velocity $\dot{x_{t+1}}$ are updated by

$$x_{t+1} \doteq bound[x_t + \dot{x}_{t+1}]$$
$$\dot{x}_{t+1} \doteq bound[\dot{x}_t + 0.001A_t - 0.0025\cos(3x_t)]$$

  where *bound* operation enforce $-1.2 \leq x_{t+1} \leq 0.5$, $-0.07 \leq \dot{x_{t+1}} \leq 0.07$. When x_{t+1} reaches the left bound, $\dot{x_{t+1}} = 0$. When x_{t+1} reaches the right bound, the episode ends.

- each episode starts from $x_t \in [-0.6, -0.4)$ and zero velocity

- 8 tilings with each tile covers 1/8 distance in each dimension
- approximate:

$$\hat{q}(s,a,\mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s,a) = \sum_{i=1}^{d} w_i \cdot x_i(s,a) \tag{10.3}$$

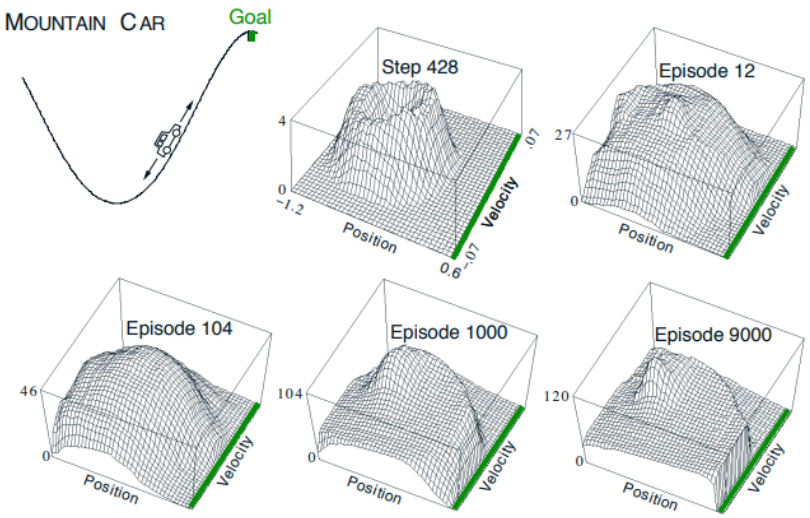

Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s,a,\mathbf{w})$) learned during one run.

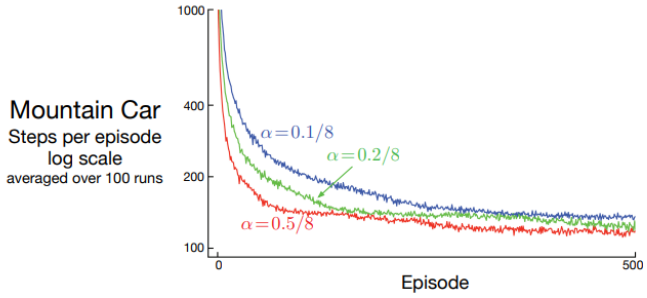

Figure 10.2: Mountain Car learning curves for the semi-gradient Sarsa method with tile-coding function approximation and ε-greedy action selection. ■

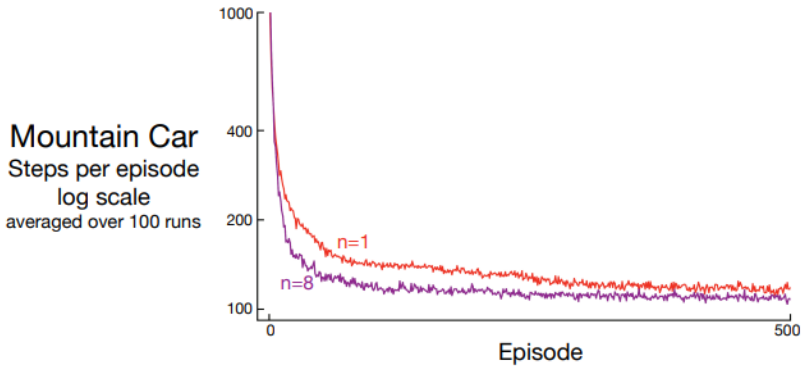

Figure 10.3: Performance of one-step vs 8-step semi-gradient Sarsa on the Mountain Car task. Good step sizes were used: α=0.5/8 for $n = 1$ and $\alpha = 0.3/8$ for $n = 8$.

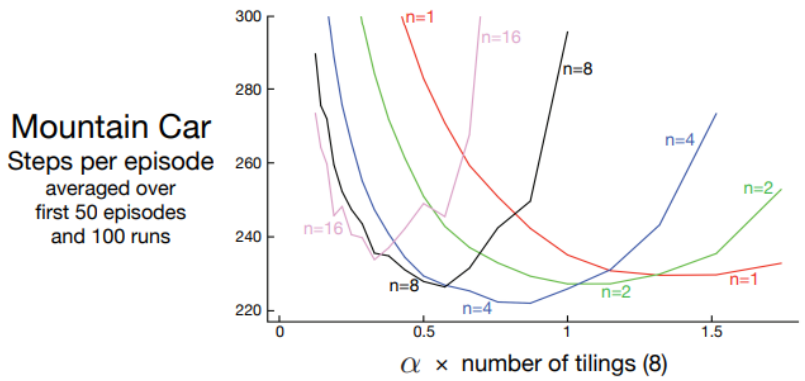

Figure 10.4: Effect of the α and n on early performance of n-step semi-gradient Sarsa and tile-coding function approximation on the Mountain Car task.